Федеральное агентство по рыболовству
БГАРФ ФГБОУ ВО «КГТУ»
Калининградский морской рыбопромышленный колледж

Утверждаю
Заместитель начальника колледжа
по учебно-методической работе

А.И.Колесниченко

Подписано цифровой подписью: Колесниченко Андрей Игоревич
Причина: Я утвердил этот документ
Дата: 25 Март 2025 г. 12:21:00
DN: E=uc_fk@roskazna.ru, S=77 Москва, ИНН ЮЛ=7710568760, ОГРН=1047797019830, STREET="Большой Златоустинский переулок, д. 6, строение 1", L=г. Москва, C=RU, O=Казначейство России, CN=Федеральное казначейство

ООД.08 ИНФОРМАТИКА

Методическое пособие для выполнения практических занятий

по специальности

2 семестр

26.02.03 Судовождение

МО–26 02 03-ООД.08.СР

РАЗРАБОТЧИКИ	Зеньков С.В., Иванова Т.Ю., Сукорская А.О.
ЗАВЕДУЮЩИЙ ОТДЕЛЕНИЕМ	В.В.Феоктистов
ГОД РАЗРАБОТКИ	2023
ГОД ОБНОВЛЕНИЯ	2025

Содержание

Тема 2.7 Моделирование в среде графических редакторов

Практическое занятие №1 Представление о моделировании в среде графических редакторов. Моделирование геометрических фигур растровой графики

Цель занятия.

1.Дать практические навыки работы с графическим редактором (GIMP). Создание геометрических фигур. Освоение инструментов рисования;

2 Формировать ОК 01, ОК 02..

*Оборудование***:** ПК, графический редактор Gimp.

Исходные данные:

Папка на РС «Практическое занятие №1 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны выполнить задания по вариантам.

Теоретический материал

Растровая графика

Растровое изображение — изображение, представляющее собой сетку пикселей или точек цветов (обычно прямоугольную) на компьютерном мониторе, бумаге и других отображающих устройствах и материалах. Важными характеристиками изображения являются:

количество пикселов — разрешение. Может указываться отдельно количество пикселов по ширине и высоте (1024*768, 640*480,…) или же, редко, общее количество пикселей (часто измеряется в мегапикселах);

количество используемых цветов или «глубина цвета» (эти характеристики имеют следующую зависимость: $N = 2^I$, где N - количество цветов, а I - глубина цвета);

цветовое пространство (цветовая модель) RGB, CMYK, XYZ, YCbCr и др.

Создается растровая графика фотоаппаратами, сканерами, непосредственно в растровом редакторе, также путем экспорта из векторного редактора или в виде скриншотов.

Достоинства

Растровая графика позволяет создать (воспроизвести) практически любой рисунок, вне зависимости от сложности, в отличие, например, от векторной, где невозможно точно передать эффект перехода от одного цвета к другому без потерь в размере файла.

Распространённость — растровая графика используется сейчас практически везде: от маленьких значков до плакатов.

Высокая скорость обработки сложных изображений, если не нужно масштабирование.

Растровое представление изображения естественно для большинства устройств ввода-вывода графической информации, таких как мониторы (за исключением векторных), матричные и струйные принтеры, цифровые фотоаппараты, сканеры.

Недостатки

Большой размер файлов с простыми изображениями.

Невозможность идеального масштабирования.

Невозможность вывода на печать на плоттер.

Из-за этих недостатков для хранения простых рисунков рекомендуют вместо даже сжатой растровой графики использовать векторную графику.

Форматы

Растровые изображения обычно хранятся в сжатом виде. В зависимости от типа сжатия может быть возможно или невозможно восстановить изображение в точности таким, каким оно было до сжатия (сжатие без потерь или сжатие с потерями соответственно). Так же в графическом файле может храниться дополнительная информация: об авторе файла, фотокамере и её настройках, количестве точек на дюйм при печати и др.

BMP или WindowsBitmap — обычно используется без сжатия, хотя возможно использование алгоритма RLE.

GIF (GraphicsInterchangeFormat) — устаревающий формат, поддерживающий не более 256 цветов одновременно. Всё ещё популярен из за

поддержки анимации, которая отсутствует в чистом PNG, хотя ПО начинает поддерживать APNG.

PCX устаревший формат, позволявший хорошо сжимать простые рисованые изображения (при сжатии группы подряд идущих пикселов одинакового цвета заменяются на запись о количестве таких пикселов и их цвете).

PNG (PortableNetworkGraphics) .

JPEG очень широко используемый формат изображений. Сжатие основано на усреднении цвета соседних пикселей(информация о яркости при этом не усредняется) и отбрасывании высокочастотных составляющих в пространственном спектре фрагмента изображения. При детальном рассмотрении сильно сжатого изображения заметно размытие резких границ и характерный муар вблизи них.

TIFF поддерживает большой диапазон изменения глубины цвета, разные цветовые пространства, разные настройки сжатия (как с потерями, так и без) и др.

RAW хранит информацию, непосредственно получаемую с матрицы цифрового фотоаппарата или аналогичного устройства без применения к ней каких-либо преобразований, а также хранит настройки фотокамеры. Позволяет избежать потери информации при применении к изображению различных преобразований (потеря информации происходит в результате округления и выхода цвета пиксела за пределы допустимых значений). Используется при съёмке в сложных условиях (недостаточная освещённость, невозможность выставить баланс белого и т.п.) для последующей обработки на компьютере (обычно в ручном режиме). Практически все полупрофессиональные и профессиональные цифровые фотоаппараты позволяют сохранять RAW изображения. Формат файла зависит от модели фотоаппарата, единого стандарта не существует

К программным средствам обработки растровой графики относятся растровые графические редакторы: GIMP, Paint.NET, TuxPaint, AdobePhotoshop, AdobeFireworks, CorelPhoto-Paint, CorelPaintShopPro, CorelPainter, MicrosoftPaint.

GIMP — многоплатформенное программное обеспечение для работы над изображениями. GIMP является акронимом, означающим **GNU ImageManipulationProgram.** Редактор GIMP пригоден для решения множества задач по изменению изображений, включая ретушь фотографий, объединение и созданий изображений. Программа GIMP многофункциональна. Еѐ можно

использовать как простой графический редактор, как профессиональное приложение по ретуши фотографий, как сетевую систему пакетной обработки изображений, как программу для рендеринга изображений, как преобразователь форматов изображения и т.д. GIMP спроектирован расширяемым при помощи дополнений, реализующих любые возможные функции. Передовой интерфейс для разработки сценариев позволяет легко автоматизировать выполнение любых задач любого уровня.

Одной из сильных сторон GIMP является его доступность из многих источников для многих операционных систем. GIMP входит в состав большинства дистрибутивов GNU/Linux. GIMP также доступен и для других операционных систем вроде MicrosoftWindows™ или Mac OS X™ от Apple (Darwin). GIMP — свободное программное обеспечение, выпускаемое под

лицензией GPL(GeneralPublicLicense). GPL предоставляет пользователям право доступа к исходному коду программ и право изменять его. **Краткий обзор возможностей и функций GIMP**

Полный набор инструментов для обработки растровой графики

Возможность работы с векторной графикой

Создание анимации

Работа с принтером и сканером

Захват изображений

Множество подключаемых модулей (plug-in)

Быстрое создание различных логотипов для web-дизайна

и многое другое...

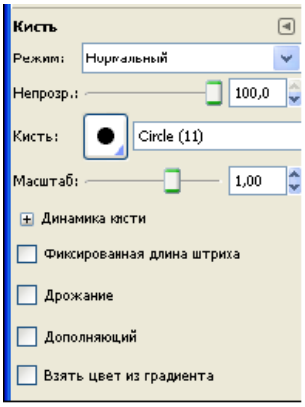
Основное диалоговое окно GIMP

Основное окно состоит из нескольких основных элементов: **инструментов и диалога цвета.** Инструменты позволяют производить определенные действия над **уже открытым** изображением. Свойства любого инструмента можно вызвать двойным щелчком на его иконке.

Диалог цвета позволяет выбрать типы воздействия инструментов. Так, диалог цвета позволяет выбрать цвет пера и фона, а так же переключать их, нажав на стрелочки.

Для рисования в нашем распоряжении есть *Карандаш*, *Кисть*, *Ластик*, *Аэрограф*, *Штамп*, *Размыватель*, *Чернила*, *Осветление* и *Палец*. Инструменты Карандаш, Кисть, Ластик, Аэрограф чувствительны к размеру и виду кисти. Выбрать их можно в диалоге

Кисти



Кисть также может работать и в других режимах: **Добавление** (обратный режиму вычитания), **Осветление** (операция деления) или **Затемнение** (операция умножения). С ее помощью вы также сможете изменять тон и яркость

изображения. Есть возможность изменять размер кисти, ее жесткость, непрозрачность и цвет в зависимости от скорости движения по холсту, в зависимости от силы нажима, да и просто кисть может изменять свои параметры случайно. В этих же режимах работают Карандаш, Аэрограф, Чернила.

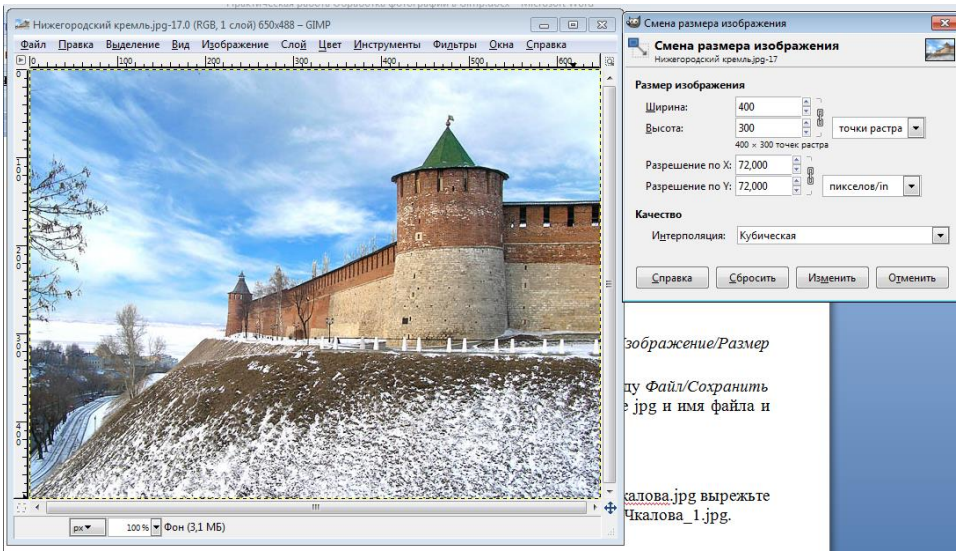
Пример варианта задания

Задание 1. Изменение размеров изображения. У изображения Нижегородский кремль.jpg изменить размеры, установив размер 400х300 и сохранив результат под именем Нижегородский кремль _1.jpg.

Алгоритм

Запустить программу Gimp.

Для изменения размеров изображения выполните команду *Изображение/Размер изображения*, интерполяция – *Кубическая*, нажмите *Изменить.*



Сохраните рисунок как Город_1.jpg. Для этого выполните команду *Файл/Сохранить как ….* В появившемся диалоговом окне выберите расширение jpg и имя файла и нажмите кнопку *Сохрани,* качество - 85.

Закройте рисунок.

Задание 2. Кадрирование изображения. Из изображения Памятник_Чкалова.jpg вырежьте памятник и сохраните результат под именем Памятник_Чкалова_1.jpg.

Алгоритм

Загрузить файл Памятник_Чкалова.jpg.jpg.

Для выполнения кадрирования выберите инструмент *Кадрирование* и выделите прямоугольную область памятника.

Сохраните рисунок.

Задание 3. Поворот изображения. Фотографию Пизанская башня.jpg приведите в порядок – выпрямите башню и сохраните под именем Пизанская башня_1.jpg.

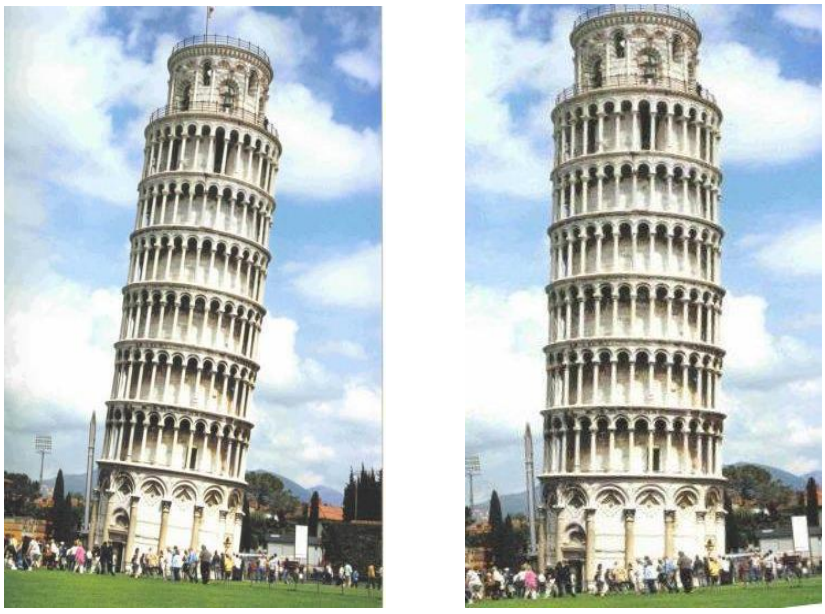
Алгоритм

Загрузить файл Пизанская башня.jpg.

Для выполнения поворота выполните команду *Инструмент/Инструменты преобразования/Вращение,* угол вращения -7 градусов.

Кадрируйте полученное изображение.

Сохраните рисунок как Пизанская башня_1.jpg.



Содержание отчета:

1. Наименование практического занятия

2. Цель занятия

3. Вариант задания

4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

5. Список используемых источников

6. Выводы и предложения

7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1.Расшифруйте аббревиатуру GIMP.

2.Перечислите возможности редактора GIMP.

3.Перечислите основные компоненты диалогового окна GIMP.

4.Что из себя представляет окно изображения?

5.Перечислите основные компоненты панели инструментов.

Практическое занятие №2 Моделирование в векторном редакторе. Работа с объектами векторного редактора

Цель занятия:

1. Познакомить с объектами в программе INKSCAPE;
2. Изучить графический интерфейс и панель инструментов программы
3. INKSCAPE;
4. Научить строить графические примитивы;
5. Формировать ОК 01, ОК 02.

Исходные данные:

Папка на РС «Практическое занятие №2 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия (Электронный учебник INKSCAPE в папке)

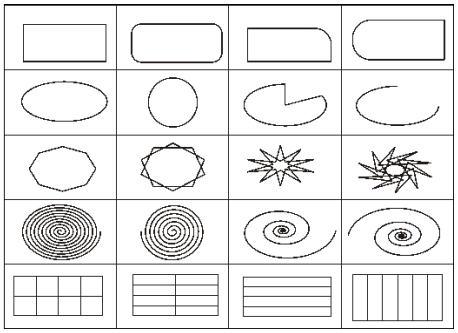
2. Обучающие должны выполнить задания по вариантам.

Пример варианта задания

Задание №1

Открыть программу Мой компьютер-Общие документы-Мои рисунки-INKSCAPE

Используя инструменты Прямоугольник, Эллипс, Спираль, Полигон и Миллиметровка, построить следующие изображения:



Задание №2

С помощью простых фигур создайте следующее изображение:

Выводы и предложения проделанной работы:
Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант занятия
4. Результат работы сохранить файлом в своей папке
5. Список используемых источников
6. Выводы и предложения

7. Даты и подписи курсанта и преподавателя

Вопросы для самопроверки:

1. Что является объектом в программе INKSCAPE
2. Структура объекта INKSCAPE
3. Как выполняется копирование объектов
4. Как выполняется окрашивание объектов
5. Как выполняется объединение объектов в группу

Практическое занятие №3 Моделирование в векторном редакторе. Закраска рисунков и контуров

Цель занятия:

1.Продолжить знакомство с векторным редактором Inkscape;

2. Научить работать с основными инструментами закраски и операциями над контурами;

2. Формировать ОК 01, ОК 02.

Оборудование: ПК, векторный редактор Inkscape

Исходные данные:

Папка на РС «Практическое занятие №3 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия (Электронный учебник INKSCAPE в папке)

2. Обучающие должны выполнить задания по вариантам.

Теоретический материал

Векторная графика

Векторное изображение - это графический объект, построенный из геометрических примитивов, таких как точки, линии, сплайны и многоугольники.

Рассмотрим, к примеру, такой графический примитив, как окружность радиуса r. Для её построения необходимо и достаточно следующих исходных данных:

координаты центра окружности;

значение радиуса r;

цвет заполнения (если окружность не прозрачная);

цвет и толщина контура (в случае наличия контура).

Преимущества

Размер, занимаемой описательной частью, не зависит от реальной величины объекта, что позволяет, используя минимальное количество информации, описать сколько угодно раз большой объект файлом минимального размера.

В связи с тем, что информация об объекте хранится в описательной форме, можно бесконечно увеличить графический примитив, например, дугу окружности, и она останется гладкой. С другой стороны, если кривая представлена в виде ломаной линии, увеличение покажет, что она на самом деле не кривая.

Параметры объектов хранятся и могут быть легко изменены. Также это означает что перемещение, масштабирование, вращение, заполнение и т. д. не ухудшат качества рисунка. Более того, обычно указывают размеры в аппаратно-независимых единицах (англ. device-independentunit), которые ведут к наилучшей возможной растеризации на растровых устройствах.

При увеличении или уменьшении объектов толщина линий может быть задана постоянной величиной, независимо от реального контура.

Недостатки

Не каждый объект может быть легко изображен в векторном виде — для подобного оригинальному изображению может потребоваться очень большое количество объектов и их сложности, что негативно влияет на количество памяти, занимаемой изображением, и на время для его отображения (отрисовки).

Перевод векторной графики в растр достаточно прост. Но обратного пути, как правило, нет — трассировка растра, при том что требует значительных вычислительных мощностей и времени, не всегда обеспечивает высокое качества векторного рисунка.

Форматы векторной графики: .cdr, .ai, .cmx, .eps, .fla, .svg, .swf, .wmf.

К программным средствам создания и обработки векторной графики относятся следующие ГР: CorelDraw, AdobeIllustrator, а также векторизаторы (трассировщики) - специализированные пакеты преобразования растровых изображений в векторные.

Как в растровой, так и в векторной графике необходим способ кодирования цвета.

Графический редактор Inkscape обладает достаточным набором инструментов для создания иллюстраций и довольно удобным интерфейсом. Он представляет собой программу ориентированую на решение конкретной задачи – создание иллюстративной графики.

Иллюстративная графика – это прикладная ветвь машинной график, сравнительно недавно выделившаяся в отдельное направление наряду с графикой деловой, инженерной и научной. К области иллюстративной графики относятся в первую очередь рисунки, коллажи, рекламные объявления, заставки, постеры – всё, что принято называть художественной продукцией. Объекты иллюстративной графики отличаются от объектов других прикладных областей своей первичностью – они не могут быть построены автоматически по некоторым исходным данным, без участия художника или дизайнера.

Основные принципы работы

Перед началом работы с Inkscape нужны общие представления о возможностях этой программы. Основным понятием в любом редакторе векторной графики, является понятие **объекта.** Работа над любой иллюстрацией заключается в создании объектов, их редактировании и расположении в нужных местах. При этом сначала создается приблизительная форма объектов, после чего форма уточняется путем добавления, удаления и перемещения узлов контура. После создания необходимой формы задается цвет контура и выбирается заливка. В этом редакторе можно создавать как стандартные фигуры (прямоугольники, эллипсы, многоугольники, и т.д.), так и произвольные, состоящие из прямых и кривых линий.

Средства работы с текстом позволяют создавать небольшие текстовые документы, оформленные рисунками.

Применение различных эффектов помогает создать красивое изображение из простых объектов. Каждый рисунок состоит из одной или нескольких фигур, которые могут накладываться и полностью или частично закрывать друг друга.

Итак, основные приемы работы с Inkscape:

Создание простых геометрических фигур или произвольных кривых и ломаных, замкнутых или разомкнутых;

Редактирование любого объекта: изменение цвета контура и заливки, изменение формы объекта;

Размещение всех объектов в нужных местах, определение порядка взаимного перекрытия объектов;

Вставка и форматирование текста;

Вставка готовых картинок или ранее созданных иллюстраций в документ.

Понятие объекта в Inkscape

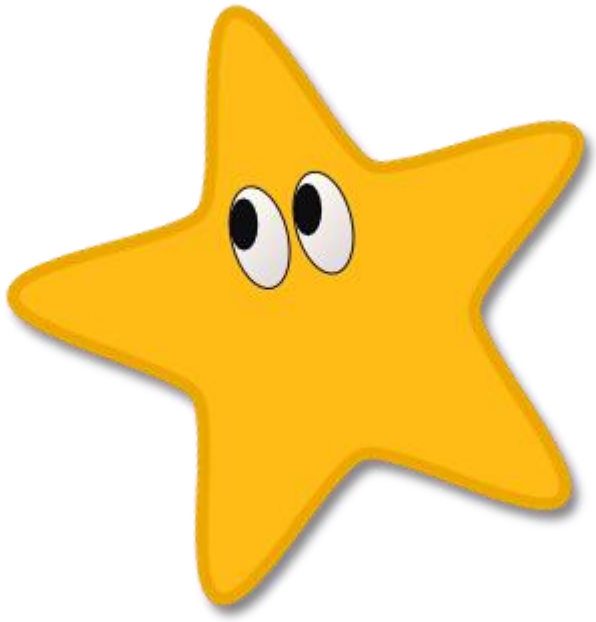
Любое изображение в векторном формате состоит из множества составляющих частей, которые редактируются независимо друг от друга. Главными «кирпичиками», из которых составляется изображение, являются **объекты. Объектом** называется элемент изображения: прямая, круг, прямоугольник, кривая, замкнутая кривая, многоугольник и другие.

Любой векторный объект Inkscape имеет ряд общих характеристик. Каждый объект состоит из некоторого количества точек или узлов, соединенных прямыми или кривыми линиями — **сегментами**. Координаты узлов и параметры сегментов определяют внешний вид объекта. Сегменты объекта образуют контур, который имеет свой цвет и толщину. Область внутри объекта можно закрасить или залить одним цветом, смесью цветов или узором. Эту область принято называть заливкой. У одного объекта не может быть различных заливок или соединительных линий разной толщины и цвета. Для создания сложных изображений требуется использовать множество объектов.

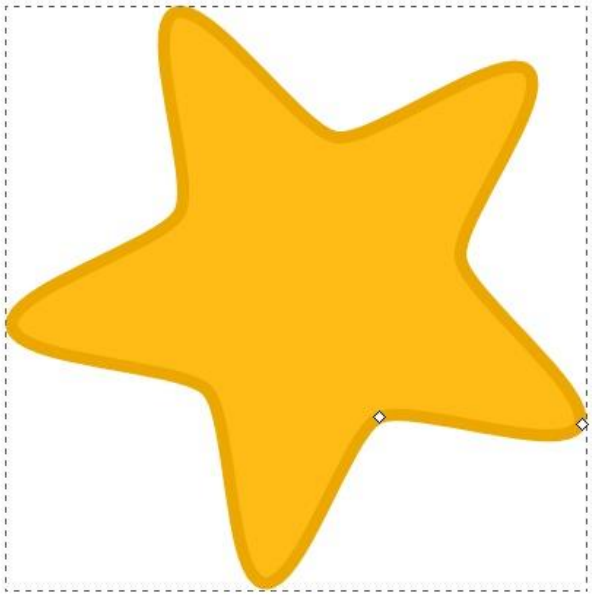
Важными объектами Inkscape являются плавно изогнутые **кривые**, с помощью которых можно построить любой произвольный контур. Они называются **кривыми Безье**. Математик Пьер Безье (Pierre Bezier) открыл, что произвольную кривую можно задать с помощью двух векторов, находящихся в начале и конце кривой. Это положение легло в основу описания кривых Безье в Inkscape. Кроме положения начальной и конечной точки (то есть узлов кривой), внешний вид кривой определяется кривизной, то есть ее изогнутостью между двумя узлами. Кривизна определяется двумя параметрами кривой в каждом узле, которые графически представлены с помощью отрезков, выходящих из узлов. Эти отрезки называются **манипуляторами кривизны.**

Пример варианта задания

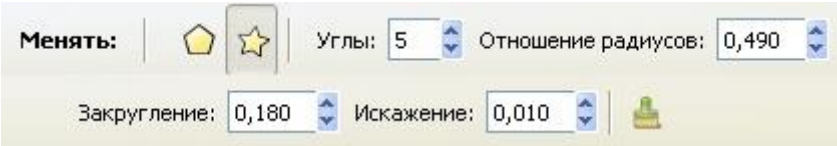
ЗВЕЗДОЧКА С ГЛАЗАМИ



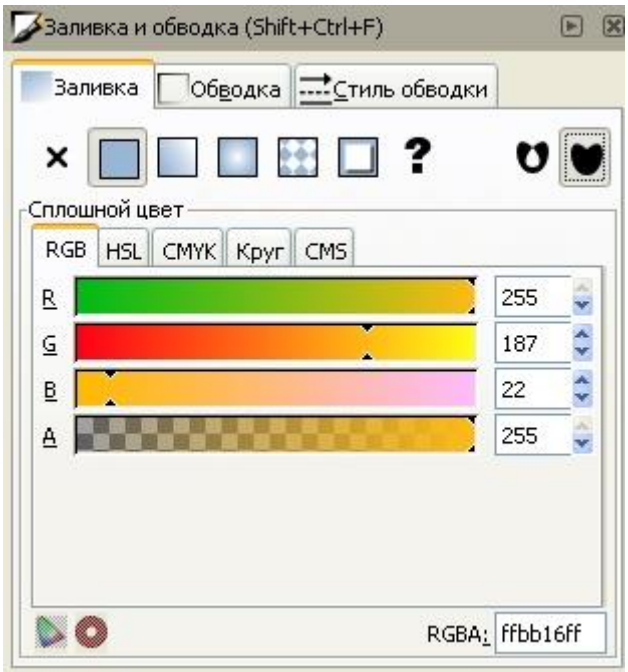
1 Возьмите инструмент для рисования звезд и просто создадим с его помощью вот такую пятиугольную звезду.

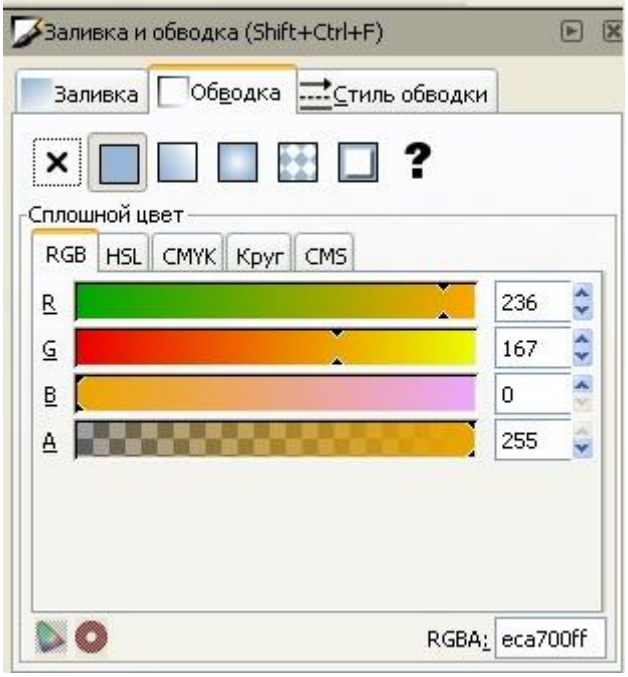


Ниже приведен скриншот контекстной панели инструмента звезды. Параметры, указанные на этом скриншоте, как раз отвечают за скругление углов, количество лучей и т.д. Подробнее можно прочитать в разделе инструкция с описанием этого инструмента.

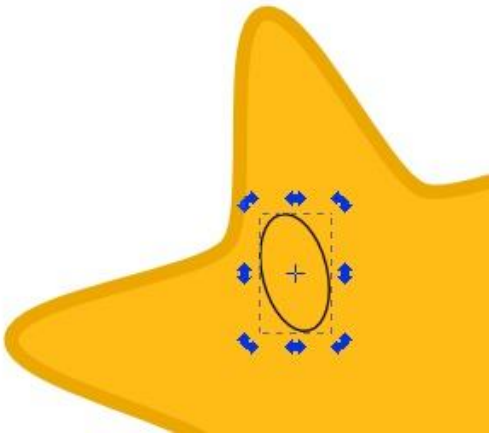


На двух следующих рисунках приведены параметры для цвета заливки звезды и параметры для цвета обводки. Это окошко открывается по комбинации клавиш **Ctrl+Shift+F** и будет активным для редактирования, если нарисованная вами звездочка выделена, т.е. сама активна. Толщину обводки можно поменять на третьей закладке.

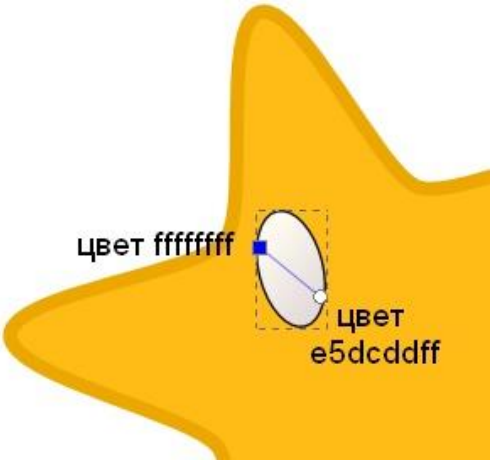




2 Теперь нарисуем глазик. Для этого нам понадобится инструмент эллипс. На скриншоте ниже как раз показан нарисованный нами эллипс для будущего глаза звезды. Этот эллипс пока без заливки, но уже с нужной толщиной обводки в 1 пиксель черного цвета. Поворачивать, перемещать и изменять размер эллипса можно с помощью инструмента инструмента выделения и трансформации. Подробнее можно прочитать в разделе инструкция с описанием этого инструмента.



Сделаем заливку для эллипса глаза в виде линейного градиента. Расположение направляющей градиента показано на рисунке. Если направляющая у вас сразу не видна, после того как на закладке заливки вы выбрали тип линейный градиент, то активируйте инструмент градиент в боковом окне инструментов и все появится. Если щелкнуть инструментом градиент на крайние точки направляющей, то можно задать их цвета. Цвета точек направляющей градиента показаны на рисунке ниже.



Теперь, когда белок глаза готов, нарисуем еще один овал, который будет абсолютно черным. Сделать черную заливку и обводку не должно составить у вас

труда. Расположите второй овал-зрачок так, как вам нравится, и смотря какую эмоцию звездочке вы хотите придать. Как сделали мы, видно на рисунке ниже. Теперь сгруппируйте оба овала, что бы они стали одним целым. Для этого выделите их оба и нажмите комбинацию клавиш **Ctrl+G**.

3 Теперь можно легко сделать второй глазик. Для этого надо сделать копию первого. Т.е. продублировать его. Продублировать объект можно по комбинации клавиш **Ctrl+D**. При этом вы не заметите визуальной разницы, т.к. копия объекта располагается прямо поверх копируемого. Но теперь вы можете сдвинуть ее мышью или стрелочками клавиатуры и увидите, как под ней будет появляться точно такой же объект. Расположите правильно второй глазик звезды. Рисунок готов.

Можно добавить звезде тень. Сделать это можно в меню "Фильтры" - "Свет и тень" - "Отбрасывать тень".

Содержание отчета:

1. Наименование практического занятия

2. Цель занятия

3. Вариант задания

4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

5. Список используемых источников

6. Выводы и предложения

7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Для чего предназначены векторные графические редакторы?

2.В чем заключаются основные отличия векторных изображений от растровых?

3.В каких сферах деятельности векторные изображения нашли наиболее широкое применение?

4.Что является элементарным объектом векторной графики?

Практическое занятие №4 Моделирование в программе Dia

Цель занятия :

1. Познакомить с интерфейсом программы DIA проектирование и моделирование компьютерной сети

2. Дать навык графического ввода и редактирования объектов библиотеки Dia.

3. Научитьмоделировать компьютерные сети в программе Dia.

4. Формировать ОК 01, ОК 02.

Исходные материалы и данные:

ПК, или DIA

Использованные источники: [Электронный учебник, Dia]

Исходные данные:

Папка на РС «Практическое занятие №4 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия (Электронный учебник Dia в папке)

2. Обучающие должны выполнить задания по вариантам.

Теоретическая часть

Dia – бесплатный редактор для создания диаграмм и схем. Коммерческим аналогом этой программы является продукт компании Microsoft – Visio.

помощью Dia возможно создание многих видов структурированных диаграмм и схем, в том числе:

- блок-схемы;
- диаграммы UML;
- сетевые диаграммы;
- ER-диаграммы (проектирование баз данных);
- упрощенные схемы электрических цепей и другие.

В программе поддерживается множество языков и региональных стандартов, среди прочих есть русский и украинский.

Dia позволяет экспортировать данные в более чем 25 форматов векторной и растровой графики, в том числе векторные SVG, DXF, FIG, VDX (MS Visio), PDF и растровые рисунки BMP, GIF, JPG, PNG, TIF. «Родной» формат программы - Dia Native Diagram (DIA).

Интерфейс у Dia простой, создать диаграмму достаточно просто даже пользователю, впервые работающему с программой. До версии 0.97 панель инструментов и рабочая область располагаются в отдельных окнах. Это так называемый однодокументный интерфейс (CSDI). При работе с несколькими файлами для каждого открывается отдельное окно, а панель инструментов в этом случае постоянно находится поверх остальных окон. Поначалу это непривычно, но в процессе использования оказывается вполне удобно.

Dia предоставляет на выбор пользователя большой набор геометрических фигур, библиотеку клипартов, электрические схемы, пиктограммы по

компьютерным сетям Cisco, а также кибернетические, гидравлические, логические и многие другие символы.

Среди доступных возможностей можно выделить рисование кривых Безье, поддержку слоев, поиск элементов схемы, введение новых символов, определяемых в XML-файлах с помощью подмножества тегов SVG для изображения фигур, загрузка и сохранение диаграммы в своем XML-формате.

Панель инструментов Dia

Панель содержит все необходимые инструменты для создания диаграмм и схем. Ниже указано назначение каждой кнопки панели инструментов Dia.

1 – выделение объекта

2 – увеличение масштаба

3 – перемещение по документу/диаграмме

4 – вставка/редактирование текста

5 – создание прямоугольного объекта, рамка

6 – создание эллипса

7 – многоугольник

8 – Безье-угольник (фигура создается из кривых)

9 – линия

10 – дуга

11 – зигзаг

12 – ломанная

13 – кривая Безье

14 – вставка фонового изображения

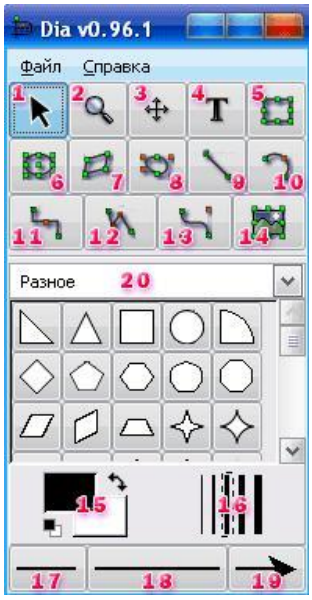


Рис. 1. Панель инструментов Dia

15 – выбор цвета линий и фона объекта

16 – толщина линий

17 – стиль начала отрезка

18 – стиль основной линии

19 – стиль конца отрезка

20 – список библиотеки элементов

Настройки Dia

Привязка к сетке

В рабочей области отображается сетка. Для того, чтобы при создании объекта (например, рисование рамки) выполнялась автоматическая подгонка к узлам сетки, необходимо в меню Вид выбрать пункт Выравнивать по сетке.

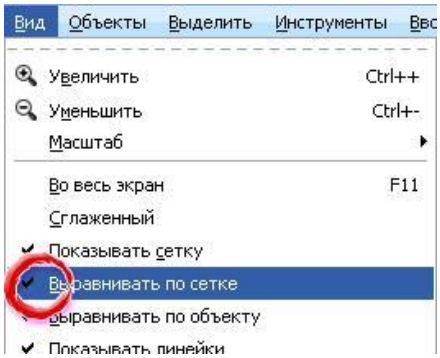


Рис. 2. Выравнивание по сетке

Теперь, когда мы рисуем или перемещаем рамку, ее углы автоматически «*прилипают*» к узлам сетки. И в этом случае расположить объект между линиями сетки будет невозможно.

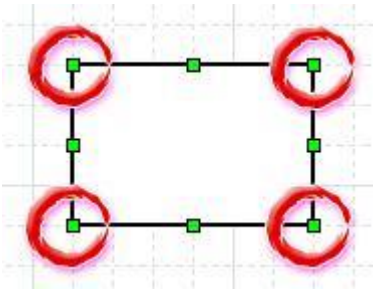


Рис. 3. Прилипание углов объектов к узлам сетки ***Внешний вид сетки***

меню *Диаграмма* выбираем пункт *Свойства*. В окне есть две закладки: *Сетка*: настройка расстояния между линиями. Снять галочку «*Динамическая сетка*» и ввести интервал по x и y. Чтобы увидеть изменение сетки без закрытия окна, достаточно нажать кнопку Применить.

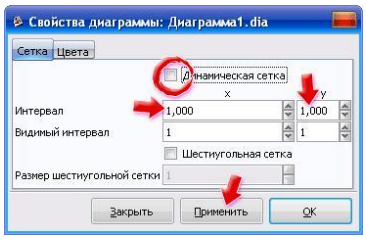


Рис. 4. Настройка сетки диаграммы

Цвет: вторая вкладка позволяет настроить цвет фона в диаграмме Dia, цвет линий сетки и цвет границ страницы.

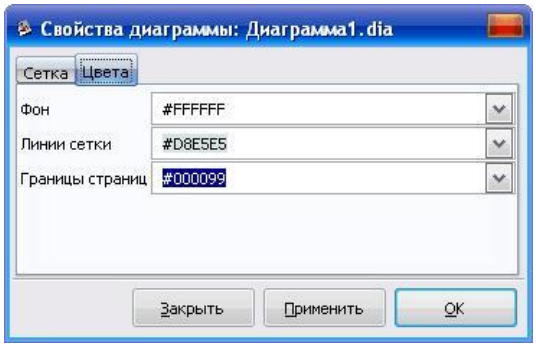


Рис. 5. Настройка цветов в диаграмме

Настройка страницы

В меню *Файл (окно Диаграмма)* выберите пункт *Настройка* страницы... В окне можно выбрать ориентацию страницы и размер бумаги, указать отступы и масштабирование.

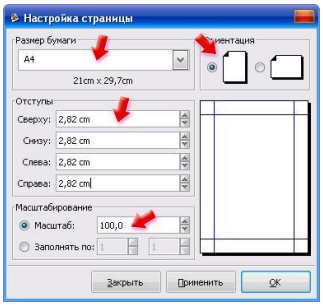


Рис. 6. Настройка страницы

Изменение объектов в библиотеках

Мы уже говорили о том, что в программе Dia доступно несколько библиотек с самыми различными типами объектов. Есть возможность и переносить элементы из одной библиотеки в другую. Это очень удобно, если требуется создать свою библиотеку, в которой будут собраны самые необходимые элементы. В меню *Файл* выберите пункт *Категории* и объекты... (или нажмите клавишу F9).

Рис. 7. Категории и объекты

Здесь мы выбираем библиотеки, между которыми необходимо переместить элементы. Указываем элемент и жмем кнопку Копировать. В результате он переносится в другую библиотеку.

При желании можно создать и новую библиотеку, для этого предусмотрена кнопка Создать.

Кнопки Вверх, Вниз предназначены для изменения позиции элемента в списке. Так, нажатие кнопки Вверх перемещает элемент на одну позицию влево (ближе к началу).

После настройки библиотеки необходимо подтвердить изменения кликом на кнопке Применить.

напоследок, название любого элемента можно изменить нажатием кнопки Редактировать.

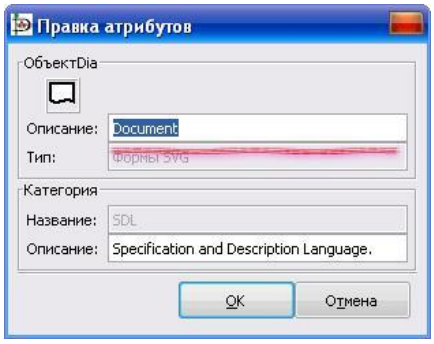


Рис. 8. Правка атрибутов

Структура элементов в диаграмме Dia

Чтобы увидеть список всех элементов, созданных в диаграмме, заходим в меню Файл и выбираем пункт Дерево диаграммы... (или жмем клавишу F8).

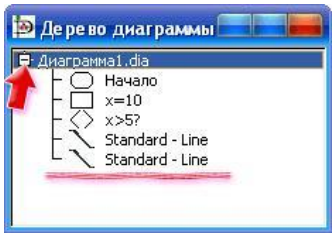


Рис. 9. Дерево диаграммы

В списке отображены значки элементов и текстовые подписи к ним. Двойной клик на элементе в списке позволяет выделить соответствующий объект на диаграмме.

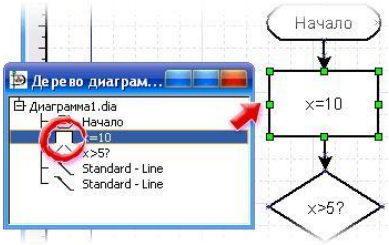


Рис. 10. Выделение объектов через дерево диаграммы

Кроме того, правый клик в списке открывает контекстное меню с дополнительными возможностями, применяемыми к элементам диаграммы Dia.

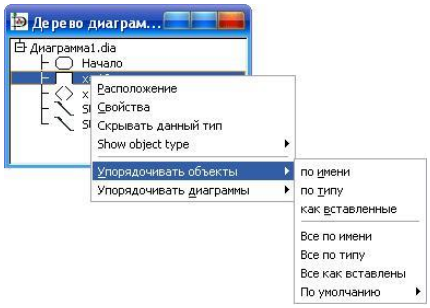


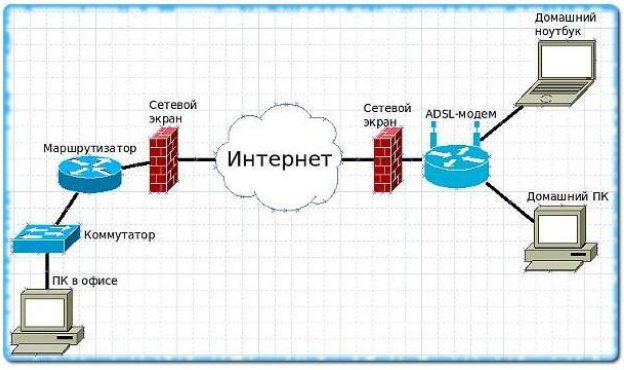
Рис. 11. Контекстное меню объектов диаграммы

Здесь можно открыть окно свойств для выбранного элемента, скрывать все элементы данного типа и еще одна удобная функция - упорядочить все элементы по различным критериям.

Пример варианта задания

Создание схем компьютерных сетей

Создайте новый документ и поместите там следующую схему сети (используйте библиотеки «Cisco - Сеть», «Cisco — Компьютеры»):



1. Сохраните созданную схему.

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Для чего используется программа Dia ?
2. С каким расширением сохраняются файлы , выполненные в этой программе?
3. Назовите управляющие кнопки в окне приложения Dia?
4. Основные возможности программы Dia. Современные версии программного продукта?
5. Состав программного пакета Dia. Основные библиотеки?
6. Интерфейс программного комплекса Dia. Рабочее окно и основные меню программы?
7. Как производятся основные операции по проектированию схемы компьютерной сети?

Раздел 3 Разработка web-сайта. Язык разметки гипертекста HTML
Тема 3.1 Моделирование web-страницы

Практическое занятие №5. Структура HTML-документа. Понятие и виды тегов. Форматирование текстовой информации.

Цель занятия :

1. Познакомить с языком разметки страниц HTML;

2. Познакомить с основными тегами, используемы в HTML;

3. Научиться создавать HTML код;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Использованные источники: [https://htmlbook.ru/samhtml- электронный ресурс]

Исходные данные:

Папка на РС «Практическое занятие №5 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Теоретический материал

HTML по праву считается самым базовым языком веб-разработки, так как его используют для структурирования и отображения страниц сайтов и всего размещенного на них контента. Таким образом, HTML-код выступает в роли инструкции для браузера, помогая ему понять, как правильно показать содержимое веб-страницы. HTML, хоть он считается не полноценным языком программирования, а лишь языком гипертекстовой разметки, но он позволяет браузерам преобразовывать гипертекст в различные элементы контента, делая сайты удобными, понятными и привлекательными для их посетителей.

Технология гипертекстовых документов, связанных между собой гиперссылками, была представлена им в 1989 году британским ученым – Тимом Бернерсом-Ли .Первая версия языка гипертекстовой разметки HTML была опубликована в 1991 году и содержала 18 структурных элементов-дескрипторов, получивших название «теги» (tags). Они обеспечили максимально простой и лаконичный дизайн языка, который со временем дополнялся и усложнялся. Помимо тегов, разметка HTML содержит их модификаторы, называемые атрибутами. На сегодняшний день

существует 140 различных тегов и атрибутов HTML, некоторые из них уже устарели и не поддерживаются современными браузерами.

В основе языка гипертекстовой разметки HTML находится комбинация тегов, которые указывают браузеру, как правильно преобразовать HTML-код в визуальные объекты на страницах сайта. Они представляют собой парную (как правило) конструкцию, при помощи которой можно задать значения тексту или другим данным, находящимся внутри нее.

Плюсы языка HTML

HTML имеет немало преимуществ, которые обеспечили ему популярность и широкую сферу применения на протяжении нескольких десятилетий. Фактически, на этом языке создавался весь современный интернет. К основным плюсам HTML можно отнести такие тезисы:

- Его код выполняется во всех распространенных интернет-браузерах, включая Google Chrome, Mozilla Firefox, Internet Explorer, Opera, Safari и т.д. Писать же HTML-код можно, по сути, в любом текстовом редакторе, хоть в обычном блокноте.
- Комбинация из HTML, CSS и JavaScript позволяет создавать полноценные сайты не только с голым текстовым контентом, но еще с дизайном (в CSS прописываются отступы, выравнивания, позиционирование, прозрачность, границы, тени и многое другое), скриптами (например, для обработки действий пользователей и мультимедийным контентом (есть встроенный аудио-видеоплеер). При помощи HTML можно обмениваться данными с другими сайтами, но для их обработки понадобится PHP.
- Язык имеет очень простой синтаксис, хорошо упорядоченную и последовательную разметку. Благодаря этому, он очень легко учится – освоить его базовую часть с нуля и научиться создавать простые сайты на HTML можно буквально в течение часа.
- Язык полностью бесплатен и находится в открытом доступе.
- Массово используется как профессиональными веб-разработчиками, так и рядовыми пользователями, оптимально подходит для создания простых любительских сайтов.
- Официальные стандарты языка HTML разрабатываются и обновляются консорциумом World Wide Web (W3C).

- Легко интегрируется с другими популярными языками веб-программирования, включая Node.js, PHP и т.д.
- Большой выбор шаблонов для админ-панелей и других инструментов для разработки и управления сайтами на HTML. Многие из них предоставляют широкое разнообразие макетов, более 1000 HTML-страниц и компонентов, а также сотни плагинов и расширений.

Вместе с тем, у HTML есть и определенные минусы, которые также нельзя обойти стороной. К ним относят такие факты:

- HTML не является полноценным языком программирования, так как в нем нет условий, функций, переменных, операторов и других элементов, необходимых для разработки программ и приложений. Вместо этого, язык содержит только набор тегов, помогающих браузеру правильно отображать содержимое веб-страниц.
- В основном, подходит только для разработки статических веб-страниц. Для добавления динамических элементов требуется подключение других языков веб-разработки: JavaScript или PHP.
- Не позволяет разработчикам реализовать логику, поэтому каждую страницу сайта на HTML нужно создавать с нуля отдельно, даже если все они используют одни и те же элементы: заголовки, колонтитулы и т.д.

Современные сайты давно уже не разрабатываются на чистом HTML, однако этот язык по-прежнему незаменим в процессе их верстки. Связка HTML+CSS не устарела и в 2023 году, это очень мощный и функциональный инструмент, позволяющий создавать красивые и удобные веб-интерфейсы.
Таким образом, хотя бы базовое знание HTML является очень полезным навыком для веб-разработчика и веб-дизайнера. Впрочем, владение языком разметки гипертекста будет нелишним и для интернет-предпринимателей.

Пример варианта задания

Приведен несложный пример HTML кода.

«**Первая веб-страница**»

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
```

```
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 <title>Моя первая веб-страница</title>
</head>
<body>

 <h1>Заголовок страницы</h1>
 <p>Основной текст.</p>

</body>
</html>
```

Чтобы посмотреть результат примера в действии, проделайте следующие шаги.

1. В Windows откройте программу Блокнот (Пуск > Выполнить > набрать «notepad» или Пуск > Программы > Стандартные > Блокнот).

2. Наберите или скопируйте код в Блокноте (рис. 1.1).

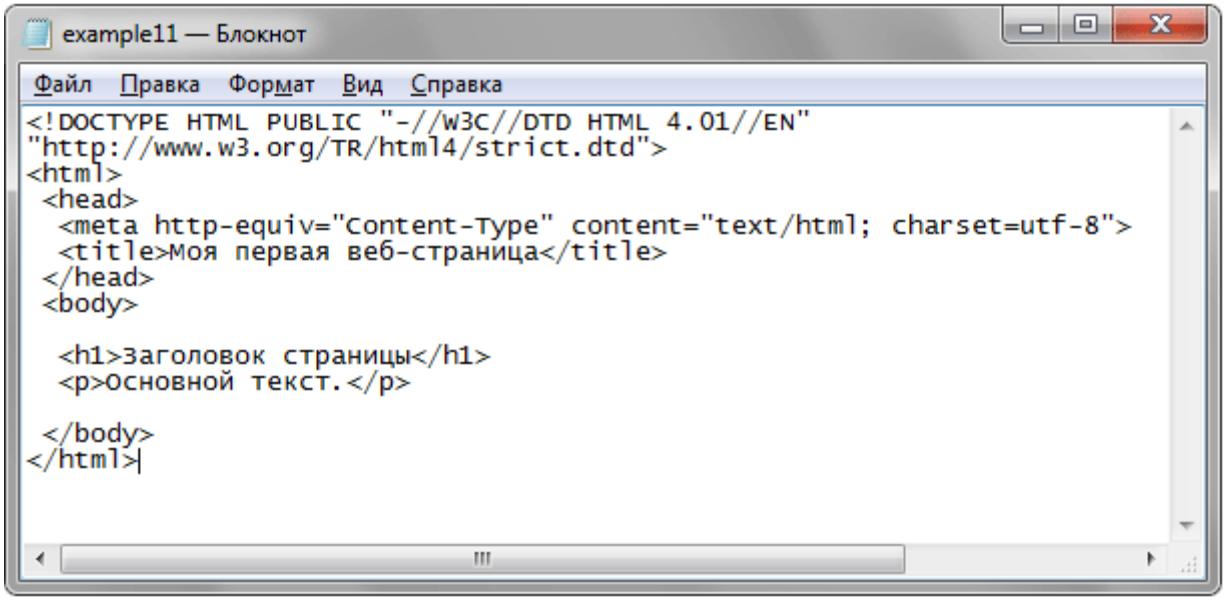


Рис. 1.1. Вид HTML-кода в программе Блокнот

Рис. 1.1. Вид HTML-кода в программе Блокнот

3. Сохраните готовый документ (Файл > Сохранить как...) под именем c:\www\example11.html, при этом обязательно поставьте в диалоговом окне сохранения тип файла: Все файлы и кодировку UTF-8 (рис. 1.2). Обратите внимание, что расширение у файла должно быть именно html.

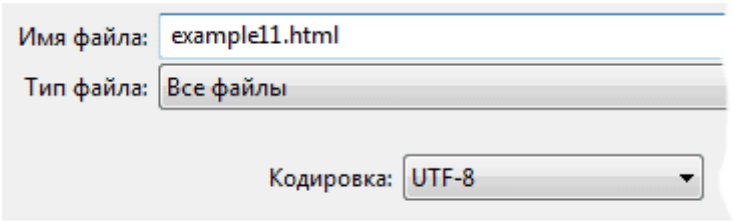


Рис. 1.2. Параметры сохранения файла в Блокноте

4. Запустите браузер Internet Explorer или Google Chrome

5. В браузере выберите пункт меню Файл > Открыть и укажите путь к вашему файлу.

6. Если все сделано правильно, то в браузере вы увидите результат, как показано на рис. 1.3.

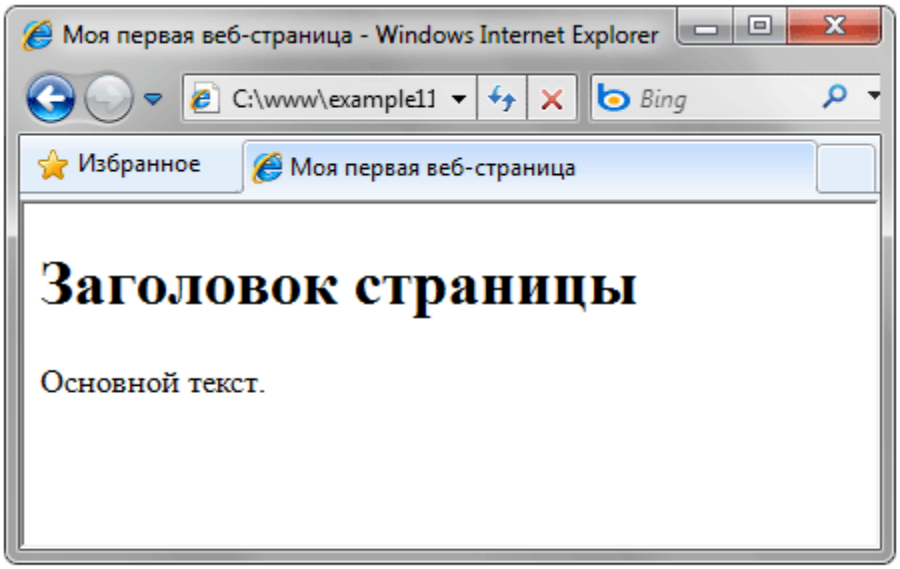


Рис. 1.3. Вид веб-страницы в окне браузера

В случае возникновения каких-либо ошибок проверьте правильность набора кода согласно примеру 1.1, расширение файла (должно быть html) и путь к документу.

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Является ли язык HTML полноценным языком программирования?
2. Сколько существует тегов и атрибутов в языке HTML?
3. Назовите плюсы HTML?
4. Назовите минусы HTML?

Практическое занятие № 6. Применение форматирования текстовой информации.

Цель занятия :

1. Продолжить знакомиться с языком разметки страниц HTML;
2. Познакомить с новыми тегами, используемы в HTML;
3. Научиться форматировать текст в HTML коде;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Исходные данные:

Папка на РС «Практическое занятие №6 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Хранения содержания веб-страницы (контента)

Элемент **<body>** предназначен для хранения содержания веб-страницы (контента), отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера **<body>**. К такой информации относится текст, изображения, теги, скрипты JavaScript и т.д.

Тег **<body>** также применяется для определения цветов ссылок и текста на веб-странице. Подобная практика в HTML 4 осуждается и взамен для указания

цветовой схемы рекомендуется использовать стили, применяя их к селектору **BODY**. Тем не менее, большинство атрибутов до сих пор поддерживается разными браузерами.

Часто тег **<body>** используется для размещения обработчика событий, например, onload, которое выполняется после того, как документ завершил загрузку в текущее окно или фрейм.

Открывающий и закрывающий теги **<body>** на веб-странице не являются обязательными, однако хорошим стилем считается их использование, чтобы определить начало и конец HTML-документа.

Синтаксис

```
<body>
  ...
</body>
```

Атрибуты

alink Устанавливает цвет активной ссылки.
background Задает фоновый рисунок на веб-странице.
bgcolor Цвет фона веб-страницы.
bgproperties Определяет, прокручивать фон совместно с текстом или нет.
bottommargin Отступ от нижнего края окна браузера до контента.
leftmargin Отступ по горизонтали от левого края окна браузера до контента.
link Цвет ссылок на веб-странице.
rightmargin Отступ от правого края окна браузера до контента.
scroll Устанавливает, отображать полосы прокрутки или нет.
text Цвет текста в документе.
topmargin Отступ от верхнего края окна браузера до контента.
vlink Цвет посещенных ссылок.

Закрывающий тег- открывающий и закрывающий теги не обязательны.

Отработайте первый пример, загрузите данный код, создайте web-страницу с данным текстом.

Пример

```
<!DOCTYPE HTML>
<html>
```

```
<head>
 <title>Тег BODY</title>
 <meta charset="utf-8">
</head>
<body onload="alert('Документ загружен')">
```

 Для эффективной работы не обойтись без необходимых и привычных инструментов, в том числе и при написании кода HTML. Поэтому для начальной разработки веб-страниц или даже небольшого сайта — так называется набор страниц, связанных между собой ссылками и единым оформлением, нам понадобятся следующие программы:•Текстовый редактор.• Браузер для просмотра результатов.•Валидатор — программа для проверки синтаксиса HTML и выявления ошибок в коде.
•Графический редактор.•Справочник по тегам HTML.

```
 <p>При использовании события ONLOAD тега <body> выполняется скрипт, написанный на языке JavaScript, в данном случае он выводит сообщение, что документ загружен.

.</p>


 </body>
</html>
```

Результат текущего примера показан на рис. 1. При использовании события onload тега **<body>** выполняется скрипт, написанный на языке JavaScript, в данном случае он выводит сообщение, что документ загружен.

Пример варианта задания

Создание фонового рисунка на Web-странице

Для задания фонового рисунка на Web -странице используется в теге **<body> атрибут** background

Background-определяет изображение, которое будет использоваться в качестве фонового рисунка. В отличие от обычных изображений, для фона не устанавливаются ширина и высота, и он всегда отображается в натуральную величину с масштабом 100%. Если рисунок по размеру меньше окна браузера, то картинка повторяется по горизонтали вправо и вниз, выстраиваясь, как мозаика. В

качестве фона допускается использовать анимированные изображения в формате GIF, но они отвлекают внимание читателей.

Синтаксис

```
<body background="URL">

...

</body>
```

Значения

Любой допустимый адрес изображения — можно использовать относительный или абсолютный путь. Пример : file:///C:/Users/user/Desktop/фото.jpg

Отработайте второй пример, загрузите данный код, изменив адрес изображения.

Пример

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <title>Тег BODY, атрибут background</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 </head>
 <body background="images/snow.gif">
  ...
 </body>
</html>
```

Изменение цвет текста в документе.

Для изменения текста в Web-странице используется в теге <body> атрибут text

Text-устанавливает цвет текста, используемого на веб-странице по умолчанию. Цвета отдельных элементов можно легко изменять с помощью стилей.

Синтаксис

```
<body text="цвет">

...

</body>
```

Значения

Веб-цвет

Чтобы не запоминать совокупность цифр, вместо них можно использовать имена широко используемых цветов. В таблице приведены имена популярных названий цветов.

Табл. 3. Названия некоторых цветов

Имя цвета	Цвет	Описание	Шестнадцатеричное значение
aqua		Голубой	#00ffff
black		Черный	#000000
blue		Синий	#0000ff
fuchsia		Фуксия	#ff00ff
gray		Серый	#808080
green		Зеленый	#008000
lime		Светло-зеленый	#00ff00
maroon		Темно-красный	#800000
navy		Темно-синий	#000080
olive		Оливковый	#808000
purple		Фиолетовый	#800080
red		Красный	#ff0000
silver		Светло-серый	#c0c0c0
teal		Сине-зеленый	#008080
white		Белый	#ffffff
yellow		Желтый	#ffff00

Значение по умолчанию

#000000

Отработайте третий пример, загрузите данный код, изменив цвет текста.

Пример

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <title>Тег BODY, атрибут text</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 </head>
 <body text="maroon">
  <p>Пример текста</p>
 </body>
</html>
```

Создайте Web-страницу, название которой «Практическое занятие №6», содержимое контента «***Цвет фона веб-страницы задан как #fa8e47. Символ решетки # перед числом означает, что оно шестнадцатеричное. Первые две цифры (fa) определяют красную составляющую цвета, цифры с третьей по четвертую (8e) — зеленую, а последние две цифры (47) — синюю. В итоге получится такой цвет***. fa8e47», цвет текста определяете самостоятельно. Web-страница должна иметь фоновый рисунок.

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите тег в языке HTML , позволяющий создавать фон странице с рисунком?
2. Назовите атрибуты для тега **body** в языке HTML?
3. Какой атрибут помогает изменить цвет контента?
4. В HTML цвет задается одним из двух путей, назовите их?

Практическое занятие №7. Создание списков (нумерованные, маркированные, списки перечислений) в Web-странице.

Цель занятия :

1. Продолжить знакомиться с языком разметки страниц HTML;
2. Познакомить с новыми тегами, используемы в HTML;
3. Научиться форматировать текст со списком в HTML коде;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Исходные данные:

Папка на РС «Практическое занятие №7 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

**Определение отдельного элемента списка. Тег **

Тег <li> определяет отдельный элемент списка. Внешний тег <ul> или <ol> устанавливает тип списка — маркированный или нумерованный.

Синтаксис

```
<ul>
  <li>элемент маркированного списка</li>
</ul>

<ol>
  <li>элемент нумерованного списка</li>
</ol>
```

Атрибуты

type -устнавливает вид маркера нумерованного или маркированного списка.

value -число, с которого будет начинаться нумерованный список.

Закрывающий тег -не обязателен.

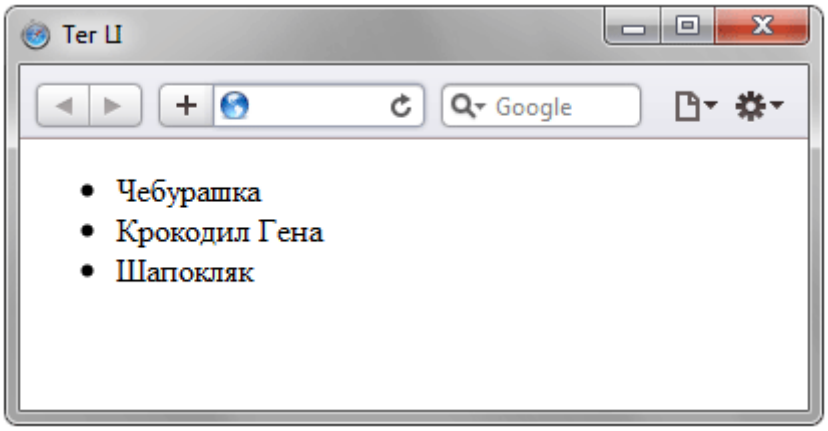
Отработайте первый пример, загрузите данный код

Пример

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Тег LI</title>
 </head>
 <body>

  <ul>
   <li>Чебурашка</li>
   <li>Крокодил Гена</li>
   <li>Шапокляк</li>
  </ul>

 </body>
```

```
</html>
```

Результат данного примера показан на рис. 1.



Создание разного маркированного списка. Тег

Тег <ul> устанавливает маркированный список. Каждый элемент списка должен начинаться с тега <li>. Если к тегу <ul> применяется таблица стилей, то элементы <li> наследуют эти свойства.

Синтаксис

```
<ul>
  <li>элемент маркированного списка</li>
</ul>
```

Закрывающий тег -обязателен.

Атрибуты

type -устанавливает вид маркера списка.

Синтаксис

```
<ul type="disc | circle | square">...</ul>
```

Значения

Для маркированного списка маркеры могут принимать один из трех видов: кружок (disc), окружность (circle) и квадрат (square). Значения атрибута type и получаемый вид показан в табл. 1.

Табл 1. Значения атрибута type

Код	Пример
<ul type="disc"> ...	• Чебурашка • Крокодил Гена • Шапокляк
<ul type="circle"> ...	○ Чебурашка ○ Крокодил Гена ○ Шапокляк
<ul type="square"> ...	▪ Чебурашка ▪ Крокодил Гена ▪ Шапокляк

Отработайте первый пример, с разными вариантами маркировки.

Определение нумерованного списка. Тег

Тег **<ol>** устанавливает нумерованный список. Каждый элемент списка должен начинаться с тега **<li>**. Если к тегу **<ol>** применяется таблица стилей, то элементы **<li>** наследуют эти свойства.

Синтаксис

```
<ol>

  <li>элемент нумерованного списка</li>

  <li>элемент нумерованного списка</li>

</ol>
```

Атрибуты

type -устанавливает вид маркера списка.
reversed- нумерация в списке становится по убыванию (3,2,1).
start- задаёт число, с которого будет начинаться нумерованный список.

Закрывающий тег-обязателен.

Отработайте второй пример с нумерованным списком.

Пример

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Тег OL</title>
 </head>
 <body>

  <ol>
```

```
    <li>Чебурашка</li>
    <li>Крокодил Гена</li>
    <li>Шапокляк</li>
  </ol>

 </body>
</html>
```

Пример варианта задания

С помощью тегов **<ol>** и **<li>** постройте списки, показанные ниже. Списки должны корректно отображаться в браузерах .

Первый список, начинается с 1

01. Баал
02. Агарес
03. Марбас
04. Пруфлас
05. Аамон

Второй список, начинается с 6

06. Барбатос
07. Буер
08. Гасион
09. Ботис

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите тег в языке HTML , позволяющий создавать маркированный список и каковы его атрибуты?

2. Назовите атрибуты для тега <li> в языке HTML?
3. Какой атрибут помогает изменить нумерацию в обратном порядке. ?
4. Как начать нумерацию с определенного значения, например с 10 ?

Практическое занятие№8. Вставка графических объектов

Цель занятия :

1. Продолжить знакомиться с языком разметки страниц HTML;
2. Познакомить с новыми тегами, используемы в HTML;
3. Научиться вставлять графические объекты в HTML коде;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Исходные данные:

Папка на РС «Практическое занятие №8 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Тег <img> предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG. Адрес файла с картинкой задаётся через атрибут src. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут border="0" в тег <img>.

Рисунки также могут применяться в качестве карт-изображений, когда картинка содержит активные области, выступающие в качестве ссылок. Такая карта по внешнему виду ничем не отличается от обычного изображения, но при этом оно может быть разбито на невидимые зоны разной формы, где каждая из областей служит ссылкой.

Синтаксис

HTML
```
<img src="URL" alt="альтернативный текст">
```

Атрибуты

align-Определяет как рисунок будет выравниваться по краю и способ обтекания текстом.
alt-Альтернативный текст для изображения.
border-Толщина рамки вокруг изображения.
height-Высота изображения.
hspace-Горизонтальный отступ от изображения до окружающего контента.
ismap-Говорит браузеру, что картинка является серверной картой-изображением.
longdesc-Указывает адрес документа, где содержится аннотация к картинке.

lowsrc-Адрес изображения низкого качества.
src-Путь к графическому файлу.
vspace-Вертикальный отступ от изображения до окружающего контента.
width-Ширина изображения.
usemap-Ссылка на тег **<map>**, содержащий координаты для клиентской карты-изображения.

Закрывающий тег-не требуется.

Отработайте первый пример, загрузите данный код, изменив текст, цвет текста. Картинку загрузите со своего компьютера. Поменяйте название страницы.

Пример

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Тег IMG</title>
 </head>
 <body>

  <p><a href="lorem.html"><img src="images/girl.png"
  width="189" height="255" alt="lorem"></a>
  Lorem ipsum dolor sit amet...</p>

 </body>
</html>
```

Изображение, помещаемое на веб-страницу, можно поместить в рамку различной ширины. Для этого служит атрибут border тега **<img>**. По умолчанию рамка вокруг изображения не отображается за исключением случая, когда рисунок является ссылкой. При этом цвет рамки совпадает с цветом ссылок, заданных с помощью стиля или атрибута link тега **<body>**.

Чтобы убрать рамку, следует задать атрибут border="0".

Синтаксис

```
<img border="толщина рамки">
```

Значения

Любое целое положительное число в пикселах.

Примерный вариант задания

Отработайте первый пример, загрузите данный код и добавьте рамку к рисунку. Измените ширину и высоту изображения. Добавьте альтернативный текст для изображения .

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите тег в языке HTML , позволяющий создавать графические объекты?
2. Назовите атрибуты для этого тега в языке HTML?
3. Какой атрибут помогает изменить рамку рисунка ?
4. Какой атрибут помогает добавлять текст к рисунку?

Практическое занятие №9. Создание гиперссылок. Якоря

Цель занятия :

1. Продолжить знакомиться с языком разметки страниц HTML;

2. Познакомить с новыми тегами, используемы в HTML;

3. Научиться вставлять гиперссылки и якори в HTML коде;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Исходные данные:

Папка на РС «Практическое занятие №9 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Для создания гиперссылок и якорей в веб-странице используются следующие теги:

Тег **<a>** является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия
атрибутов name или href тег **<a>** устанавливает ссылку или якорь. Якорем называется закладка внутри страницы, которую можно указать в качестве цели ссылки. При использовании ссылки, которая указывает на якорь, происходит переход к закладке внутри веб-страницы.

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. В качестве значения атрибута href используется адрес документа (URL, Universal Resource Locator, универсальный указатель ресурсов), на который происходит переход. Адрес ссылки может быть абсолютным и относительным. Абсолютные адреса работают везде и всюду независимо от имени сайта или веб-страницы, где прописана ссылка. Относительные ссылки, как следует из их названия, построены относительно текущего документа или корня сайта.

Синтаксис

```
<a href="URL">...</a>

<a name="идентификатор">...</a>
```

Атрибуты

accesskey Активация ссылки с помощью комбинации клавиш.

coords Устанавливает координаты активной области.
download Предлагает скачать указанный по ссылке файл.
href Задает адрес документа, на который следует перейти.
hreflang Идентифицирует язык текста по ссылке.
name Устанавливает имя якоря внутри документа.
rel Отношения между ссылаемым и текущим документами.
rev Отношения между текущим и ссылаемым документами.
shape Задает форму активной области ссылки для изображений.
tabindex Определяет последовательность перехода между ссылками при нажатии на кнопку Tab
target Имя окна или фрейма, куда браузер будет загружать документ.
title Добавляет всплывающую подсказку к тексту ссылки.
type Указывает MIME-тип документа, на который ведёт ссылка.

Закрывающий тег -обязателен.

Пример

```
<!DOCTYPE HTML>
<html>
 <head>
   <meta charset="utf-8">
  <title>Тег A</title>
 </head>
 <body>
  <p><a href="images/xxx.jpg">Посмотрите на мою фотографию!</a></p>
  <p><a href="tip.html">Как сделать такое же фото?</a></p>
</body>
</html>
```

Отработайте первый пример, загрузите данный код, используйте любую картинку , находящуюся в папке «Изображения» и любую web-страницу для перехода на них по гиперссылке. Текст для гиперссылки можете изменить, придумав свой. Название практического занятия возьмите за основу титула страницы.

Отработка атрибутов тега <a>

Атрибут download- При наличии атрибута download браузер не переходит по ссылке, а предложит скачать документ, указанный в адресе ссылки.

Синтаксис

```
<a download>Ссылка</a>
```

Обязательный атрибут-нет.

Пример

```
<!DOCTYPE html>
<html>
 <head>
```

```
  <meta charset="utf-8">
  <title>download</title>
 </head>
 <body>
  <p><a href="images/xxx.jpg">Открыть файл в браузере</a>
  <p><a href="images/xxx.jpg" download>Скачать файл</a>
</body>
</html>
```

Отработайте второй пример, загрузите данный код, используйте адрес картинки с вашего компьютера.

Атрибут name - используется для определения якоря внутри страницы. Вначале следует задать в соответствующем месте закладку и установить ее имя при помощи атрибута name тега **<a>**. Имя ссылки на закладку начинается символом #, после чего идет название закладки. Название выбирается любое, соответствующее тематике. Можно также делать ссылку на закладку, находящуюся в другой веб-странице и даже другом сайте. Для этого в адресе ссылки надлежит указать ее адрес и в конце добавить символ решетки # и имя закладки.

Между тегами **** и **** текст писать не обязательно, так как требуется лишь указать местоположение перехода по ссылке.

Синтаксис

```
<a name="закладка">...</a>
```

Обязательный атрибут

Обязателен для якорей.

Значения -любой текст с учетом регистра.

Значение по умолчанию- Нет.

Пример

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тег A, атрибут name</title>
 </head>
 <body>

  <p><a name="top"></a></p>
  <p style="height:3000px;">Здесь много-много текста.
   Прокручивай его вниз. </p>

  <p><a href="#top">Наверх</a></p>

 </body>
</html>
```

Примерный вариант задания

Отработайте третий пример, загрузите данный код. Посмотрите, появилась ли прокрутка и ссылка наверх. Добавьте во внутреннее содержание страницы рисунок и небольшой текст.

Содержание отчета:
Наименование практического занятия
Цель занятия
Вариант задания
Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
Список используемых источников
Выводы и предложения
Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите тег в языке HTML , позволяющий создавать гиперссылки на другие объекты?
2. Назовите атрибуты для этого тега в языке HTML?
3. Какой атрибут помогает устанавливать имя якоря внутри документа ?
4. Какой атрибут помогает скачивать объект?

Практическое занятие №10. Построение и редактирование таблиц

Цель занятия :

1. Продолжить знакомиться с языком разметки страниц HTML;
2. Познакомить с новыми тегами, используемы в HTML;
3. Научиться строить и редактировать таблицы в HTML коде;
4. Формировать ОК 01, ОК 02..

Исходные материалы и данные:

ПК, Браузер, Блокнот.

Исходные данные:

Папка на РС «Практическое занятие №10 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Для построения таблиц в HTML коде существуют тег <table>

Описание

Элемент **<table>** служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов **<tr>** и **<td>**. Внутри **<table>** допустимо использовать следующие элементы: **<caption>**, **<col>**, **<colgroup>**, **<tbody>**, **<td>**, **<tfoot>**, **<th>**, **<thead>** и **<tr>**.

Таблицы с невидимой границей долгое время использовались для верстки веб-страниц, позволяя разделять документ на модульные блоки. Подобный способ применения таблиц нашел воплощение на многих сайтах, пока ему на смену не пришел более современный способ верстки с помощью слоев.

Синтаксис

```
<table>
  <tr>
    <td>...</td>
  </tr>
</table>
```

Атрибуты

align -Определяет выравнивание таблицы.
background Задает фоновый рисунок в таблице.
bgcolor Цвет фона таблицы.
border Толщина рамки в пикселах.
bordercolor Цвет рамки.
cellpadding Отступ от рамки до содержимого ячейки.
cellspacing Расстояние между ячейками.
cols Число колонок в таблице.
frame Сообщает браузеру, как отображать границы вокруг таблицы.
height Высота таблицы.
rules Сообщает браузеру, где отображать границы между ячейками.
summary Краткое описание таблицы.

width Ширина таблицы.
Закрывающий тег -обязателен.

Пример

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Таблица размеров обуви</title>
 </head>
 <body>
  <table border="1">
   <caption>Таблица размеров обуви</caption>
   <tr>
    <th>Россия</th>
    <th>Великобритания</th>
    <th>Европа</th>
    <th>Длина ступни, см</th>
   </tr>
   <tr><td>34,5</td><td>3,5</td><td>36</td><td>23</td></tr>
   <tr><td>35,5</td><td>4</td><td>36⅔</td><td>23-23,5</td></tr>
   <tr><td>36</td><td>4,5</td><td>37⅓</td><td>23,5</td></tr>
   <tr><td>36,5</td><td>5</td><td>38</td><td>24</td></tr>
   <tr><td>37</td><td>5,5</td><td>38⅔</td><td>24,5</td></tr>
   <tr><td>38</td><td>6</td><td>39⅓</td><td>25</td></tr>
   <tr><td>38,5</td><td>6,5</td><td>40</td><td>25,5</td></tr>
   <tr><td>39</td><td>7</td><td>40⅔</td><td>25,5-26</td></tr>
   <tr><td>40</td><td>7,5</td><td>41⅓</td><td>26</td></tr>
   <tr><td>40,5</td><td>8</td><td>42</td><td>26,5</td></tr>
   <tr><td>41</td><td>8,5</td><td>42⅔</td><td>27</td></tr>
   <tr><td>42</td><td>9</td><td>43⅓</td><td>27,5</td></tr>
   <tr><td>43</td><td>9,5</td><td>44</td><td>28</td></tr>
   <tr><td>43,5</td><td>10</td><td>44⅔</td><td>28-28,5</td></tr>
   <tr><td>44</td><td>10,5</td><td>45⅓</td><td>28,5-29</td></tr>
   <tr><td>44,5</td><td>11</td><td>46</td><td>29</td></tr>
   <tr><td>45</td><td>11,5</td><td>46⅔</td><td>29,5</td></tr>
   <tr><td>46</td><td>12</td><td>47⅓</td><td>30</td></tr>
   <tr><td>46,5</td><td>12,5</td><td>48</td><td>30,5</td></tr>
   <tr><td>47</td><td>13</td><td>48⅔</td><td>31</td></tr>
   <tr><td>48</td><td>13,5</td><td>49⅓</td><td>31,5</td></tr>
  </table>
 </body>
</html>
```

Отработайте первый пример, загрузите данный код.

Пример варианта задания

Создайте свою таблицу по образцу

A	B	¬A инверсия	AVB дизъюнкция	A&B конъюнкция	A→B импликация	A↔B эквиваленция
0	0	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	1	0	0	0
1	1	0	1	1	1	1

Задайте фоновый рисунок или цвет фона таблицы, измените толщину и цвет рамки таблицы.

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите тег в языке HTML , позволяющий создавать таблицу в web-странице?
2. Назовите атрибуты для этого тега в языке HTML?
3. Какой атрибут помогает изменять цвет таблицы и цвет рамок таблицы ?
4. Какой атрибут помогает создать фоновый рисунок таблицы?

Практическое занятие №11. Создание интерактивной страницы

Цель занятия:

1. Научиться создавать группы Web-страниц методом преобразования документов MS Office.

Исходные материалы и данные:

ПК, Браузер Internet Explorer, MS Word, MS Excel.

Использованные источники:[6, стр. 467], [].

Исходные данные:

Папка на РС «Практическое занятие №11 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия ([https://htmlbook.ru/samhtml- электронный ресурс] в папке)

2. Обучающие должны выполнить задания по вариантам.

Пример варианта задания

Создание группы Web-страниц методом преобразования документов MS Office

1. Подготовить папку для размещения Ваших документов.

2. Создать документ Word следующего содержания.

Главный заголовок, например, Объект WordArt:

Далее 3-4 абзаца о своем происхождении (краткая автобиография).

Затем разместить текст:

"Далее Вы можете узнать подробности:"

И создать оглавление, состоящее, например, из двух пунктов:

Мои увлечения

Мои друзья

3. Оформить документ и сохранить в своей папке под именем *main.doc*

4. Создать документ Word, посвященный Вашим увлечениям. Сохранить документ под именем *hobby.doc* в своей папке. Документ должен быть оформлен, иметь нижний колонтитул и кроме текста содержать рисунки.

5. Создать книгу Excel с таблицей по приведенному ниже образцу, сохранить под именем *friends.xls*

Мои друзья			
Имя	**Возраст**	**Рост**	**Вес**
Вася	18	189	90
Зина	22	170	67
Коля	45	165	60
Лена	25	180	70
Среднее значение	**27,5**	**176**	**71,75**

Для вычисления средних значений должны быть использованы *формулы*.

5.1. Построить **графики**, иллюстрирующие сведения о Ваших друзьях. Расположить графики, под таблицей подогнать размеры таблицы и графиков.

5.2. Подготовить лист к печати: настроить параметры вкладки "Страница…" в режиме Предварительный просмотр, создать колонтитулы.

6. Установить связи между документами с помощью гиперссылок.

6.1. Открыть главный документ main.doc и последовательно выделяя заголовки разделов, закрепить за ними гиперссылки ("Меню – Вставить") на соответствующие документы.

6.2. Сохранить документ и проверить работоспособность гиперссылки. Возврат в Главный документ выполнять с помощью кнопки  на панели инструментов

7. В главном документе установить закладку на заголовок *Мои увлечения*. Дать ей название "Хобби". Сохранить документ.

8. Создать в конце каждого вспомогательного документа гиперссылки, обеспечивающие возврат в основной документ.

8.1. Подготовить рисунок для обеспечения возврата из вспомогательных документов в главный. Например, рисунок  можно получить с помощью создания графической копии активного окна в буфере (Alt+PrintScreen) и дальнейшего редактирования рисунка в редакторе Paint.

8.2. Вставить в конец каждого из документов рисунок и закрепить за ним гиперссылку на документ main.doc. В файле *hobby.doc* гиперссылка должна обеспечивать переход на закладку "Хобби".

9. Сохранить документы и проверить работу гиперссылок.

Создать группу связанных Web-страниц методом преобразования подготовленных документов.

10.1. Подготовить папку для Web-документов с именем My_Web.

10.2. Последовательно раскрывая подготовленные ранее документы, сохранить их в папке *My_Web*, указав:

Тип файла: *Web-страница (*.htm; *.html)*

10.3. Закрыть все документы, проанализировать изменения, произошедшие в структуре папок.

11. Просмотреть Web-документы, начиная с *main.htm*. Проанализировать, какие элементы документов изменились или вовсе исчезли. Попытаться сделать переход по гиперссылке. Убедиться в том, что связи между Web-страницами нуждаются в редактировании.

12. Отредактировать Web-документы, изменить гиперссылки, выполнить дополнительное оформление.

Внимание! Для перехода из Браузера в режим редактирования нужно воспользоваться меню "Файл" – "Править в Microsoft Word for Windows" или кнопкой  на панели инструментов.

13. Сохранить и закрыть все документы. Предъявить работу Web-страниц преподавателю.

Выводы и предложения проделанной работы:

Содержание отчета:

1. Наименование практического занятия

2. Цель занятия
3. Вариант занятия
4. Результат работы сохранить файлом в своей папке
5. Список используемых источников
6. Выводы и предложения
7. Даты и подписи курсанта и преподавателя

Вопросы для самопроверки :

1. Какими способами можно создать Web-страницы в MS Word ?
2. Как MS Word изменяет структуру папок на диске при сохранении новой Web-страницы
3. Какие новые приемы оформления документа появляются при работе с Web-страницами, а какие становятся недоступными?
4. Как вставляется гиперссылка на другой документ?

Раздел 4 Сетевые технологии
Тема 4.1 Компьютерные сети, локальные сети. Сеть Интернет

Практическое занятие №12 Объединение компьютеров в локальную сеть. Моделирование компьютерной сети с помощью программы Dia.

Цель занятия :

1. Изучить структуру компьютерной сети, технологией передачи и обработки данных.
2. Знать устройства, необходимые для организации компьютерной сети
3. Научиться моделировать компьютерные сети с помощью программы DIA

4. Формирование ОК5, ОК6, ЛР26, ЛР30

Исходные материалы и данные:

ПК, или DIA

Использованные источники:

Исходные данные:

Папка на РС «Практическое занятие №12 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия – ([Электронный учебник, Dia]в папке)

2. Обучающие должны выполнить задания по вариантам.

Теоретическая часть

Под *компьютерной сетью* понимают комплекс аппаратных и программных средств, предназначенных для обмена информацией и доступа пользователей к единым ресурсам сети.

Основное назначение компьютерных сетей - обеспечить совместный доступ пользователей к информации (базам данных, документам и т.д.) и ресурсам (жесткие диски, принтеры, накопители CD-ROM, модемы, выход в глобальную сеть и т.д.).

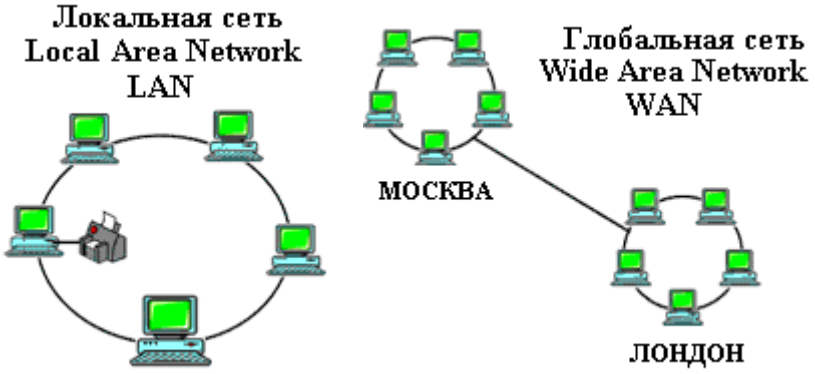
Абоненты сети – объекты, генерирующие или потребляющие информацию.

Абонентами сети могут быть отдельные ЭВМ, промышленные роботы, станки с ЧПУ (станки с числовым программным управлением) и т.д. Любой абонент сети подключён к станции.

Станция – аппаратура, которая выполняет функции, связанные с передачей и приёмом информации.

Для организации взаимодействия абонентов и станции необходима физическая передающая среда.

Физическая передающая среда – линии связи или пространство, в котором распространяются электрические сигналы, и аппаратура передачи данных.



Одной из основных характеристик линий или каналов связи является скорость передачи данных (пропускная способность).

На базе физической передающей среды строится

коммуникационная сеть. Таким образом, компьютерная сеть – это совокупность абонентских систем и коммуникационной сети.

По типу используемых ЭВМ выделяют *однородные и неоднородные сети.* В неоднородных сетях содержатся программно несовместимые компьютеры.

По территориальному признаку сети делят на *локальные* и *глобальные.*

Локальные сети (LAN, LocalAreaNetwork) объединяют абонентов, расположенных в пределах небольшой территории, обычно не более 2–2.5 км.

Локальные компьютерные сети позволят организовать работу отдельных предприятий и учреждений, в том числе и образовательных, решить задачу организации доступа к общим техническим и информационным ресурсам.

Глобальные сети (WAN, WideAreaNetwork) объединяют абонентов, расположенных друг от друга на значительных расстояниях: в разных районах города, в разных городах, странах, на разных континентах (например, сеть Интернет).

Взаимодействие между абонентами такой сети может осуществляться на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные компьютерные сети позволят решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам.

Основные компоненты коммуникационной сети:

передатчик;

приёмник;

сообщения (цифровые данные определённого формата: файл базы данных, таблица, ответ на запрос, текст или изображение);

средства передачи (физическая передающая среда и специальная аппаратура, обеспечивающая передачу информации).

Топология локальных сетей

Под топологией компьютерной сети обычно понимают физическое расположение компьютеров сети относительно друг друга и способ соединения их линиями.

Топология определяет требования к оборудованию, тип используемого кабеля, методы управления обменом, надежность работы, возможность расширения сети. Существует три основных вида топологии сети: шина, звезда и кольцо.

Шина(bus), при которой все компьютеры параллельно подключаются к одной

линии связи, и информация от каждого компьютера одновременно передается ко всем остальным компьютерам. Согласно этой топологии создается одноранговая сеть. При таком соединении компьютеры могут передавать информацию только по очереди, так как линия связи единственная.

Достоинства:

простота добавления новых узлов в сеть (это возможно даже во время работы сети);

сеть продолжает функционировать, даже если отдельные компьютеры вышли из строя;

недорогое сетевое оборудование за счет широкого распространения такой топологии.

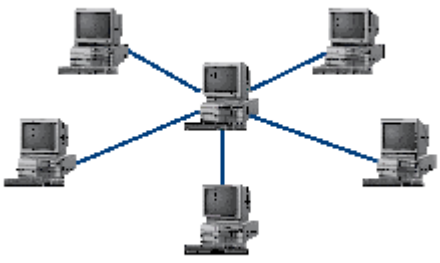
Недостатки:

сложность сетевого оборудования;

сложность диагностики неисправности сетевого оборудования из-за того, что все адаптеры включены параллельно;

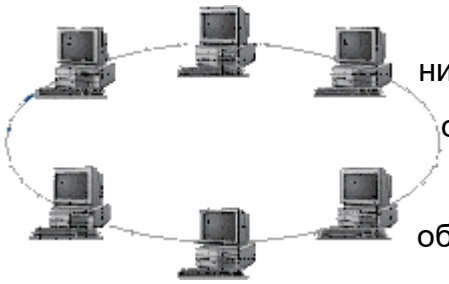
обрыв кабеля влечет за собой выход из строя всей сети;

ограничение на максимальную длину линий связи из-за того, что сигналы при передаче ослабляются и никак не восстанавливаются.



Звезда (star), при которой к одному центральному компьютеру присоединяются остальные периферийные компьютеры, причем каждый из них использует свою отдельную линию связи. Весь обмен информацией идет исключительно через центральный компьютер, на который ложится очень большая нагрузка, поэтому он предназначен только для обслуживания сети.

Достоинства:



выход из строя периферийного компьютера никак не отражается на функционировании оставшейся части сети;

простота используемого сетевого оборудования;

все точки подключения собраны в одном месте, что позволяет легко контролировать работу сети, локализовать неисправности сети путем отключения от центра тех или иных периферийных устройств;

не происходит затухания сигналов.

Недостатки:

выход из строя центрального компьютера делает сеть полностью неработоспособной;

жесткое ограничение количества периферийных компьютеров;

значительный расход кабеля.

Кольцо (ring), при котором каждый компьютер передает информацию всегда только одному компьютеру, следующему в цепочке, а получает информацию только от предыдущего в цепочке компьютера, и эта цепочка замкнута. Особенностью кольца является то, что каждый компьютер восстанавливает приходящий к нему сигнал, поэтому затухание сигнала во всем кольце не имеет никакого значения, важно только затухание между соседними компьютерами.

Достоинства:

легко подключить новые узлы, хотя для этого нужно приостановить работу сети;

большое количество узлов, которое можно подключить к сети (более 1000);

высокая устойчивость к перегрузкам.

Недостатки:

выход из строя хотя бы одного компьютера нарушает работу сети;

обрыв кабеля хотя бы в одном месте нарушает работу сети.

В отдельных случаях при конструировании сети используют комбинированную топологию. Например, дерево (tree)– комбинация нескольких звезд.

Каждый компьютер, который функционирует в локальной сети, должен иметь сетевой адаптер (сетевую карту). Функцией сетевого адаптера является

передача и прием сигналов, распространяемых по кабелям связи. Кроме того, компьютер должен быть оснащен сетевой операционной системой.

При конструировании сетей используют следующие виды

кабелей:

неэкранированная витая пара. Максимальное расстояние, на котором могут быть расположены компьютеры, соединенные этим кабелем, достигает 90 м. Скорость передачи информации - от 10 до 155 Мбит/с; экранированная витая пара. Скорость передачи информации - 16 Мбит/с на расстояние до 300 м.

коаксиальный кабель. Отличается более высокой механической прочностью, помехозащищённостью и позволяет передавать информацию на расстояние до 2000 м со скоростью 2-44 Мбит/с;

Волоконно-оптический кабель. Идеальная передающая среда, он не подвержен действию электромагнитных полей, позволяет передавать информацию на расстояние до 10 000 м со скоростью до 10 Гбит/с.

Понятие о глобальных сетях

Глобальная сеть – это объединения компьютеров, расположенных на удаленном расстоянии, для общего использования мировых информационных ресурсов. На сегодняшний день их насчитывается в мире более 200. Из них наиболее известной и сетей в глобальных сетях нет какого-либо единого центра управления. Основу сети составляют десятки и сотни тысяч компьютеров, соединенных теми или иными каналами связи. Каждый компьютер имеет уникальный идентификатор, что позволяет "проложить к нему маршрут" для доставки информации. Обычно в глобальной сети объединяются компьютеры, работающие по разным правилам (имеющие различную архитектуру, системное программное обеспечение и т.д.). Поэтому для передачи информации из одного вида сетей в другой используются шлюзы.

Шлюзы (gateway) – это устройства (компьютеры), служащие для объединения сетей с совершенно различными протоколами обмена.

Протокол обмена – это набор правил (соглашение, стандарт), определяющий принципы обмена данными между различными компьютерами в сети.

Протоколы условно делятся на базовые (более низкого уровня), отвечающие за передачу информации любого типа, и прикладные (более высокого уровня), отвечающие за функционирование специализированных служб.

Главный компьютер сети, который предоставляет доступ к общей базе данных, обеспечивает совместное использование устройств ввода-вывода и взаимодействия пользователей называется *сервером.*

Компьютер сети, который только использует сетевые ресурсы, но сам свои ресурсы в сеть не отдает, называется *клиентом* (часто его еще называют *рабочей станцией*).

Для работы в глобальной сети пользователю необходимо иметь соответствующее аппаратное и программное обеспечение.

Программное обеспечение можно разделить на два класса:

программы-серверы, которые размещаются на узле сети, обслуживающем компьютер пользователя;

программы-клиенты, размещенные на компьютере пользователя и пользующиеся услугами сервера.

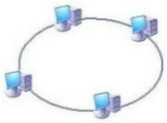
Глобальные сети предоставляют пользователям разнообразные услуги: электронная почта, удаленный доступ к любому компьютеру сети, поиск данных и программ и так далее.

Для повышения скорости передачи данных по телефонным линиям разработана техология ADSL (AsymmetricDigitalSubscriberLine - асимметричная цифровая абонентская линия).

Как правило, пользователь загружает из Интернета на свой компьютер большой объём информации, а в обратном направлении передаёт значительно меньший объём информации.

Пример варианта задания

Заполнить таблицу

			
топология			
Достоинства			

Недостатки			
Экономические затраты на кабель			
Возможность нелегального подключения			
Возможность подключения абонента без остановки работы сети			
Возможность обмена информацией без сервера			
Влияет ли поломка компьютера на работу сети			

Выполните в программе Dia построение представленных топологий компьютерных сетей.

Выводы и предложения проделанной работы

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант задания

Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

Список используемых источников

Выводы и предложения

Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

8. Что называют компьютерной сетью?
9. Что означает топология локальной сети?
10. Какие виды топологии ЛС вам известны?
11. Какие устройства необходимы для организации компьютерной сети?

Практическое занятие №13 Характеристика каналов связи. Определение скорости и времени передачи данных. IP адресация в сети Интернет.

*Цель занятия***:**

1. Знать характеристики каналов связи
2. Уметь определять скорость и время передачи данных
3. Иметь представление об адресации в сети Интернет
4. Формирование ОК5, ОК6, ЛР26, ЛР30

Оборудование: ПК, теоретический материал

Исходные данные:

Папка на РС «Практическое занятие №13 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны выполнить задания по вариантам.

Теоретический материал

Для организации взаимодействия абонентов и станции необходима физическая передающая среда.

Физическая передающая среда – линии связи или пространство, в котором распространяются электрические сигналы, и аппаратура передачи данных.

Одной из основных характеристик линий или каналов связи является скорость передачи данных (пропускная способность).

Скорость передачи данных – количество бит информации, передаваемой за единицу времени.

Обычно скорость передачи данных измеряется в битах в секунду (бит/с) и кратных единицах Кбит/с и Мбит/с.

Соотношения между единицами измерения: 1 Кбит/с =1024 бит/с; 1 Мбит/с =1024 Кбит/с; 1 Гбит/с =1024 Мбит/с.

На базе физической передающей среды строится коммуникационная сеть. Таким образом, компьютерная сеть – это совокупность абонентских систем и коммуникационной сети.

Специальное оборудование, подключаемое к телефонной линии, обеспечивает достаточно высокую входящую и более низкую исходящую скорость передачи данных

Объём переданной информации I вычисляется по формуле:

$$I = q \cdot t$$

У каждого компьютера в сети Интернет есть свой уникальный адрес — Uniform Resource Locator (URL).

Цифровые адреса состоят из четырех целых десятичных чисел, разделённых точками, каждое из этих чисел находится в интервале 0…255.

Пример: 225.224.196.10.

Максимальное количество IP-адресов, которое может быть использовано в подсети определённого размера, называется **subnet mask (маской подсети).**

В терминологии сетей TCP/IP **маской подсети** или **маской сети** называется битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу самого узла в этой сети.

Например, **узел с IP-адресом 12.34.56.78 и маской подсети 255.255.255.0 находится в сети 12.34.56.0/24**

В следствии того, что в двоичном виде маска представляет из себя непрерывную последовательность нулей или единиц, то в десятичном представлении, каждый октет сетевой маски может принимать только ограниченное число значений, а именно:

0, 128, 192, 224, 240, 248, 252, 254, 255.

Чтобы получить адрес сети, зная IP-адрес и маску подсети, необходимо применить к ним операцию поразрядной конъюнкции (логическое И).

Например,

IP-адрес: (192.168.1.2)

11000000 10101000 00000001 00000010

Маска подсети: (255.255.255.0)

11111111 11111111 11111111 00000000

Проведя порязрдную конъюнкцию получим

Адрес сети: 11000000 10101000 00000001 00000000 192. 168. 1. 0

Адрес сети: 192.168.1.0

Пример, адрес сети **192.168.0.0/16** (255.255.0.0) означает, что под адрес сети занято **16 бит**. Если адрес перевести в двоичное исчисление, то первые16 бит это – 192.168. Это и есть адрес сети: 192.168.0.0

11111111.11111111.00000000.00000000.

Если мы видим обозначение **"/24"**, это значит что используется 24 бита, в виде единиц (1), слева направо. Например:

/14 = 255.255.0.0 = **11111111.11111**00.00000000.00000000

/20 = 255.255.240.0 = **11111111.11111111.1111**0000.00000000

Как посчитать, сколько же адресов может быть в сети.

Важное замечание: **адресов в любой сети всегда четное!** Более того, оно **всегда кратно степени двойки.**

То есть **число адресов – это число, равное два в степени: число бит, оставшееся от вычитания количества бит под адрес сети из полного числа бит.** Всего в адресе 32 бита, в нашем случае под адрес сети **192.168.0.0** выделено **16 бит**, под адреса остается тоже 16. Это значит, чтобы узнать количество адресов в данной сети надо два возвести в 16 степень. Это будет **65536 адресов**.

Количество нулей в правом октете (правых октетах) связано с количеством хостов (hosts) — компьютеров в одной подсети, а **количество единиц** — с количеством самих подсетей.

Например, маска подсети с восемью нулями в четвёртом октете:

11111111 . 11111111 . 11111111 . **00000000**,

означает, что в этой сети может быть всего **256** (2^8=256) хостов (компьютеров) с адресами от 0 до 255.

Пример варианта задания

Решите задачи на определение скорости , времени и объема файла при передачи информации по сети.

1) Максимальная скорость передачи данных в локальной сети 100 Мбит/с. Сколько страниц текста можно передать за 1 сек, если 1 страница текста содержит 50 строк и на каждой строке - 70 символов.

2) Скорость передачи данных через ADSL-соединение равна 128000 бит/с. Через данное соединение передают файл размером 625 кбайт. Определите время передачи файла в секундах

3) Через ADSL-соединение файл размером 2500 Кбайт передавался 40 сек. Сколько секунд потребуется для передачи файла размером 2750 Кбайт.

4)Модем передаёт данные со скоростью 56 Кбит/сек. Передача текстового файла заняла 4,5 мин. Определите, сколько страниц содержал переданный текст, если известно, что он был представлен в кодировке ASCII, а на одной странице - 3072 символа.

Примеры решения задач на определение адреса компьютера

1) *Для некоторой подсети используется маска* ***255.255.252.0.***

Сколько различных адресов компьютеров допускает эта маска?

Примечание. На практике два из возможных адресов не используются для адресации узлов сети: адрес сети, в котором все биты, отсекаемые маской, равны 0, и широковещательный адрес, в котором все эти биты равны 1.

Решение

Каждая часть IP-адреса (всего 4 части) занимает 8 бит

Поскольку младшая часть маски 255.255.252.**0** нулевая, 8 бит уже свободны

Третья часть маски **252** = 255 – 3 = **11111100**$_2$ содержит 2 нулевых бита

общее число нулевых битов **N = 10 (8 + 2)**, число свободных адресов $\mathbf{2^N = 2^{10} =1024}$

Поскольку из них 2 адреса не используются (адрес сети и широковещательный адрес) для узлов сети остается

1024 – 2 = 1022 адреса

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что такое скорость передачи данных и в чем измеряется ?
2. От чего зависит скорость передачи информации?
3. Как рассчитать скорость передачи информации ?
4. Влияет ли топология сети на скорость передачи информации?
5. Как формируется адрес компьютера в сети?
6. Что такое маска подсети?

Тема 4.2 Сетевое хранение данных цифрового контента

Практическое занятие №14 Разграничение прав доступа в сети. Общее дисковое пространство в локальной сети. Облачные хранилища данных.

Цель занятия:

1.Изучить методы разграничения прав доступа в сети, понятие общего дискового пространства и облачного хранилища данных.

2. Формирование ОК5, ОК6, ЛР26, ЛР30

Исходные данные: ПК, Microsoft Word

Содержание и порядок выполнения задания:

Исходные данные:

Папка на РС «Практическое занятие №14 2 семестр» с теоретическим материалом по теме и заданиями для самостоятельного выполнения по вариантам.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны выполнить задания по вариантам.

Теоретическая часть

Разграничение прав доступа в сети

Основные понятия

При рассмотрении вопросов информационной безопасности используются понятия субъекта и объекта доступа. Субъект доступа может производить некоторый набор операций над каждым объектом доступа. Эти операции могут быть доступны или запрещены определенному субъекту или группе субъектов. Доступ к объектам обычно определяется на уровне операционной системы ее архитектурой и текущей политикой безопасности.

Рассмотрим некоторые определения, касающиеся методов и средств разграничения доступа субъектов к объектам:

Метод доступа к объекту – операция, которая определена для данного объекта. Ограничить доступ к объекту возможно именно с помощью ограничения возможных методов доступа.

Владелец объекта – субъект, который создал объект несет ответственность за конфиденциальность информации, содержащейся в объекте, и за доступ к нему.

Право доступа к объекту – право на доступ к объекту по одному или нескольким методам доступа.

Разграничение доступа – набор правил, который определяет для каждого субъекта, объекта и метода наличие или отсутствие права на доступ с помощью указанного метода.

Модели разграничения доступа

Наиболее распространенные модели разграничения доступа:

- дискреционная (избирательная) модель разграничения доступа;
- полномочная (мандатная) модель разграничения доступа.

Дискреционная

- любой объект имеет владельца;
- владелец имеет право произвольно ограничивать доступ субъектов к данному объекту;
- для каждого набора субъект – объект – метод право на доступ определен однозначно;
- наличие хотя бы одного привилегированного пользователя (например, администратора), который имеет возможность обращаться к любому объекту с помощью любого метода доступа.

В дискреционной модели определение прав доступа хранится в матрице доступа: в строках перечислены субъекты, а в столбцах – объекты. В каждой ячейке матрицы хранятся права доступа данного субъекта к данному объекту. Матрица доступа современной операционной системы занимает десятки мегабайт.

Полномочная модель характеризуется следующими правилами:

- каждый объект обладает грифом секретности. Гриф секретности имеет числовое значение: чем оно больше, тем выше секретность объекта;
- у каждого субъекта доступа есть уровень допуска.

Допуск к объекту в этой модели субъект получает только в случае, когда у субъекта значение уровня допуска не меньше значения грифа секретности объекта.

Преимущество полномочной модели состоит в отсутствии необходимости хранения больших объемов информации о разграничении доступа. Каждым субъектом выполняется хранение лишь значения своего уровня доступа, а каждым объектом – значения своего грифа секретности.

Виды методов разграничения доступа:

Разграничение доступа по спискам

Суть метода состоит в задании соответствий: для каждого пользователя задается список ресурсов и права доступа к ним или для каждого ресурса определяется список пользователей и права доступа к этим ресурсам. С помощью списков возможно установление прав с точностью до каждого пользователя. Возможен вариант добавления прав или явного запрета доступа. Метод доступа по спискам используется в подсистемах безопасности операционных систем и систем управления базами данных.

Использование матрицы установления полномочий

При использовании матрицы установления полномочий применяется матрица доступа (таблица полномочий). В матрице доступа в строках записываются идентификаторы субъектов, которые имеют доступ в компьютерную систему, а в столбцах – объекты (ресурсы) компьютерной системы.

В каждой ячейке матрицы может содержаться имя и размер ресурса, право доступа (чтение, запись и др.), ссылка на другую информационную структуру, которая уточняет права доступа, ссылка на программу, которая управляет правами доступа и др.

Данный метод является достаточно удобным, так как вся информация о полномочиях сохраняется в единой таблице. Недостаток матрицы – ее возможная громоздкость.

Разграничение доступа по уровням секретности и категориям

Разграничение по степени секретности разделяется на несколько уровней. Полномочия каждого пользователя могут быть заданы в соответствии с максимальным уровнем секретности, к которому он допущен.

Парольное разграничение доступа

Парольное разграничение использует методы доступа субъектов к объектам с помощью пароля. Постоянное использование паролей приводит к неудобствам для пользователей и временным задержкам. По этой причине методы парольного разграничения используются в исключительных ситуациях.

На практике принято сочетать разные методы разграничений доступа. Например, первые три метода усиливаются парольной защитой. Использование разграничения прав доступа является обязательным условием защищенной компьютерной системы.

Разграничение прав доступа пользователей

Разграничение прав доступа пользователей сети - это настройки, связанные с сегментированием на отдельные части и определение правил взаимодействия этих частей друг с другом. Если говорить техническим языком, это процесс создания VLAN для каждого конкретного подразделения, и настройки доступности этих VLAN между собой.

VLAN (Virtual Local Area Network) - это виртуальное разделение сети на части (локальные сети). По умолчанию, коммутатор, считает все свои интерфейсы (порты) в одной и той же локальной сети. С помощью дополнительной конфигурации, можно создавать отдельные подсети и выделять определенные порты коммутатора для работы в этих сетях. Лучшим определением VLAN можно считать то, что VLAN - это один широковещательный домен.

Разграничение прав доступа пользователей требуется, когда в Вашей организации есть ресурсы, которые предназначены для конкретных специалистов (бухгалтерские отчеты, например). Тем самым, можно создать отдельный VLAN для специалистов из бухгалтерии, запретив доступ к отчетности из других подразделений.

Ограничение доступа в социальные сети

Если вы не хотите, чтобы Ваши родственники (сотрудники) имели доступ к конкретным ресурсам (социальным сетям, запрещенным сайтам), можно предложить 4 доступных способа это сделать:

— Запретить доступ локально на конкретном ПК. Сделать это можно через файл /etc/hosts.

— Настройка ACL (Access Control List) на граничном маршрутизаторе. Смысл заключается в запрете доступа из конкретной подсети, к конкретным адресам.

— Настройка DNS (Domain Name System) сервера. Суть метода, в запрете разрешения конкретных доменных имен. Это означает, что при вводе в адресную строку браузера сайта vk.com, например, данное доменное имя не будет преобразовано в IPv4 адрес, и пользователь не сможет зайти на этот сайт.

— Специальное ПО.

Общее дисковое пространство в локальной сети

Каждый пользователь получает в распоряжение не только свой собственный диск, но и дисковое пространство любой машины, включенной в сеть (естественно, со всем его содержимым). Чтобы любой пользователь сети мог располагать вашим дисковым пространством, надо установить на компьютере соответствующий режим разрешения подобных операций с помощью драйвера сети. В связи с этим весьма актуальным является вопрос защиты данных вашего дискового пространства от их возможной порчи (возможно, непреднамеренной) со стороны других пользователей сети. При желании можно установить режим «только для чтения», не позволяющий другим пользователям ничего удалять с вашего диска, или «открыть» для просмотра не весь диск, а лишь одну или несколько директорий.

В каждой локальной сети всегда есть возможность обмена между пользователями текстовыми сообщениями и файлами, что для любой организации является немаловажным преимуществом, позволяющим избавиться от утомительной беготни сотрудников по различным отделам данного учреждения и использования служебного телефона для звонков в соседнюю комнату.

Физически обмен данными в сети осуществляется так: каждая из машин, включенных в сеть, имеет свой собственный номер — идентификатор;

информация от конкретного компьютера поступает в сеть в виде отдельных порций, их называют пакетами. Пакеты снабжаются информацией о том, какой машине в сети они предназначены. Далее пакет свободно перемещается по сети, сравнивая свой номер с идентификатором каждой конкретной машины. В случае их совпадения сообщение передается данной машине. Следует заметить, что рассылка данных и сообщений по сети возможна одновременно для всех пользователей этой сети: можно, например, послать сообщение не одному конкретному пользователю, а группе пользователей или всем пользователям сети одновременно, в том числе и себе самому.

Облачные хранилища данных

Облачное хранилище – это модель облачных вычислений, которая дает возможность хранить данные и файлы в Интернете, пользуясь услугами поставщика облачных вычислений, к которому вы подключаетесь либо через общедоступный Интернет, либо через частное сетевое соединение. Поставщик обеспечивает безопасное хранение и обслуживание серверов хранилища, инфраструктуры и сети, а также управление ими. Благодаря этому вы получаете доступ к данным тогда, когда он вам нужен, практически в неограниченном масштабе и с эластичным объемом. Если вы пользуетесь облачным хранилищем, то отпадает необходимость покупать и обслуживать собственную инфраструктуру хранилища данных, что дает гибкость, масштабируемость и надежность доступа к данным в любое время и в любом месте.

Облачное хранилище предоставляется поставщиком облачных сервисов, который владеет оборудованием для хранения данных и обеспечивает его работу за счет крупных центров обработки данных, расположенных в различных точках мира. Поставщики облачных хранилищ отвечают за состояние ресурсов, безопасность и надежность, обеспечивая доступность данных для приложений по Интернету по модели с оплатой по мере использования. Обычно вы подключаетесь к облачному хранилищу либо через Интернет, либо через выделенное частное подключение с использованием интернет-портала, веб-сайта или мобильного приложения. Когда клиенты приобретают облачное хранилище у поставщика сервисов, они делегируют ему большинство аспектов хранения данных, в том числе пространство, безопасность, доступность данных, серверы

хранилищ, вычислительные ресурсы и доставку данных по сети. Ваши приложения получают доступ к облачному хранилищу через традиционные протоколы хранения данных или напрямую через интерфейс программирования приложений (API). Поставщики облачных хранилищ предлагают дополнительные сервисы, предназначенные для защиты, сбора и анализа данных в огромных масштабах, а также управления ими.

Существует три типа облачных хранилищ данных: объектные хранилища, файловые хранилища и блочные хранилища. Каждый из них предлагает свои преимущества, подходящие для определенных примеров использования.

Объектное хранилище

Организациям требуется хранить крупные и растущие объемы неструктурированных данных, таких как фотографии, видео, данные машинного обучения (ML), показания датчиков, аудиофайлы и интернет-контент других типов. Поиск масштабируемого, эффективного и доступного хранилища для них может оказаться проблематичным. Объектное хранилище – это архитектура хранения данных для крупных хранилищ неструктурированных данных. Объекты хранят данные в том формате, в котором они поступают, и дают возможность настраивать метаданные таким образом, чтобы упростить доступ к данным и их анализ. Объекты не упорядочены в виде иерархии файлов и папок. Они хранятся в безопасных корзинах, которые обеспечивают практически неограниченную масштабируемость. Также они обходятся дешевле при хранении крупных объемов данных.

Для приложений, разработанных в облаке, как правило, требуются такие преимущества объектного хранилища, как широкие возможности масштабирования и характеристики метаданных. Решения объектных хранилищ идеально подходят для разработки с нуля современных приложений, для которых требуется гибкость и возможность масштабирования. Кроме того, эти хранилища можно использовать для импорта данных из существующих хранилищ с целью аналитики, резервного копирования или архивации.

Файловое хранилище

Хранилище на основе файлов или файловое хранилище широко используется различными приложениями и хранит данные в виде иерархии папок и файлов. Этот тип хранилища часто называется сервером сетевого хранилища данных (NAS) с широко используемыми протоколами файлового уровня блока

сообщений сервера (SMB), который используется на инстансах Windows, и сетевой файловой системы (NFS), используемой в Linux.

Блочное хранилище

Корпоративные приложения, например базы данных или системы планирования ресурсов предприятия (ERP), часто нуждаются в выделенном хранилище с низкими задержками для каждого из узлов. Такое хранилище работает аналогично хранилищу с прямым подключением (DAS) или сети хранения данных (SAN). В этом случае вы можете использовать сервис облачного хранилища, который хранит данные в виде блоков. Каждый блок имеет собственный идентификатор для быстрого сохранения и получения данных.

Вопросы обеспечения надежного хранения, безопасности и доступности критически важных корпоративных данных имеют первостепенную важность. При рассмотрении варианта хранения данных в облаке существует несколько фундаментальных требований:

Надежность и доступность

Облачное хранилище упрощает и улучшает традиционные процессы, применяемые в центрах обработки данных для обеспечения надежности и доступности данных. При использовании облачного хранилища данные хранятся с избыточностью на нескольких устройствах в одном или нескольких центрах обработки данных.

Безопасность

Облачное хранилище позволяет вам контролировать, где хранятся ваши данные, кто может получать к ним доступ и какие ресурсы использует ваша организация в любой момент времени. В идеале все данные должны шифроваться – как при хранении, так и при передаче. Разрешения и контроль доступа должны работать в облаке точно так же, как и в локальных хранилищах данных.

Облачное хранилище имеет несколько примеров использования в области управления приложениями, данными и обеспечения непрерывности бизнеса.

Рассмотрим несколько приведенных ниже примеров.

Аналитика и озера данных

Традиционные локальные решения для хранения данных могут оказаться непредсказуемыми в вопросах стоимости, производительности и масштабируемости, особенно с течением времени. Проекты, связанные с аналитикой, требуют наличия крупномасштабных, доступных и надежных пулов хранилищ данных с высокой доступностью. Часто подобные пулы называют «озерами данных».

Озера данных, построенные на базе объектных хранилищ, хранят информацию в исходной форме и содержат расширенные метаданные, позволяющие выборочно извлекать и использовать данные в целях анализа. Облачные озера данных могут быть расположены в центре множества систем хранения и обработки больших данных и аналитических движков, что позволит вам выполнить следующий проект быстрее и с большей степенью релевантности.

Резервное копирование и аварийное восстановление

Резервное копирование и аварийное восстановление крайне важны для защиты данных и обеспечения их доступности, но растущие требования к объему хранилища могут оказаться постоянной проблемой. Облачное хранилище обеспечивает низкую стоимость, высокую надежность и практически безграничные возможности масштабирования для решений резервного копирования и восстановления данных. Встроенные политики управления данными, могут выполнять автоматическую миграцию данных в более экономичные хранилища на основании частотных или временных параметров, за счет чего можно создавать архивные хранилища, позволяющие облегчить соблюдение юридических или нормативных требований. Эти преимущества предоставляют широкие возможности масштабирования в отраслях финансовых услуг, здравоохранения, наук о жизни, СМИ и развлечений, где постоянно создаются большие объемы неструктурированных данных с необходимостью длительного хранения.

Тестирование и разработка программного обеспечения

Среды тестирования и разработки программного обеспечения часто требуют создания, использования и последующего удаления отдельных, независимых и дублирующих сред хранения. Помимо временных затрат, с этими процессами могут быть связаны серьезные начальные капиталовложения.

Многие из крупнейших и наиболее прибыльных компаний мира создают приложения в рекордно быстрые сроки благодаря гибкости, производительности и низкой стоимости облачного хранилища. Даже работу простейших статичных веб-сайтов можно улучшить с низкими затратами. ИТ-специалисты и разработчики обращаются к решениям для хранения данных с оплатой по факту использования, которые избавляют их от проблем с управлением и масштабированием.

Перенос данных в облако

Доступность, устойчивость и низкие затраты на облачное хранилище могут быть очень весомыми аргументами. С другой стороны, ИТ-персонал, который работает с администраторами по хранению данных, резервному копированию, сетям, безопасности и обеспечению соответствия нормативным требованиям, может испытывать сомнения относительно фактической передачи больших объемов данных в облако. Для некоторых людей передача данных в облако может являться проблемой. Гибридные, периферийные сервисы и сервисы передачи данных помогают вам в физическом мире и упрощают передачу данных в облако.

Соответствие требованиям

Хранение конфиденциальных данных в облаке может поставить вопрос о регулировании и соответствии требованиям, особенно если данные находятся в системах хранилищ, ограниченных определенными требованиям. Средства контроля соответствия облачных данных призваны помочь в развертывании данных и применении к ним комплексных средств обеспечения соответствия нормативным требованиям, благодаря которым вы будете отвечать требованиям, выдвигаемым практически любым регулятивным органом в мире. Поставщики облачных сервисов, часто с использованием модели общей ответственности, дают клиентам возможность эффективно и экономично управлять рисками в ИТ-среде и гарантируют эффективное управление рисками путем обеспечения соответствия с помощью проверенных, широко признанных платформ и программ.

Хранилище приложений с оптимизацией для облака

Приложения с оптимизацией для облака используют контейнерные и без серверные технологии, чтобы отвечать ожиданиям клиентов и обеспечивать это соответствие быстро и гибко. Обычно эти приложения состоят из небольших и независимых компонентов со слабой взаимозависимостью, называемых

микросервисами, которые взаимодействуют на внутреннем уровне, делясь друг с другом данными или состоянием. Сервисы облачного хранилища обеспечивают управление данными для таких приложений и предоставляют решения текущих проблем хранения данных в облачной среде.

Архивирование

В наши дни предприятия сталкиваются с серьезными проблемами, вызванными экспоненциальным ростом объемов данных. Количество применений данных возрастает благодаря машинному обучению (ML) и аналитике. Чтобы обеспечить выполнение нормативных требований, информацию необходимо хранить долго. Клиентам необходимо заменить локальную инфраструктуру с архивами на ленточных накопителях и дисках такими решениями, которые обеспечивают улучшение надежности хранения данных, их мгновенное извлечение, повышение уровня безопасности, соблюдение нормативных требований, а также доступность данных для использования расширенной и бизнес-аналитики.

Гибридное облачное хранилище

Многие организации знают о преимуществах облачного хранилища, но используют приложения, развернутые локально, которым необходим доступ к данным с минимальной задержкой или быстрая передача данных в облако. Гибридные облачные архитектуры хранения данных позволяют подключить локальные приложения и системы к облачному хранилищу. Благодаря этому вы можете снизить расходы, оптимизировать процесс управления и использовать инновационные инструменты для работы с данными.

Хранилище базы данных

Поскольку блочное хранилище отличается высокой производительностью и готовностью к обновлению данных, многие организации используют его для транзакционных баз данных. Благодаря ограниченным метаданным блочное хранилище способно обеспечивать очень низкую задержку, которая требуется для высокопроизводительных рабочих нагрузок и чувствительных к задержке приложений, таких как базы данных.

Блочное хранилище позволяет разработчикам создать надежную, масштабируемую и высокоэффективную транзакционную базу данных. Поскольку каждый блок является автономным блоком, база данных работает оптимально даже при увеличении объема хранимых данных.

Машинное обучение и IoT

Пользуясь облачным хранилищем, вы можете обрабатывать, хранить и анализировать данные недалеко от местонахождения приложений, а затем копировать данные в облако для дальнейшего анализа. Облачное хранилище дает возможность эффективно и экономично хранить данные, а также использовать машинное обучение, искусственный интеллект (ИИ) и расширенную аналитику, чтобы получать ценную аналитическую информацию и внедрять инновации в своей работе.

Интернет вещей (Internet of Things, IoT) — это множество физических объектов, подключенных к интернету и обменивающихся данными. Концепция IoT может существенно улучшить многие сферы нашей жизни и помочь нам в создании более удобного, умного и безопасного мира. Примеры Интернета вещей варьируются от носимых вещей, таких как умные часы, до умного дома, который умеет, например, контролировать и автоматически менять степень освещения и отопления.

Выводы и предложения проделанной работы:

Содержание отчета:

Наименование практического занятия

Цель занятия

Вариант занятия

Список используемых источников

Выводы и предложения

Даты и подписи курсанта и преподавателя

Вопросы для самопроверки:

1. Назовите виды методов разграничения доступа к сети.
2. Для чего требуется разграничение прав доступа пользователей?
3. Как физически осуществляется обмен данными в сети.
4. Что такое облачное хранилище?
5. Как работает облачное хранилище?
6. Какие существуют типы облачного хранилища?
7. Какие требования к облачному хранилищу следует принять во внимание?
8. Назовите примеры использования облачных хранилищ.

Практическое занятие №15 Цифровой след. Службы и сервисы Интернета(почта, форумы, видеокон-ференции, социальные сети) Организация личного информационного пространства

*Цель занятия***:**

1.Познакомиться с понятием «Цифровой след», угрозами безопасности и рекомендации по защите личных данных и управлению репутацией в сети Интернет.

2.Рассмотреть достоинства и недостатки служб и сервисов Интернета.

Получить рекомендации по формированию личного информационного пространства.

3. Формировать ОК1, ОК2,ОК7,ЛР 4,10,26,30

Оборудование: ПК

Исходные данные:

Папка на РС «Практическое занятие №15 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы.

Теоретическая часть

Цифровой след. Определение и описание

Цифровой след, иногда называемый цифровой тенью или электронным следом – это данные, которые вы оставляете при использовании интернета. Эти данные включают посещаемые веб-сайты, отправляемые электронные письма и информацию, указываемую в онлайн-формах. Цифровой след можно использовать для отслеживания действий человека и его устройств в интернете. Пользователи интернета активно или пассивно создают собственный цифровой след.

Что такое цифровой след?

Каждый раз при использовании интернета вы оставляете за собой информационный след, называемый цифровым следом. Расширению цифрового

следа способствуют публикации в социальных сетях, подписки на информационные рассылки, оставленные отзывы и покупки в интернете.

Процесс расширения цифрового следа не всегда очевиден, например, веб-сайты могут отслеживать активность, устанавливая файлы cookie на ваше устройство, а приложения могут считывать данные без вашего ведома. Как только вы предоставляете организации доступ к вашей информации, она сможет продавать или передавать ее третьим лицам. В худшем случае ваши личные данные могут быть скомпрометированы в результате утечки.

Применительно к цифровым следам часто используются термины «активный» и «пассивный».

Активный цифровой след

Пользователь оставляет активный цифровой след, когда намеренно делится информацией о себе: делает публикации в социальных сетях или оставляет сообщения на сайтах или онлайн-форумах. Если пользователь вошел на веб-сайт с использованием зарегистрированного имени или профиля, все опубликованные им сообщения будут составлять его активный цифровой след. Также активный цифровой след остается при заполнении онлайн-форм, например, подписке на информационные рассылки, или при согласии принимать файлы cookie в браузере.

Пассивный цифровой след

Пассивный цифровой след создается, когда информация о пользователе собирается без его ведома. Это происходит, например, когда на веб-сайте собирается информация о том, сколько раз пользователи посещали сайт, откуда эти пользователи и их IP-адреса. Это скрытый процесс, о котором пользователи могут не догадываться. Другим примером использования пассивного следа является анализ рекламодателями ваших лайков, репостов и комментариев в социальных сетях с целью последующего профилирования и отображения вам определенного контента.

Почему важны цифровые следы?

Цифровые следы важны по следующим причинам:

- Они относительно постоянны. Как только информация становятся общедоступной (полностью или частично), как, например, публикация в Facebook, автор практически не может контролировать, как она будет использоваться другими людьми.
- Цифровой след может отражать цифровую репутацию человека, которая теперь считается такой же важной, как и репутация за пределами сети.
- Прежде чем принимать решения о найме, работодатели могут проверять цифровые следы своих потенциальных сотрудников, особенно их социальные сети. Колледжи и университеты могут проверять цифровые следы своих будущих студентов перед зачислением на учебу.
- Публикуемые в интернете сообщения и фотографии могут быть неверно истолкованы или изменены, что может привести к непреднамеренному оскорблению.
- Контент, предназначенный для узкой группы, может распространиться на более широкий круг и испортить отношения и дружбу.
- Киберпреступники могут использовать ваш цифровой след в целях фишинга, для доступа к учетной записи или для создания ложных профилей на основе ваших данных.

Поэтому стоит задуматься о том, что ваш цифровой след говорит о вас. Многие пытаются управлять своим цифровым следом, с осторожностью выполняя действия в сети и в первую очередь, контролируя потенциально собираемые данные.

Примеры цифрового следа

Цифровой след пользователя интернета может включать сотни составляющих. Ниже описаны лишь некоторые действия, увеличивающие цифровой след.

Онлайн покупки

- Совершение покупок на сайтах электронной коммерции.

- Подписка на купоны или создание учетной записи.
- Загрузка и использование приложений для покупок.
- Подписка на рассылку новостей бренда.

Интернет-банкинг

- Использование мобильного банковского приложения.
- Покупка и продажа акций.
- Подписка на финансовые публикации и блоги.
- Открытие счета кредитной карты.

Социальные медиа

- Использование социальных сетей на компьютере и устройствах.
- Вход на другие веб-сайты с использованием учетной записи социальной сети.
- Общение с друзьями и знакомыми.
- Обмен информацией, данными и фотографиями со знакомыми.
- Регистрация на сайте знакомств или в приложении.

Чтение новостей

- Подписка на новостные издания в интернете.
- Просмотр статей в новостном приложении.
- Подписка на информационные рассылки.
- Репосты прочитанных статей и информации.

Здоровье и фитнесс

- Использование фитнес-трекеров.
- Использование приложений для получения медицинской помощи.
- Регистрация адреса электронной почты в тренажерном зале.
- Подписка на блоги о здоровье и фитнесе.

Защита цифрового следа

Поскольку работодатели, университеты и другие лица могут проверить ваши данные в интернете, рекомендуется с внимательностью относиться к

цифровому следу. Ниже приведены рекомендации по защите личных данных и управлению репутацией в сети.

Используйте поисковые системы для проверки своего цифрового следа. Введите свое имя в поисковую систему. Укажите имя и фамилию, используйте все варианты написания. Если вы меняли имя, выполните поиск как текущего, так и прежнего имени. Просмотр результатов поиска даст вам представление об общедоступной информации о вас. Если какой-либо из результатов поиска показывает вас не в лучшем свете, можно связаться с администраторами сайта и узнать, могут ли они удалить эту информацию. Настройка оповещений Google – один из способов отслеживать информацию по вашему имени. Уменьшите количество источников информации, в которых упоминается ваше имя.

Например, веб-сайты, посвященные недвижимости, и сайт whitepages.com могут содержать о вас больше информации, чем хотелось бы. Эти сайты могут содержать личную информацию: номер телефона, адрес и возраст. Если вас это не устраивает, можно связаться с администраторами веб-сайтов и запросить удаление информации.

Ограничьте объем предоставляемых данных

Каждый раз при предоставлении личной информации вы расширяете свой цифровой след, а также увеличиваете вероятность того, что компания, хранящая ваши данные, воспользуется ими не по назначению или подвергнется взлому, в результате чего ваши данные могут попасть злоумышленникам. Поэтому прежде чем заполнять форму, подумайте, стоит ли это делать. Есть ли другие способы получить информацию или услугу без предоставления личных данных?

Проверьте параметры конфиденциальности

Параметры конфиденциальности в социальных сетях позволяют контролировать, кто видит ваши публикации. Проверьте, настроены ли эти параметры на комфортном для вас уровне. Например, Facebook позволяет ограничивать видимость публикаций для друзей и создавать специальные списки тех, кто может видеть определенные публикации. Однако не забывайте, что параметры конфиденциальности защищают вас только в конкретной социальной сети.

Избегайте раскрытия излишней информации в социальных сетях

Социальные сети позволяют легко общаться с людьми, однако провоцируют на раскрытие излишней информации. Подумайте, стоит ли указывать свое местоположение, раскрывать планы поездок или другую личную информацию. Не указывайте номер телефона и адрес электронной почты в разделе «Информация» в социальных сетях. Также не рекомендуется ставить лайки вашему банку, компании, предоставляющей медицинские услуги, аптеке и прочим организациям, поскольку это может указать киберпреступникам на ваши важные учетные записи.

Избегайте незащищенных веб-сайтов

Убедитесь, что вы совершаете транзакции на защищенном веб-сайте. Его веб-адрес должен начинаться с https://, а не с http://; буква s означает «безопасный» и указывает на наличие у сайта сертификата безопасности. Слева от адресной строки также должен отображаться значок замка. Не разглашайте конфиденциальную информацию, особенно платежные данные, на незащищенных сайтах.

Не указывайте личные данные при использовании публичных сетей Wi-Fi

Публичная сеть Wi-Fi менее безопасна, чем ваша личная сеть: неизвестно, кто ее настраивал и кто может иметь к ней доступ. Избегайте предоставления личной информации при использовании публичных сетей Wi-Fi.

Удаляйте старые учетные записи

Один из способов уменьшить свой цифровой след – удалить старые учетные записи, например, неиспользуемые профили в социальных сетях и подписки на не интересующие вас информационные рассылки. Удаление неиспользуемых учетных записей снижает вероятность утечки данных.

Создавайте надежные пароли и используйте менеджер паролей

Надежный пароль помогает обеспечить безопасность в интернете. Надежный пароль является длинным – состоит не менее чем из 12 символов, а в

идеале больше, и содержит сочетание заглавных и строчных букв, символов и цифр. Чем сложнее ваш пароль, тем сложнее его взломать. Использование менеджера паролей позволяет создавать, хранить и управлять всеми паролями с помощью единой защищенной учетной записи. Пароли необходимо хранить в секрете, никому не сообщать и нигде не записывать. Рекомендуется не использовать один пароль для всех учетных записей, а также регулярно менять пароли.

Сохраняйте конфиденциальность медицинских документов

Соблюдайте правила защиты данных и регулярно проверяйте свои медицинские документы. Похитители личных данных нацелены на медицинскую и финансовую информацию. Если преступники используют вашу личную информацию для получения медицинских услуг от вашего имени, их медицинские документы могут объединиться с вашими.

Не выполняйте авторизацию через Facebook

Входить на сайты и в приложения через Facebook довольно удобно. Однако при каждом входе на сторонний веб-сайт с использованием учетных данных Facebook, вы разрешаете компании-владельцу сайта получать ваши данные Facebook, что подвергает вашу личную информацию потенциальному риску.

Поддерживайте актуальность программного обеспечения

Устаревшее программное обеспечение может содержать множество цифровых следов. Если не установить последние обновления, киберпреступники могут получить доступ к этой информации. Используя уязвимости в программном обеспечении, они могут с легкостью получить доступ к устройствам и данным. Регулярное обновление программного обеспечения позволяет предотвратить это, поскольку устаревшее программное обеспечение является более уязвимым для атак злоумышленников.

Настройте использование мобильного устройства

Установите пароль для мобильного устройства, чтобы в случае утери никто, кроме вас, не мог получить к нему доступ. При установке приложений ознакомьтесь с пользовательским соглашением. Для многих приложений в нем описано, какую информацию оно собирает и для чего она может использоваться. Приложения могут собирать личные данные, такие как электронная почта, местоположение и действия в интернете. Прежде чем использовать приложение, убедитесь, что вас устраивает, какую информацию оно собирает.

Оценивайте материалы перед публикацией

На основе ваших публикаций и комментариев в интернете, а также по отзывам других людей формируется мнение о вас. Некоторые аспекты вашего цифрового следа, например, загруженные фотографии, комментарии в блогах, видео на YouTube и публикации в Facebook, могут показать вас совсем не с той стороны, с которой вы бы хотели. Создавайте положительный цифровой след, публикуя только то, что создает вам желаемый образ.

В случае взлома примите немедленные меры

Если вы предполагаете, что ваши данные могли быть скомпрометированы в результате взлома, немедленно примите меры. Если речь идет о финансовых потерях, сообщите о нарушении в банк или компанию, выпустившую кредитную карту. Измените все пароли, которые могли быть раскрыты. Если скомпрометированный пароль использовался для других учетных записей, измените его везде.

Используйте VPN

Использование виртуальной частной сети (VPN) помогает защитить цифровой след. VPN маскирует IP-адрес, что практически не позволяет отследить ваши действия в сети. Это повышает конфиденциальность при работе в интернете и не позволяет веб-сайтам устанавливать файлы cookie, отслеживающие вашу историю просмотров. Например, Kaspersky Secure Connection позволяет установить безопасное соединение между вашим устройством и интернет-сервером, чтобы никто не смог отслеживать или получать доступ к передаваемым данным.

Электронная почта (англ.*email, e-mail*,от англ.*electronic mail*) — технология и предоставляемые ею услуги по пересылке и получению электронных сообщений (называемых «письма» или «электронные письма») по распределённой (в том числе глобальной) компьютерной сети.

Электронная почта по составу элементов и принципу работы практически повторяет систему обычной (бумажной) почты, заимствуя как термины (почта, письмо, конверт, вложение, ящик, доставка и другие), так и характерные особенности - простоту использования, задержки передачи сообщений, достаточную надёжность и в то же время отсутствие гарантии доставки.

Достоинствами электронной почты являются: легко воспринимаемые и запоминаемые человеком адреса вида имя_пользователя@имя_домена; возможность передачи, как простого текста, так и форматированного, а также произвольных файлов; независимость серверов (в общем случае они обращаются друг к другу непосредственно); достаточно высокая надёжность доставки сообщения; простота использования человеком и программами.

Недостатки электронной почты: наличие такого явления, как спам (массовые рекламные и вирусные рассылки); теоретическая невозможность гарантированной доставки конкретного письма; возможные задержки доставки сообщения (до нескольких суток); ограничения на размер одного сообщения и на общий размер сообщений в почтовом ящике (персональные для пользователей).

Основным отличием (и достоинством е-мейл) от прочих систем передачи сообщений (например, служб мгновенных сообщений) ранее являлась возможность *отложенной доставки* сообщения, а также развитая (и запутанная, из-за длительного времени развития) система взаимодействия между независимыми почтовыми серверами (отказ одного сервера не приводил к неработоспособности всей системы).

В настоящее время любой начинающий пользователь может завести свой бесплатный электронный почтовый ящик, достаточно зарегистрироваться на одном из интернет порталов.

Электронная почта доступна не только в латинских доменных зонах, но и в кириллической зоне .РФ

Чат (англ. *chat* — болтать) — средство обмена сообщениями по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение. Характерной

особенностью является коммуникация именно в реальном времени или близкая к этому, что отличает чат от форумов и других «медленных» средств.

Под словом чат обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ мгновенного обмена сообщениями, например, ICQ или даже SMS.

Видеоконференция (англ.*videoconference*) — область информационной технологии, обеспечивающая одновременно двустороннюю передачу, обработку, преобразование и представление интерактивной информации на расстояние в режиме реального времени с помощью аппаратно-программных средств вычислительной техники.

Взаимодействие в режиме видеоконференций также называют сеансом видеоконференцсвязи.

Видеоконференцсвязь (сокращенное название ВКС) — это телекоммуникационная технология интерактивного взаимодействия двух и более удаленных абонентов, при которой между ними возможен обмен аудио- и видеоинформацией в реальном масштабе времени с учетом передачи управляющих данных.

Цели внедрения видеоконференцсвязи

Видеоконференция применяется как средство оперативного принятия решения в той или иной ситуации; при чрезвычайных ситуациях; для сокращения командировочных расходов в территориально распределенных организациях; повышения эффективности; проведения судебных процессов с дистанционным участием осужденных, а также как один из элементов технологий телемедицины и дистанционного обучения.

Во многих государственных и коммерческих организациях видеоконференция приносит большие результаты и максимальную эффективность, а именно:

- снижает время на переезды и связанные с ними расходы;
- ускоряет процессы принятия решений в чрезвычайных ситуациях;
- сокращает время рассмотрения дел в судах общей юрисдикции;
- увеличивает производительность труда;
- решает кадровые вопросы и социально-экономические ситуации;
- предотвращает усталость и стресс;

— позволяет следить за состоянием рынка и быстро реагировать на его изменения; дает возможность принимать более обоснованные решения за счёт привлечения при необходимости дополнительных экспертов;

— быстро и эффективно распределяет ресурсы, и так далее.

Для общения в режиме видеоконференции абонент должен иметь терминальное устройство (кодек) видеоконференцсвязи, видеотелефон или иное средство вычислительной техники. Как правило, в комплекс устройств для видеоконференцсвязи входит:

— центральное устройство — кодек с видеокамерой и микрофоном, обеспечивающего кодирование/декодирование аудио- и видеоинформации, захват и отображение контента;

— устройство отображения информации и воспроизведения звука.

В качестве кодека может использоваться персональный компьютер с программным обеспечением для видеоконференций.

Большую роль в видеоконференции играют каналы связи, то есть транспортная сеть передачи данных. Для подключения к каналам связи используются сетевые протоколы IP или ISDN.

Существует два режима работы ВКС, которые позволяют проводить двусторонние (режим «точка точка») и многосторонние (режим «многоточка») видеоконференции.

Как правило, видеоконференцсвязь в режиме «точка-точка» удовлетворяет потребности только на начальном этапе внедрения технологии, и довольно скоро возникает необходимость одновременного взаимодействия между несколькими абонентами. Такой режим работы называется «многоточечный» или многоточечной видеоконференцсвязью. Для реализации данного режима требуется наличие активации многоточечной лицензии в кодеке при условии, если устройство поддерживает данную функцию, либо специального видеосервера MCU (англ. *Multipoint Control Unit*), или программно-аппаратной системы управления.

Интернет-телефония

VoIP (англ.*Voice over IP*; IP-телефония, произносится "*войп*") — система связи, обеспечивающая передачу речевого сигнала по сети Интернет или по любым другим IP-сетям. Сигнал по каналу связи передаётся в цифровом виде и,

как правило, перед передачей преобразовывается (сжимается) с тем, чтобы удалить избыточность.

Голосовая и видеосвязь посредством компьютерных сетей стала популярной во всём мире с начала XXI века и в настоящее время широко используется как частными пользователями, так и в корпоративном секторе. Применение систем IP-телефонии позволяет компаниям-операторам связи значительно снизить стоимость звонков (особенно международных) и интегрировать телефонию с сервисами Интернета, предоставлять интеллектуальные услуги.

Организация личного информационного пространства

Одна из основных компетенций современного общества — умение работать с информацией.

Информационные ресурсы являются важнейшей составляющей информационного пространства каждой личности, а также фактором, определяющим развитие общества в целом. Современная действительность предъявляет повышенные требования к своевременности, достоверности и полноте информации, без которой немыслима эффективная деятельность.

Информационные ресурсы, которые доступны пользователю при работе на компьютерных устройствах, называют личным информационным пространством пользователя.

Личное информационное пространство является обязательным атрибутом современного человека, а навыки по его формированию можно рассматривать как важные информационные компетенции.

В случае если компьютерное устройство не подключено к Интернету, личное информационное пространство ограничено лишь теми программами и данными, которые размещены на этом компьютере. Если же имеется возможность подключения к Интернету, информационное пространство пользователя становится практически безграничным.

Для эффективного применения компьютерных устройств важно правильно сформировать индивидуальное информационное пространство:

- установить программное обеспечение;
- обеспечить подключение к Интернету;
- настроить индивидуальный интерфейс;
- обеспечить доступ к облачным хранилищам и веб-сервисам;

- создать и сохранить необходимые документы;
- продумать структуру каталогов для хранения файлов с документами.

При организации индивидуального интерфейса важно настроить:

- скорость доступа к информационным ресурсам;
- пользовательский интерфейс периферийных устройств и программного обеспечения;
- скорость работы компьютерного устройства.

Наряду с информационными ресурсами компонентами информационного пространства являются:

- средства информационного взаимодействия — средства телекоммуникаций на уровне объединения компьютерных сетей и средств различного вида связи (телефонной, телевизионной, спутниковой);
- комплексы, использующие эти средства, могут объединяться в системы передачи/приема для информационного обеспечения общества;
- информационная инфраструктура — система организационных структур, подсистем, которые обеспечивают функционирование и развитие информационного пространства страны и средств информационного взаимодействия.

Основные функции информационного пространства:

1. Интегрирующая. Пространство информации объединяет в единую социокультурную и пространственно-коммуникативную среду разнообразные виды деятельности человека.

2. Коммуникативная. Информационное пространство образует среду интерактивной, мобильной коммуникации различных субъектов деятельности для информационного обмена.

3. Геополитическая. Пространство информации создает собственные ресурсы, существенно меняет значимость традиционных ресурсов, что способствует созданию новой среды конкуренции и геополитических отношений.

4. Актуализирующая. В информационном пространстве производится актуализация интересов различных субъектов деятельности с помощью реализации ими информационной политики.

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Список используемых источников
5. Выводы и предложения

Вопросы для обсуждения:

1. Что такое цифровой след?
2. Какие бывают варианты цифрового следа.
3. Почему важно помнить про цифровые следы?
4. Как защитить свой цифровой след?
5. Как удалить свой цифровой след?
6. Что включается в понятие «Личное информационное пространство»?
7. Какие проблемы могут возникнуть при организации личного информационного пространства?
8. Какие существуют средства для автоматизации обработки информации в информационных пространствах?
9. Перечислите требования к личному информационному пространству.
10. Что относится к сервисам сети Интернет?
11. Какие сервисы служат для онлайн общения?

Практическое занятие №16 Правовые основы работы в сети Интернет. Соблюдение мер безопасности при работе в сети Интернет

*Цель занятия***:**

1. Познакомить с законами об информации и информационной безопасности, угрозами кибербезопасности и рекомендациями по обеспечению безопасной работы в сети Интернет.

2. Формировать ОК1, ОК2,ОК7,ЛР 4,10,26,30

Исходные данные:

Папка на РС «Практическое занятие №16 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы.

Теоретическая часть

Интернет представляет собой однородную информационную сеть, без которой жизнь современного человека почти невозможна. Однако развитие этой сферы требует детального изучения, так как ни в одной из стран мира, не присутствует никакого обособленного законодательства, регулирующего правоотношения в Интернете. Практика защиты законности и правопорядка в этой области недостаточно развита и осложняется отсутствием федерального закона о российском сегменте, а также экстерриториальной природой Интернета и другими факторами.

В наиболее общей форме главные условия регулирования сферы Интернет сохранены в конституции Российской Федерации: основной закон устанавливает, что к сбору, хранению, использованию и распространению информации о частной жизни человека без его согласия не позволяется (Статья 24). Каждый гражданин РФ имеет право свободно искать, получать, передавать и распределить информацию любыми законными способами. Свобода массовой информации гарантируется. Цензура запрещена (Статья 29). Реализация указанных прав может сопровождаться ограничениями, установленными законом и необходимыми в обществе для уважения прав и репутации других людей, защиты государственной безопасности и общественного порядка, который также сохранен в конституции Российской Федерации и также в 10-й статье соглашения по защите прав и основных свобод человека.

Главные законы об информации и информационной безопасности:

1. **149-ФЗ** об информационной безопасности — устанавливает основные права и обязанности, касающиеся информации и информационной безопасности.

2. **152-ФЗ** — описывает правила работы с персональными данными.

3. 98-ФЗ — определяет, что относится к коммерческой тайне компаний.

4. **68-ФЗ** — дает определение электронной подписи и описывает, как и когда ее можно применять, какой юридической силой она обладает.

5. **187-ФЗ** — описывает правила защиты IT-инфраструктуры на предприятиях, работающих в сферах, критически важных для государства. К таким сферам относится здравоохранение, наука, оборона, связь, транспорт, энергетика, банки и некоторая промышленность.

149-ФЗ «Об информации, информационных технологиях и о защите информации»

149-ФЗ — главный закон об информации в России. Он определяет ключевые термины, например, говорит, что информация — это любые данные, сведения и сообщения, представляемые в любой форме. Также там описано, что такое сайт, электронное сообщение и поисковая система. Именно на этот закон и эти определения нужно ссылаться при составлении документов по информационной безопасности.

В 149-ФЗ сказано, какая информация считается конфиденциальной, а какая — общедоступной, когда и как можно ограничивать доступ к информации, как происходит обмен данными. Также именно здесь прописаны основные требования к защите информации и ответственность за нарушения при работе с ней.

Ключевые моменты закона об информационной безопасности:

1. Нельзя собирать и распространять информацию о жизни человека без его согласия.

2. Все информационные технологии равнозначны — нельзя обязать компанию использовать какие-то конкретные технологии для создания информационной системы.

3. Есть информация, к которой нельзя ограничивать доступ, например сведения о состоянии окружающей среды.

4. Некоторую информацию распространять запрещено, например ту, которая пропагандирует насилие или нетерпимость.

5. Тот, кто хранит информацию, обязан ее защищать, например, предотвращать доступ к ней третьих лиц.

6. У государства есть реестр запрещенных сайтов. Роскомнадзор может вносить туда сайты, на которых хранится информация, запрещенная к распространению на территории РФ.

7. Владелец заблокированного сайта может удалить незаконную информацию и сообщить об этом в Роскомнадзор — тогда его сайт разблокируют.

152-ФЗ «О персональных данных»

Этот закон регулирует работу с персональными данными — личными данными конкретных людей. Его обязаны соблюдать те, кто собирает и хранит эти данные. Например, компании, которые ведут базу клиентов или сотрудников.

Ключевые моменты закона:

1. Перед сбором и обработкой персональных данных нужно спрашивать согласие их владельца.

2. Для защиты информации закон обязывает собирать персональные данные только с конкретной целью.

3. Если вы собираете персональные данные, то обязаны держать их в секрете и защищать от посторонних.

4. Если владелец персональных данных потребует их удалить, вы обязаны сразу же это сделать.

5. Если вы работаете с персональными данными, то обязаны хранить и обрабатывать их в базах на территории Российской Федерации. При этом

данные можно передавать за границу при соблюдении определенных условий, прописанных в законе — жесткого запрета на трансграничную передачу данных нет.

Краткая инструкция по выполнению федерального закона о персональных данных 152-ФЗ

1. Обеспечить защиту данных: установить антивирус, предотвратить доступ к данным посторонних.
2. Разработать пакет документов: соглашение об обработке, политику конфиденциальности, модель угроз.
3. Назначить ответственного за обработку персональных данных и составить список тех, кто имеет доступ к данным.
4. Уведомить Роскомнадзор о том, что вы обрабатываете персональные данные.
5. Подготовить форму согласия на обработку персональных данных и получать согласие от каждого человека, чьи данные вы собираете.

98-ФЗ «О коммерческой тайне»

Этот закон определяет, что такое коммерческая тайна, как ее охранять и что будет, если передать ее посторонним. В нем сказано, что коммерческой тайной считается информация, которая помогает компании увеличить доходы, избежать расходов или получить любую коммерческую выгоду.

Ключевые моменты закона о защите информации компании:

1. Обладатель информации сам решает, является она коммерческой тайной или нет. Для этого он составляет документ — перечень информации, составляющей коммерческую тайну.

2. Некоторые сведения нельзя причислять к коммерческой тайне, например, информацию об учредителе фирмы или численности работников.

3. Государство может затребовать у компании коммерческую тайну по веской причине, например, если есть подозрение, что компания нарушает закон. Компания обязана предоставить эту информацию.

4. Компания обязана защищать свою коммерческую тайну и вести учет лиц, которым доступна эта информация.

5. Если кто-то разглашает коммерческую тайну, его можно уволить, назначить штраф или привлечь к уголовной ответственности.

63-ФЗ «Об электронной подписи»

Этот закон касается электронной подписи — цифрового аналога физической подписи, который помогает подтвердить подлинность информации и избежать ее искажения и подделки. Закон определяет, что такое электронная подпись, какую юридическую силу она имеет и в каких сферах ее можно использовать.

Ключевые моменты закона:

1. Для создания электронной подписи можно использовать любые программы и технические средства, которые обеспечивают надежность подписи. Вы не обязаны использовать для этого какое-то конкретное государственное ПО.

2. Подписи бывают простые, усиленные неквалифицированные и усиленные квалифицированные. У них разные технические особенности, разные сферы применения и разный юридический вес. Самые надежные — усиленные квалифицированные подписи, они полностью аналогичны физической подписи на документе.

3. Те, кто работает с квалифицированной подписью, обязаны держать в тайне ключ подписи.

4. Выдавать электронные подписи и сертификаты, подтверждающие их действительность, может только специальный удостоверяющий центр.

187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации»

Этот закон касается компаний, которые работают в сферах, критически важных для жизни государства — таких, что сбой в их работе отразится на здоровье, безопасности и комфорте граждан России.

К таким сферам относится здравоохранение, наука, транспорт, связь, энергетика, банки, топливная промышленность, атомная энергетика, оборонная промышленность, ракетно-космическая промышленность, горнодобывающая промышленность, металлургическая промышленность и химическая промышленность. Также сюда относят компании, которые обеспечивают работу предприятий из этих сфер, например, предоставляют оборудование в аренду или разрабатывают для них ПО.

Если на предприятии из этой сферы будет простой, это негативно отразится на жизни всего государства. Поэтому к IT-инфраструктуре и безопасности информационных систем на этих предприятиях предъявляют особые требования.

Ключевые моменты закона об информационной безопасности критически важных структур:

1. Для защиты критической инфраструктуры существует Государственная система обнаружения, предупреждения и ликвидации последствий компьютерных атак (ГосСОПКА).

2. Объекты критически важной инфраструктуры обязаны подключиться к ГосСОПКА. Для этого нужно купить и установить специальное ПО, которое будет следить за безопасностью инфраструктуры компании.

3. Одна из мер предупреждения — проверка и сертификация оборудования, ПО и всей инфраструктуры, которая используется на критически важных предприятиях.

4. Субъекты критической информационной инфраструктуры обязаны сообщать об инцидентах в своих информационных системах и выполнять требования государственных служащих. Например, использовать только сертифицированное ПО.

5. Все IT-системы критически важных предприятий должны быть защищены от неправомерного доступа и непрерывно взаимодействовать с ГосСОПКА.

6. При разработке IT-инфраструктуры критически важные предприятия должны руководствоваться 239 приказом ФСТЭК. В нем прописаны основные требования к защите информации на таких предприятиях.

7. Государство имеет право проверять объекты критически важной инфраструктуры, в том числе внепланово, например, после компьютерных инцидентов вроде взлома или потери информации.

Угрозы кибербезопасности - это различные действия, которые могут привести к нарушениям состояния защиты информации

Другими словами, это потенциально возможные события, процессы или действия, которые могут нанести ущерб информационной или киберсреде.

Кибербуллинг — преследование сообщениями, содержащими оскорбления, агрессию, запугивание; хулиганство; социальное бойкотирование с помощью различных интернет-сервисов.

Обычной кражей денег и документов сегодня уже никого не удивишь, но с развитием интернет-технологий злоумышленники переместились в интернет, и продолжают заниматься «любимым» делом.

Так появилась новая угроза: интернет-мошенничества или **фишинг**, главная цель которого состоит в получении конфиденциальных данных пользователей —

логинов и паролей. На английском языке phishing читается как фишинг (от fishing — рыбная ловля, password — пароль).

Цель кибербезопасности - защитить данные, а также спрогнозировать, предотвратить и смягчить последствия любых вредоносных воздействий, которые могут нанести ущерб информации (удаление, искажение, копирование,передача третьим лицам и т.д.).

К методам защиты относят средства, меры и практики, которые должны защищать киберпространство от угроз - случайных или злонамеренных, внешних и внутренних.

В зависимости от различных способов классификации все возможные угрозы кибербезопасности можно разделить на следующие подгруппы:

− мошенничество;

− утечки информации;

− несанкционированный доступ;

- нежелательный контент;

− потеря данных;

− кибервойны.

Кибергигиена

- это правила соблюдения простых правил цифровой безопасности при работе с интернетом, ставшим полноценной частью нашей жизни.

- это набор ежедневных привычек, знаний и навыков, которые позволяют существенно снизить риски работы в Интернете.

Рекомендации по обеспечению безопасной работы в сети Интернет:

- никому не передавать конфиденциальные данные (логин, пароль), в том числе родственникам, коллегам;

- использовать сложные пароли, состоящие из букв, цифр и специальных символов, исключить использование паролей по умолчанию, (самым популярным паролем в мире является «123456»);

- регулярно осуществлять смену паролей, обеспечить их конфиденциальность;

- использовать в работе лицензионное программное обеспечение с установленными обновлениями безопасности;

- на всех устройствах, должно быть установлено лицензионное антивирусное программное обеспечение с актуальными обновлениями;

- не использовать общественные беспроводные сети и устройства для работы с личной информацией;

- не использовать программные продукты, полученные из сомнительных источников (пиринговые и файлообменные сети), модифицированные программные продукты, не посещать ресурсы с сомнительной репутацией;

- личную информацию вводить только при безопасном соединении (URL веб-сайт должен начинаться с «https://», в интерфейсе браузера должна появиться иконка замка);

- выполнять резервное копирование важной информации.

- не входите на незнакомые сайты.

- если к вам по почте пришел файл Word или Excel, даже от знакомого лица, прежде чем открыть, обязательно проверьте его на вирусы.

- если пришло незнакомое вложение, ни в коем случае не запускайте его, а лучше сразу удалите и очистите корзину.

- при общении в Интернет не указывать свои личные данные, а использовать псевдоним (ник).

- без контроля взрослых ни в коем случае не встречаться с людьми, с которыми познакомились в сети Интернет.

- не всей той информации, которая размещена в Интернете, можно верить.

- не оставляйте без присмотра компьютер с важными сведениям на экране.

- опасайтесь подглядывания через плечо.

- не сохраняйте важные сведения на общедоступном компьютере.

- если у вас внешняя камера, отключайте ее, когда не используете. Если же камера встроена, лучше дополнительно защитить ее (закрыть крышкой) – предугадать атаку, нельзя.

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Для чего нужен Интернет?
2. Какие существуют риски при пользовании интернетом, и как их можно снизить?
3. Какие виды мошенничества существуют в сети Интернет?
4. Как защититься от мошенничества в сети Интернет?
5. Что такое безопасный чат?

6. Ты долго общаешься с одним из форумчан, и он (она) предлагает тебе встретиться. Какое место для личного знакомства ты предпочтешь: парк, кино или кафе?

7. Письмо, которое пришло на твой почтовый ящик, содержит ссылку или файл. Как ты с ним поступишь?

8. При общении в интернете кто-то просит твой номер телефона. Ты дашь его? И если да, то кому?

9. Один или несколько онлайн-собеседников критикуют тебя, часто переходят на оскорбления, всячески переиначивают и высмеивают твои слова. После общения с ними надолго остается неприятный осадок. Что ты предпримешь?

10. К тебе в контакт-лист (в Skype, социальной сети, мессенджере и т. п.) попросился кто-то, кого ты не знаешь. Что ты сделаешь?

11. Как ты назовешься при регистрации на форуме?

Практическое занятие №17 Защита информации. Вредоносные программы и антивирусы

*Цель занятия***:**

1. Познакомить с методами защиты информации, вирусами и антивирусными средствами.
2. Формировать ОК1, ОК2,ОК7,ЛР 4,10,26,30

Исходные данные:

Папка на РС «Практическое занятие №17 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы.

Защитой информации называют технические и организационные меры по предотвращению неправомерного доступа к ней или ее искажения.

Защита информации была важна всегда. Одно из средств защитить информацию – зашифровать ее. Наука о шифровании называется криптографией.

Данные в компьютерных системах подвержены риску утраты из-за неисправности или уничтожения оборудования, а также риску хищения. Способы защиты информации включают использование аппаратных средств и устройств, а также внедрение специализированных технических средств и программного обеспечения.

Способы неправомерного доступа к информации:

Залогом успешной борьбы с несанкционированным доступом к информации и перехватом данных служит четкое представление о каналах утечки информации.

Интегральные схемы, на которых основана работа компьютеров, создают высокочастотные изменения уровня напряжения и токов. Колебания распространяются по проводам и могут не только трансформироваться в доступную для понимания форму, но и перехватываться специальными устройствами. В компьютер или монитор могут устанавливаться устройства для перехвата информации, которая выводится на монитор или вводится с клавиатуры. Перехват возможен и при передаче информации по внешним каналам связи, например, по телефонной линии.

Методы защиты

На практике используют несколько групп методов защиты, в том числе:

- **препятствие на пути предполагаемого похитителя**, которое создают физическими и программными средствами;
- **управление**, или оказание воздействия на элементы защищаемой системы;
- **маскировка**, или преобразование данных, обычно – криптографическими способами;
- **регламентация**, или разработка нормативно-правовых актов и набора мер, направленных на то, чтобы побудить пользователей, взаимодействующих с базами данных, к должному поведению;
- **принуждение**, или создание таких условий, при которых пользователь будет вынужден соблюдать правила обращения с данными;
- **побуждение**, или создание условий, которые мотивируют пользователей к должному поведению.

Каждый из методов защиты информации реализуется при помощи различных категорий средств.

Основные средства – организационные и технические:

Организационные средства защиты информации

Разработка комплекса организационных средств защиты информации должна входить в компетенцию службы безопасности.

Чаще всего специалисты по безопасности:

- **разрабатывают внутреннюю документацию**, которая устанавливает правила работы с компьютерной техникой и конфиденциальной информацией;
- **проводят инструктаж** и периодические проверки персонала; инициируют подписание дополнительных соглашений к трудовым договорам, где указана ответственность за разглашение или неправомерное использование сведений, ставших известных по работе;
- **разграничивают зоны ответственности**, чтобы исключить ситуации, когда массивы наиболее важных данных находятся в распоряжении одного из сотрудников; организуют работу в общих программах документооборота и следят, чтобы критически важные файлы не хранились вне сетевых дисков;
- **внедряют программные продукты**, которые защищают данные от копирования или уничтожения любым пользователем, в том числе топ-менеджментом организации;
- **составляют планы восстановления системы** на случай выхода из строя по любым причинам.

Технические средства защиты информации

Группа технических средств защиты информации совмещает аппаратные и программные средства.

Основные:

- резервное копирование и удаленное хранение наиболее важных массивов данных в компьютерной системе – на регулярной основе;
- дублирование и резервирование всех подсистем сетей, которые имеют значение для сохранности данных;
- создание возможности перераспределять ресурсы сети в случаях нарушения работоспособности отдельных элементов;

- обеспечение возможности использовать резервные системы электропитания;
- обеспечение безопасности от пожара или повреждения оборудования водой;
- установка программного обеспечения, которое обеспечивает защиту баз данных и другой информации от несанкционированного доступа.

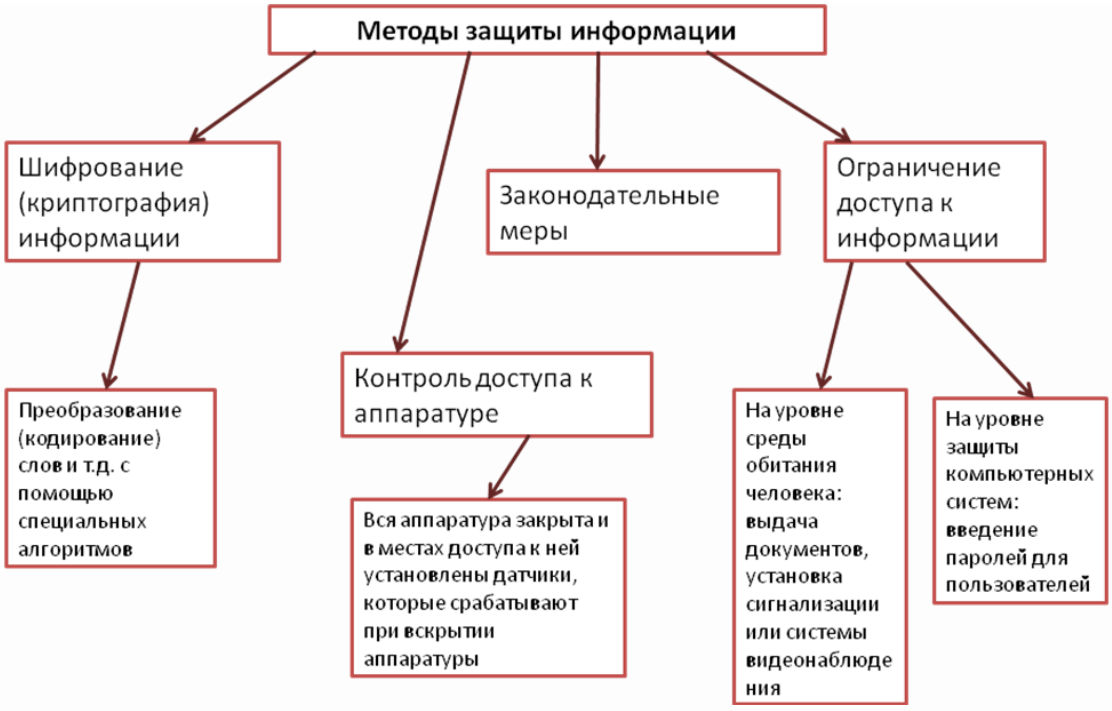
В комплекс технических мер входят и меры по обеспечению физической недоступности объектов компьютерных сетей, например, такие практические способы, как оборудование помещения камерами и сигнализацией.

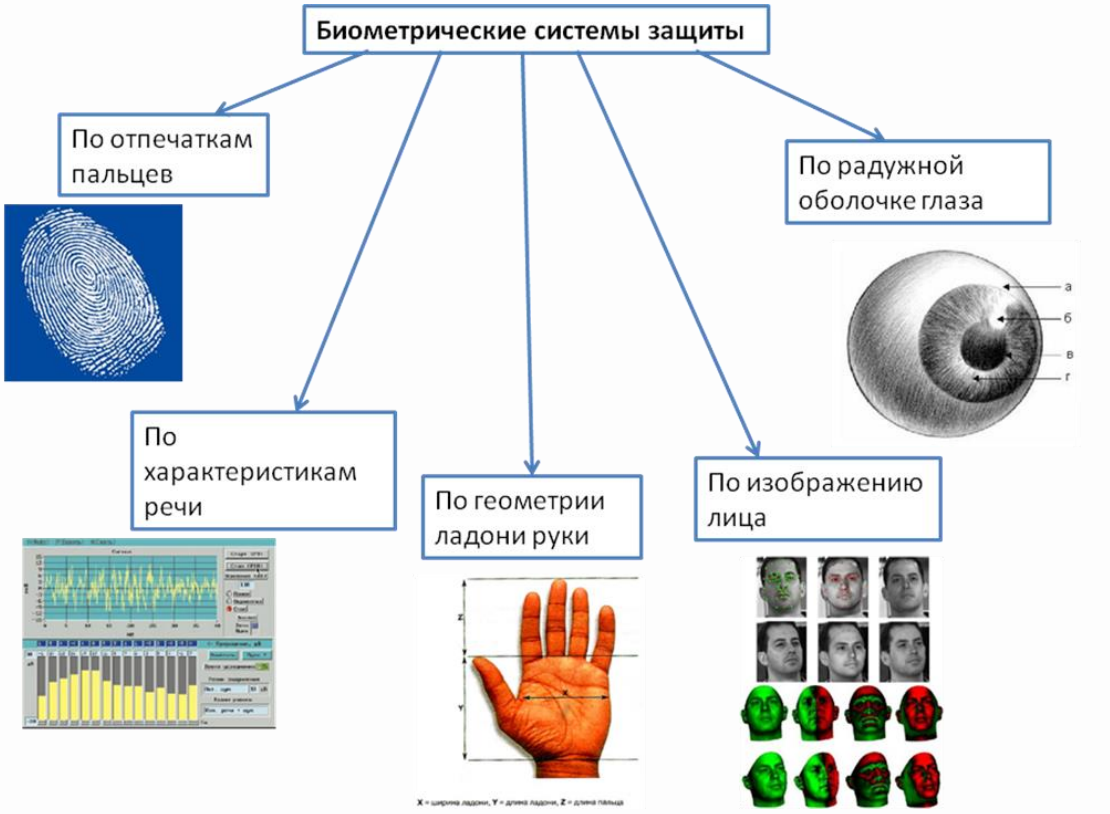
Аутентификация и идентификация

Чтобы исключить неправомерный доступ к информации применяют такие способы, как идентификация и аутентификация.

Идентификация – это механизм присвоения собственного уникального имени или образа пользователю, который взаимодействует с информацией. **Аутентификация** – это система способов проверки совпадения пользователя с тем образом, которому разрешен допуск.

Эти средства направлены на то, чтобы предоставить или, наоборот, запретить допуск к данным. Подлинность, как правила, определяется тремя способами: программой, аппаратом, человеком. При этом объектом аутентификации может быть не только человек, но и техническое средство (компьютер, монитор, носители) или данные. Простейший способ защиты – пароль.





Вредоносные программы и антивирусы.

Что такое вредоносное ПО

Если совсем просто, вредоносная программа - это программное обеспечение, специально разработанное для причинения вреда вам или вашему устройству.

Зараженный вредоносной программой ноутбук, настольный компьютер или мобильное устройство может работать медленнее или перестать работать вообще. Вредоносные программы также могут удалять или красть данные, что ставит под угрозу вашу конфиденциальность.

Как вредоносные программы проникают на ваше устройство

Вредоносные программы могут проникнуть в ваше устройство разными путями. Например, если вы пройдете по зараженной ссылке в рекламном объявлении или откроете вложение в спам-рассылке.

Если вовремя не принять меры, вредоносное ПО может нарушить работу вашего устройства и сделать ваши данные уязвимыми для кражи. К счастью, большинство вредоносных программ легко удалить с помощью Kaspersky Anti-Virus.

Типы вредоносного ПО

Вредоносные программы ведут себя по-разному. Одни могут скрываться во вложениях электронной почты или использовать веб-камеру вашего устройства, чтобы шпионить за вами. Другие (программы-вымогатели) могут удерживают ваши файлы в качестве заложников, пока вы не заплатите выкуп.

Существует несколько типов вредоносных программ. Давайте рассмотрим поведение каждого из них, чтобы вы смогли понять природу угроз, представляющих опасность для вашего устройства:

Черви

Червь – это вредоносная программа, которая многократно копирует сама себя, но не наносит прямого вреда безопасности. Черви могут распространяться по сетям, используя уязвимости каждого устройства.

Как и другие виды вредоносного ПО, червь может нанести вред вашему устройству, загружая на него вредоносные программы и замусоривая канал связи.

Adware

Adware — это программы, которые предназначены для показа рекламы на вашем компьютере, часто в виде всплывающих окон. Вы можете случайно согласиться на просмотр какого-то рекламного объявления и таким образом загрузить нежелательное ПО.

Иногда хакеры встраивают шпионское ПО в Adware, и тогда оно становится особенно опасным, поэтому будьте внимательны, не нажимаете на рекламное объявление, которое выглядит подозрительно.

Шпионское ПО

Шпионское ПО отличается от других типов вредоносных программ тем, что это не техническое определение, а общий термин для Adware, Riskware и троянских программ.

Шпионское ПО отслеживает вашу активность в интернете, наблюдает за тем, какие клавиши вы нажимаете, и собирает ваши личные данные.

Вирус

Вирус - это тип вредоносного ПО, способный к самовоспроизведению и распространению по всей системе на вашем устройстве.

Боты

Боты создаются автоматически для выполнения специальных операций.

Некоторые боты создаются для выполнения вполне легитимных задач. Например, они могут использоваться для сканирования веб-сайтов и сбора их контента с целью занесения этой информации в поисковые системы.

При злонамеренном использовании боты могут искать и собирать личные данные и передавать их киберпреступникам.

Программы-вымогатели

Программы-вымогатели блокируют доступ к вашему устройству или шифруют информацию, которая хранится на нем, а затем требуют плату за расшифровку файлов и восстановление работы системы.

Руткиты

Руткиты — это программы, используемые хакерами для предотвращения обнаружения при попытке получить несанкционированный доступ к компьютеру. Хакеры используют руткиты для удаленного доступа и кражи вашей информации.

Троянские программы

Троянская программа или троян - это вредоносная программа, маскирующаяся под обычный файл и выполняющая на компьютере пользователя вредоносные действия. При загрузке трояна сам пользователь может даже не подозревать, что на самом деле устанавливаете вредоносное ПО.

Троянские программы могут выполнять различные действия, включая кражу персональных данных. Примерами троянских программ являются:

- **Дро́пперы:** Эти программы используются хакерами, чтобы скрытно устанавливать вредоносные программы на компьютеры пользователей.
- **Троянские программы скрытой загрузки:** Эти программы способны загружать и устанавливать на компьютер-жертву новые версии вредоносных программ.
- **Шпионские программы:** Эти программы способны скрытно наблюдать за вашей активностью в интернете и отправлять информацию о ней киберпреступникам.

- **Банковские троянцы:** Эти программы маскируются под легитимные приложения и крадут банковскую информацию, когда вы их загружаете.

- **Бэкдоры:** Эти вредоносные программы скрытно проникают в ваш компьютер, используя уязвимости в установленном на нем программном обеспечении.

Почему важно использовать антивирусную защиту

Антивирусная защита - лучший способ обезопасить себя от интернет-угроз. Если вы не предпринимаете должных мер для защиты от вредоносных программ, ваше устройство и ваши личные данные (например, банковская информация) находятся под угрозой кражи и/или ненадлежащего использования. Решение для защиты от вредоносных программ – важный шаг для обеспечения безопасности ваших устройств и данных.

Ваше устройство заражено?

Ваш ноутбук, настольный компьютер или смартфон ведут себя странно? Большинство вредоносных программ ненавязчивы и не видны невооруженным глазом. Однако есть некоторые предупреждающие знаки, указывающие на то, что ваше устройство может быть заражено вредоносным ПО.

Для диагностики заражения вредоносным ПО обратите внимание на следующие предупреждающие знаки:

- Ваше устройство стало работать медленнее, и все операции занимают больше времени
- Появились приложения или программы, о которых вы ничего не знаете
- Приложения или программы постоянно «слетают» без видимой причины
- Сетевой трафик на вашем смартфоне необъяснимо возрос
- Ваш телефонный счет таинственным образом увеличился
- Вы видите всплывающие окна, когда ваш браузер закрыт
- Батарея вашего телефона быстро разряжается
- Ваш ноутбук, настольный компьютер или смартфон перегреваются

Удаление вредоносной программы

Если вы считаете, что ваш ноутбук, настольный компьютер или смартфон заражен, необходимо немедленно принять меры по удалению вредоносного ПО.

Вот 10 простых шагов для удаления вредоносных программ на вашем ноутбуке или компьютере:

- Скачайте и установите Kaspersky Anti-Virus
- Отключитесь от Интернета, чтобы избежать дальнейших действий вредоносной программы
- Загрузите компьютер в режиме «Safe Mode»
- Удалите все временные файлы с помощью функции «Disk Clean Up»
- Запустите проверку по требованию в Kaspersky Anti-Virus и следуйте приведенным инструкциям.
- Если вредоносное ПО обнаружено, удалите его или поместите файл в карантин
- Перезагрузите компьютер
- Поменяйте пароли, если вы считаете, что они могли быть взломаны
- Обновите программное обеспечение, браузер и операционную систему
- Повторно проверьте компьютер, чтобы убедиться в отсутствии других угроз.

Установите защитное решение на ваш смартфон

Смартфоны - это по сути небольшие компьютеры, которые помещаются в вашем кармане. Поскольку многие из нас пользуются смартфонами едва ли не чаще чем ноутбуками или настольными компьютерами, то следует помнить, что эти устройства также подвержены заражению вредоносным ПО. Следовательно, смартфоны, как и компьютеры, необходимо защитить от возможных вредоносных атак. Для максимальной защиты смартфона мы рекомендуем Kaspersky Antivirus для Android или Kaspersky Security Cloud для iOS, если вы пользователь iPhone.

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания

4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

5. Список используемых источников

6. Выводы и предложения

7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. В чем различие между антивирусными сканерами и мониторами?
2. Какие существуют признаки заражения компьютера вирусом?
3. Что необходимо сделать в первую очередь в случае заражения компьютера вирусом?
4. Как называются вредоносные программы, которые проникают на компьютер вместе с другими программами?
5. Как называется умышленно искаженная информация?
6. Как называется информация, к которой ограничен доступ?
7. Что называют защитой информации?
8. Какие существуют основные уровни обеспечения защиты информации?
9. Что такое несанкционированный доступ ?
10. Что такое целостность информации?
11. Что такое аутентификация?
12. Какие средства защиты информации предназначены для выполнения функций защиты информационной системы с помощью программных средств?
13. К каким средствам защиты информации относятся мероприятия, регламентирующие поведение сотрудника организации?

    A. Организационные средства

    B. Аппаратно-программные

    C. Криптографические средства

14. Какие средства защиты информации связаны применением инструментов шифрования?

    A. Организационные средства

    B. Аппаратно-программные

    C. Криптографические средства

ПРОФЕССИОНАЛЬНО-ОРИЕНТИРОВАННОЕ СОДЕРЖАНИЕ
технологический профиль
Раздел 5 Алгоритмизация и программирование. Аналитика и визуализация данных на Python

Тема 5.1 Понятие алгоритма и основные алгоритмические конструкции

Практическое занятие № 18 Алгоритмы и способы их описания. Линейные и условные алгоритмы. (составление трассировочных таблиц) Описание алгоритмов с помощью блок-схем

Цели занятия:

1.Сформировать представление об алгоритме и его свойствах;

2. Сформировать представление о способах их описания алгоритмов;

3.Сформировать представление о типах алгоритмов;

4. Сформировать представление о линейной и условных алгоритмических конструкциях;

5.Научить создавать трассировочные таблицы линейной и условных алгоритмических конструкций.

6. Формировать ОК5, ОК6, ЛР26, ЛР30

Исходные данные: теоретический материал

Содержание и порядок выполнения задания

Изучите теоретическую часть, данную в практическом занятии.

Изучите примеры и выполните задания. Результат работы алгоритма определяется с помощью трассировочных таблиц.

Теоретическая часть

Слово ***алгоритм*** происходит от латинской формы написания имени великого математика IX века **Аль-Хорезми**, который сформулировал правила выполнения арифметических действий.

Первоначально под алгоритмами понимали только правила выполнения четырёх арифметических действий над многозначными числами.

Алгоритм – описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов.

Алгоритмизация – процесс разработки алгоритма (плана действий) для решения задачи.

Шаг алгоритма – это каждое отдельное действие алгоритма.

Исполнитель – это объект, умеющий выполнять определенный набор действий. Исполнителем может быть человек, робот, животное, компьютер.

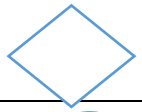
Система команд исполнителя (СКИ) – это все команды, которые исполнитель умеет выполнять.

Среда исполнителя – обстановка, в которой функционирует исполнитель.

Свойства алгоритма:

- Дискретность - (прерывность, раздельность) – разбиение алгоритма на шаги
- Результативность - получение результата за конечное количество шагов
- Массовость - использование алгоритма для решения однотипных задач
- Конечность - каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения
- Детерминированность - (определенность, точность) – каждое действие должно быть строго и недвусмысленно определено

Способы записи алгоритмов (блок-схема)

Условное обозначение	Назначение блока
	**Начало** или **конец** алгоритма
	**Ввод** или **вывод** данных. Внутри блока перечисляются данные через запятую.
	**Процесс.** Внутри блока записываются математические формулы и операции для обработки данных.
	**Проверка условия.** Внутри блока записываются логические условия. Имеет два выхода **Да(+)**и**Нет(-)**.
	**Соединительный блок**
	**Блок вывода информации на печатающее устройство**

	**Блок вывода информации на экран дисплея**
	**Направление.**

Алгоритмы могут быть заданы:

Словесно- ***Словесное*** задание описывает алгоритм с помощью слов и предложений естественного языка.

Таблично- ***Табличное*** задание служит для представления алгоритма в форме таблиц

Формульно – ***Формульное*** *задание* расчётными формулами.

Программой- задание на языке программирования

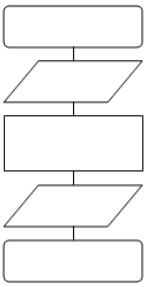
Графическое- задание или ***блок-схема*** – способ представления алгоритма с помощью геометрических фигур, называемых ***блоками***.

Типы алгоритмов

Алгоритмы бывают:

- *линейные*
- *разветвляющиеся*
- *циклические*

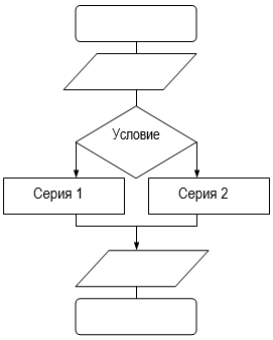
Алгоритм, в котором команды выполняются последовательно одна за другой, называется ***линейным алгоритмом.***



В ***разветвляющиеся алгоритмы*** входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (серий).

В алгоритмической структуре ***«ветвление»*** та или иная серия команд выполняется в зависимости от истинности ***условия.***

Условие может быть либо истинным, либо ложным.

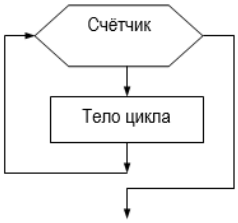


В ***циклические алгоритмы*** входит последовательность команд, выполняемая многократно. Такая последовательность команд называется ***телом цикла.***

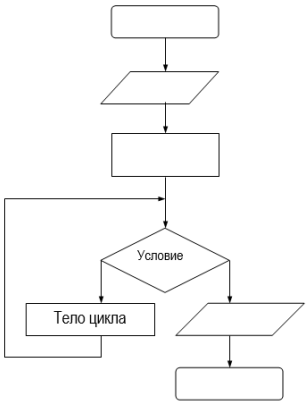
В алгоритмической структуре ***«цикл»*** серия команд (тело цикла) выполняется многократно.

Циклические алгоритмические структуры бывают двух типов:

- *циклы со счётчиком*, в которых тело цикла выполняется определённое количество раз;



- *циклы с условием, в которых тело цикла выполняется, пока условие истинно.*



Примеры выполнения заданий работы

Пример 1. Определить площадь трапеции по введенным значениям оснований (a и b) и высоты (h).

Запись алгоритма в виде блок-схемы (рис. 1):

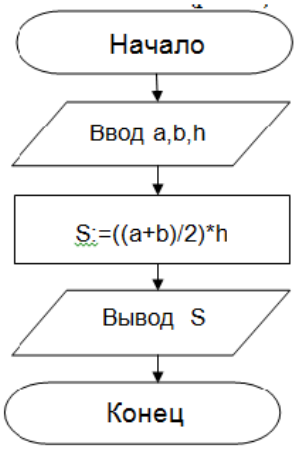
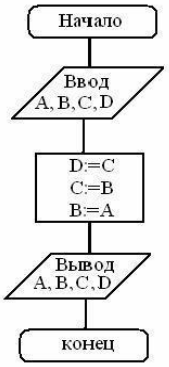


Рисунок 1. Блок-схема линейного алгоритма

Пример 2. Дан алгоритм в виде блок-схемы (рис. 11).

Найти A, B, C, D, если изначально:



a)A=0,B=0,C=5,D=10

Результат работы алгоритма определяется с помощью трассировочных таблиц:

a) A=0, B=0, C=5, D=10.

Шаг		1
Исходные значения	A	0
	B	0
	C	5
	D	10
Результат выполнения	A	0
	B	0
	C	0
	D	5
Вывод значений		0,0,0,5

Пример 2. Определить среднее арифметическое двух чисел, если **a** положительное и частное (a/b) в противном случае.

Запись алгоритма в виде блок-схемы (рис. 2):

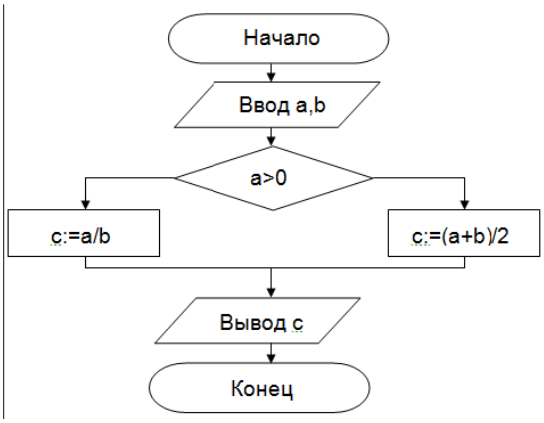


Рисунок 2. Блок-схема алгоритма с ветвлением

***Задание* 1**

Реализован некоторый алгоритм в виде блок- схемы. Найти A, B на выходе блок-схемы с заполнение трассировочных таблиц, если изначально, :

1 вариант A=0,B=0;

2вариант A=0,B=5;

3 вариант A=10,B=20;

4 вариант A=10,B=10.

5 вариант A=3, B=0;

6 вариант A=0, B=2;

7вариант A=1,B=2;

8 вариант A=12, B=15

9 вариант A=0, B=0;

10 вариант A=2, B=4;

11 вариант A=12, B=24;

12 вариант A=11, B=19

13 вариант A=3, B=8;

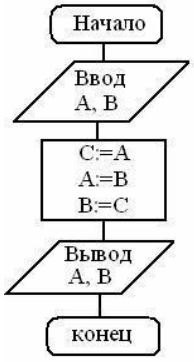
14 вариант A=2, B=7;

15 вариант A=19, B=20;

16 вариант A=15, B=10

17 вариант A=60, B=20;

18 вариант A=20, B=65;



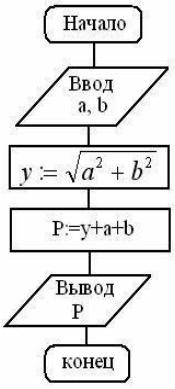
***Задание* 2.**

Даны длины двух катетов (a, b) прямоугольного треугольника. Определить периметр этого треугольника (P)с заполнением трассировочных таблиц (см. блок-схему на рис.), если:

а) a=3,b=4;

б) a=0,b=3;

в)a=6,b=8;



г)a=9,b=12.

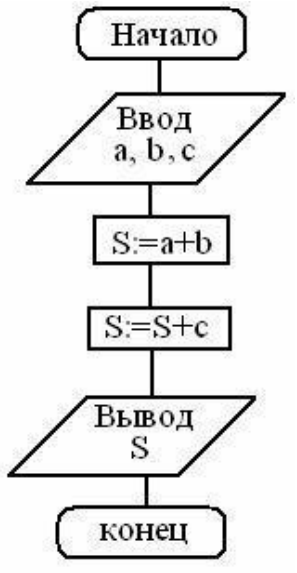
***Задание*3.**

Реализован некоторый алгоритм в виде блок- схемы (рис.). По данной блок-схеме вычислить S, заполнить трассировочные таблицы, если:

а)a=1, b=2, c=3;

б)a=9, b=0, c=1;

в) a=5, b=6, c=9.



***Задание*4.**

Дана блок-схема (рис.19). Начальные условия: a=8, b=2. Тогда после исполнения алгоритмазначение переменной g будет равно …

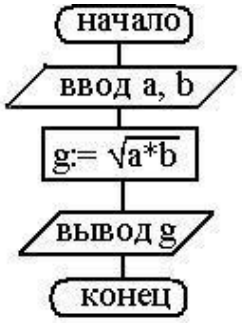
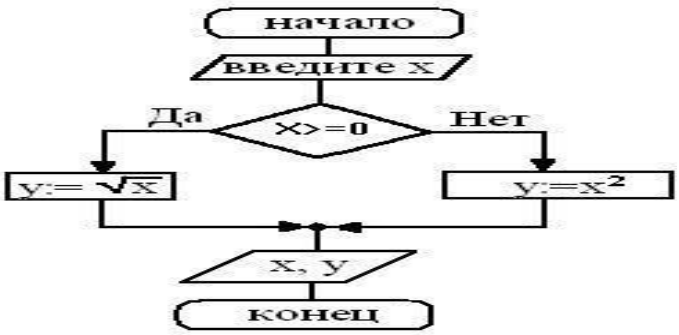


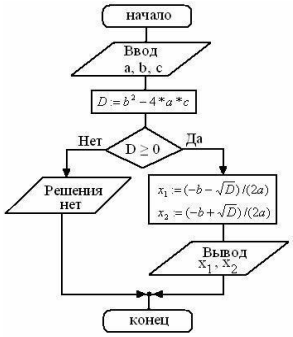
Рис. 19

Задание5. Вычислить значение функции y. Результат работы алгоритма определяется с помощью трассировочных таблиц:

(см. блок-схему на рис.), если: а) x=0; б) x=1; в) x= – 5.



Задание6. Используя блок-схему (рис.), найти корни уравнения $ax^2 + bx + c = 0$ ($a \neq 0$) , если:



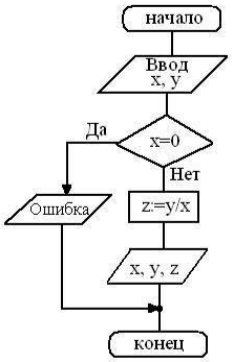
а)a=1, b=2,

б)a=1, b=4,

в) a=3, b=–8,c=3.

Результат работы алгоритма определяется с помощью трассировочных таблиц

Задание7. Реализован некоторый алгоритм в виде блок- схемы (рис. 22).



Что получится на выходе блок-схемы, если:

а) x=0, y=1; б) x=2, y=4; в) x=6, y=0?

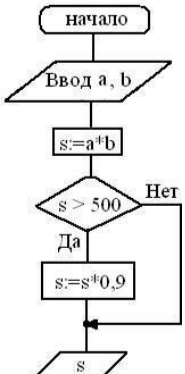
Результат работы алгоритма определяется с помощью трассировочных таблиц

Задание8. На блок-схеме (рис. 23) представлен алгоритм вычисления стоимости покупки с учетом скидки, где a – цена, b

– количество, s – сумма. Какой будет результат на выходе блок- схемы, если:

а) a=50, b=8;

б) a=200, b=5;



в) a=300, b=1;

г) a=800, b=4?

Результат работы алгоритма определяется с помощью трассировочных таблиц

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Список используемых источников
4. Выводы и предложения
5. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что такое алгоритм?

2. Назовите свойства алгоритма и способы записи алгоритмов.

4. Изобразите элементы блок-схемы.

5. Какие существуют виды алгоритмов.

6.Как представляется блок-схемой циклический алгоритм с пост условием?

7.Как представляется блок-схемой циклический алгоритм с пред условием?

Практическое занятие № 19 Циклические алгоритмы (составление трассировочных таблиц). Описание алгоритма с помощью блок-схем

Цели занятия:

1. Сформировать представление о циклических алгоритмических конструкциях;

2.Научить создавать трассировочные таблицы циклических алгоритмических конструкций.

3. Формировать ОК5, ОК6, ЛР26, ЛР30

Исходные данные: теоретический материал

Содержание и порядок выполнения задания

Изучите теоретическую часть, данную в практическом занятии.

Изучите примеры и выполните задания. Результат работы алгоритма определяется с помощью трассировочных таблиц.

Теоретическая часть

Циклическая структура (или повторение) предусматривает повторное выполнение некоторого набора действий.

Циклы позволяют записать длинные последовательности операций обработки данных с помощью небольшого числа повторяющихся команд.

Итерационным называется цикл, число повторений которого не задается, а определяется в ходе выполнения цикла. В этом случае одно повторение цикла называется итерацией.

Рекурсия – это такая ситуация, когда некоторый алгоритм непосредственно или через другие алгоритмы вызывает себя в качестве вспомогательного. Сам алгоритм при этом называется рекурсивным.

В качестве примера использования рекурсии рассмотрим задачу поиска файлов. Пусть нужно получить список всех фай-лов, например, с расширением bmp, которые находятся в указанном пользователем каталоге и во всех подкаталогах этого каталога.

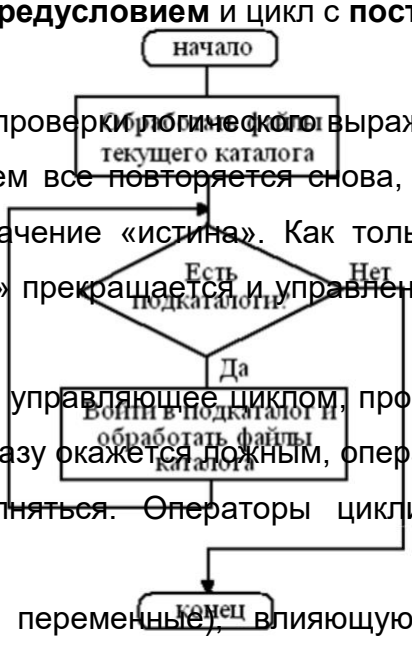
Словесно алгоритм обработки каталога может быть представлен так:

1. Вывести список всех файлов, удовлетворяющих критерию запроса.

2. Если в каталоге есть подкаталоги, то обработать каждый из этих каталогов.

Приведенный алгоритм, блок-схема которого представлена на рис. является рекурсивным. Для того чтобы обработать подкаталог, процедура обработки текущего каталога должна вызвать сама себя.

Различают цикл с **предусловием** и цикл с **постусловием.**

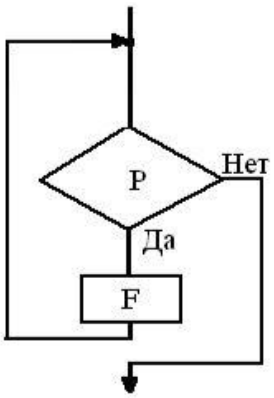


Цикл начинается с проверки логического выражения «P». Если оно истинно, то выполняется «F», затем все повторяется снова, до тех пор, пока логическое выражение сохраняет значение «истина». Как только оно становится ложным, выполнение операций «F» прекращается и управление передается по программе дальше.

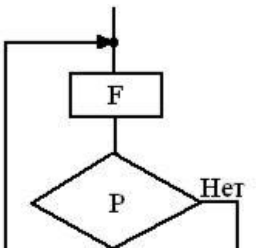
Так как выражение, управляющее циклом, проверяется в самом начале, то в случае, если условие сразу окажется ложным, операторы циклической части «F» могут вообще не выполняться. Операторы циклической части «F» должны изменять.

Переменную (или переменные), влияющую на значение логического выражения, иначе программа «зациклится» – будет выполняться бесконечно. Рассмотренная циклическая конструкция называется

цикл «**пока**», или цикл с **предусловием** (см. рис.)



Существует и иная конструкция цикла, которая предусматривает проверку условия после выполнения команд, встроенных внутрь цикла. Это цикл с **постусловием** (см. рис.)

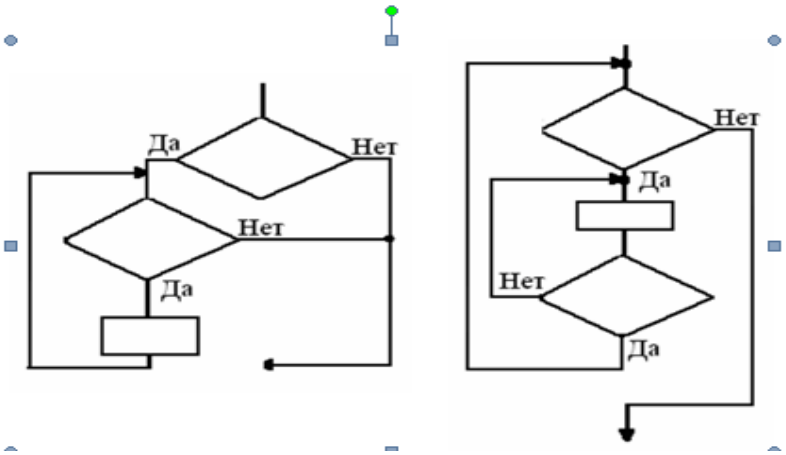


Циклические структуры можно комбинировать одну с другой – как путем организации их следований, так и путем создания суперпозиций (вложений одной структуры в другую).

Схематические изображения нескольких суперпозиций базовых алгоритмических структур представлены на рис..

Алгоритм типа «цикл, вложенный в неполную развилку»

Алгоритм типа «цикл в цикле»



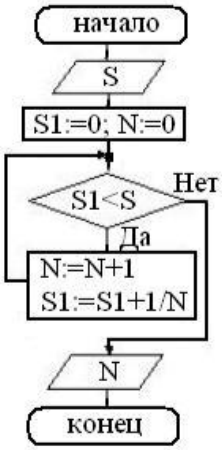
ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ НА АЛГОРИТМЫ

Цикл с предусловием

Пример 1. Дана блок-схема (рис. 14). Какое значение будет иметь N на выходе, если:

а) S=1,1; б) S=2,09?

Для определения результата воспользуемся трассировочными таблицами (а):



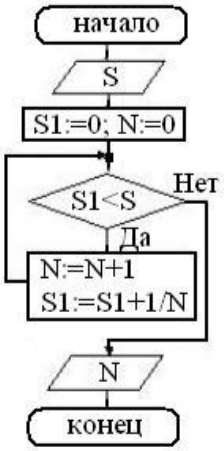
а) S=1,1:

Шаг	1	2	3
Значение S1	0	1	1,5
Значение N	0	1	2
Тело цикла	0 < 1,1 (Да)	1 < 1,1 (Да)	1,5 < 1,1 (Нет)
Результат выполнения	N=0+1=1 S1=0+1/1=1	N=1+1=2 S1=1+1/2=1,5	N = 2
Вывод значения N			2

Задание №1

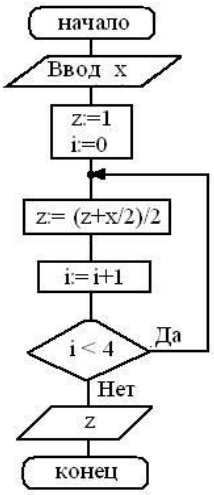
Какое значение будет иметь N на выходе, если:

S=2,09. Для определения результата составьте трассировочную таблицу



Цикл с постусловием

Пример 2. Дана блок-схема (рис.) Какое значение будет иметь z на выходе, если: а) x=2



Для определения результата воспользуемся трассировочной таблицей

а) x=2:

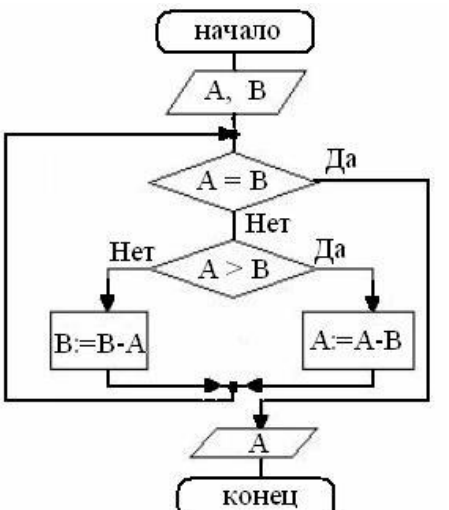
Шаг	1	2	3	4
Начальное значение z	1	1	1	1
Значение i	0	1	2	3
Результат выполнения z	z = (1+2/2)/2=1	z = (1+2/2)/2=1	z = (1+2/2)/2=1	z = (1+2/2)/2=1
Результат выполнения i	i = 0+1=1	i = 1+1=2	i = 2+1=3	i = 3+1=4
Тело цикла	1<4(Да)	2<4(Да)	3<4(Да)	4 < 4 (Нет)
Вывод z				1

Задание №2

Определите результат при x=4, x=6 , используя предыдущую блок-схему .Для определения результата воспользуйтесь трассировочной таблицей.

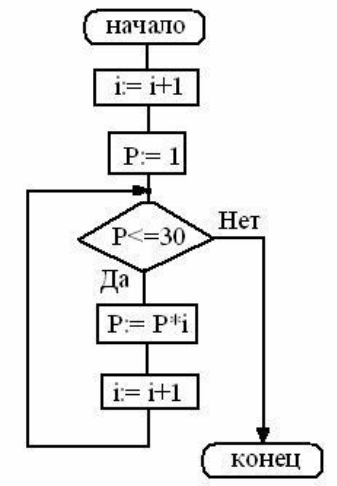
Задание №3

На блок-схеме (рис.)представлен алгоритм Евклида, определяющий наибольший общий делитель (НОД) для двух натуральных чисел A и B. Найти A на выходе блок-схемы, если: а) A=5, B=10; б) A=8, B=8



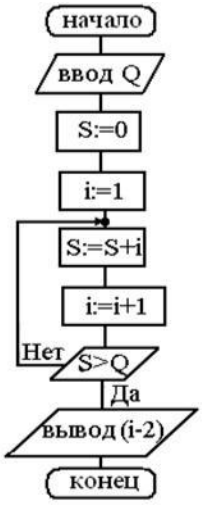
Задание №4

Дана блок-схема (рис. 26). Тогда после исполнения алгоритма переменная ***i*** примет значение …



Задание №5

Реализован некоторый алгоритм в виде блок-схемы (рис.) Что получится на выходе блок-схемы, если: Q=2;



Результат работы алгоритма определяется с помощью трассировочных таблиц

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Список используемых источников

5. Выводы и предложения
6. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что называется алгоритмом?
2. Для чего необходимы трассировочные таблицы?
3. Как представляется блок-схемой циклический алгоритм с пост условием?
4. Как представляется блок-схемой циклический алгоритм с пред условием?

Тема 5.2 Списки, графы, деревья
Практическое занятие № 20 Структура информации. Графы. Введение и понятия

Цель занятия:

1.Познакомить обучающихся с основными понятиями теории графов;

2. Формировать ОК5, ОК6, ЛР26, ЛР30

Исходные Теоретический материал, схемы, задания о вариантам

Исходные данные:

Папка на РС «Практическое занятие №20 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам.

Теоретическая часть

Понятие графа

Граф – это множество точек или вершин и множество линий или ребер, соединяющих между собой все или часть этих точек.

Вершины, прилегающие к одному и тому же ребру, называются смежными.

Два ребра, у которых есть общая вершина, также называютсясмежными(или соседними).

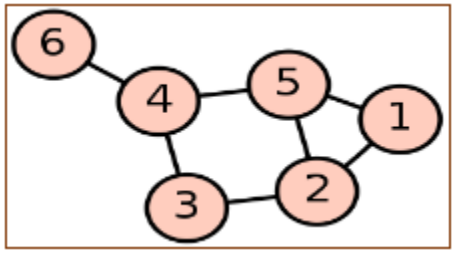


Рис. 1. Граф с шестью вершинами и семью ребрами

Элементы графа

Петля – это дуга, начальная и конечная вершина которой совпадают.

Пустым (нулевым) называется граф без ребер.

Полным называется граф, в котором каждые две вершины смежные.

Нулевой граф – граф, состоящий из «изолированных» вершин.

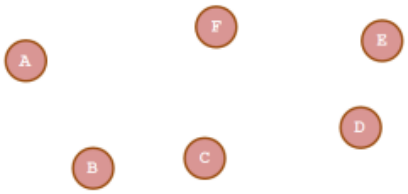


Рис. 2. Нулевой граф

Неполный граф – графы, в которых не построены все возможные ребра.

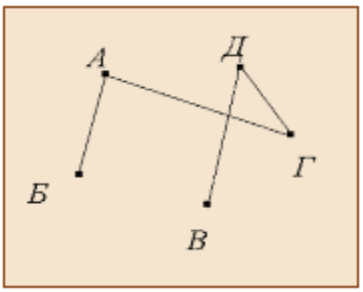


Рис. 3 Неполный граф

Степень графа

Количество рёбер, выходящих из вершины графа, называется степенью вершины. Вершина графа, имеющая нечётную степень, называется нечетной, а чётную степень – чётной.

Если степени всех вершин графа равны, то граф называется однородным. Таким образом, любой полный граф — однородный.

Кружки называются вершинами графа, линии со стрелками - дугами, без стрелок - ребрами. Граф, в котором направление линий не выделяется (все линии являются ребрами),

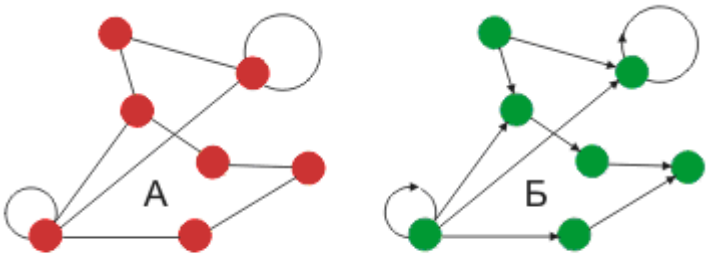


рис.1

называется неориентированным (рис. 1, А); граф, в котором направление линий принципиально (линии являются дугами) называется ориентированным (рис. 1, Б).

Ориентированный граф

Граф называется ориентированным (или орграфом), если некоторые ребра имеют направление. Это означает, что в орграфе некоторая вершина может быть соединена с другой вершиной, а обратного соединения нет. Если ребра ориентированы, что обычно показывают стрелками, то они называются дугами.

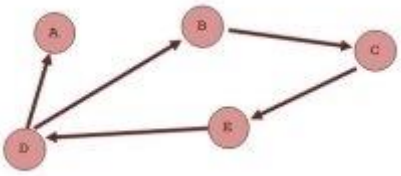


Рис. 4. Ориентированный граф

Ориентированный и неориентированный графы

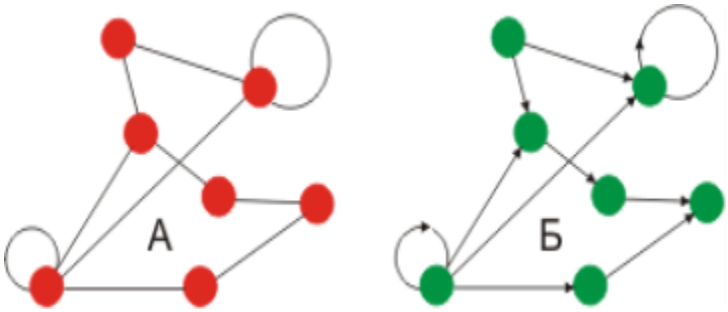


Рис. 5. Примеры неориентированного и ориентированного графов (А и Б)

Задание 1. Построить граф по заданному условию:

В соревнованиях по футболу участвуют 6 команд. Каждую из команд обозначили буквами A, B, C, D, E и F. Через несколько недель некоторые из команд уже сыграли друг с другом:

A с C, D, F;

B с C, E, F;

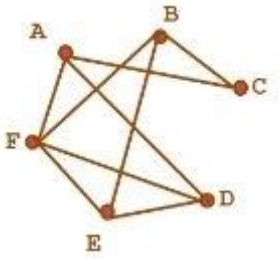
C с A, B;

D с A, E, F;

E с B, D, F;

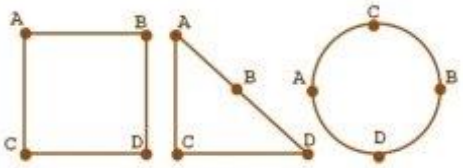
F с A, B, D.

Ответ:



Задание 2.

Определить изображают ли фигуры на рисунке один и тот же граф или нет.



1. 2) 3)

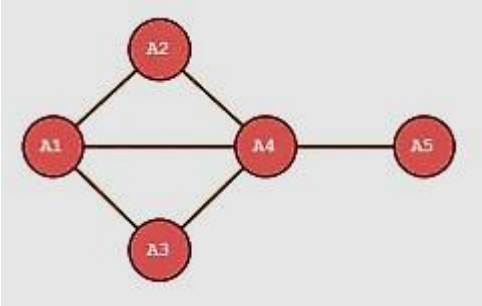
Ответ: Рисунок 1 и рисунок 2 являются изображениями одного графа. Рисунок 3 изображением другого графа.

Путь в графе

Путём в графе называется такая последовательность ребер, в которой каждые два соседних ребра имеют общую вершину и никакое ребро не встречается более одного раза.

Задание 3. Определить какая из перечисленных последовательностей путём не является.

1. (A1 A4); (A4 A5).
2. (A1 A2); (A2 A4); (A4 A5).
3. (A1 A4); (A4 A2); (A2 A1); (A1 A4); (A4, A5).
4. (A1 A4); (A4 A2); (A2 A1); (A1 A3); (A3 A4); (A4, A5).

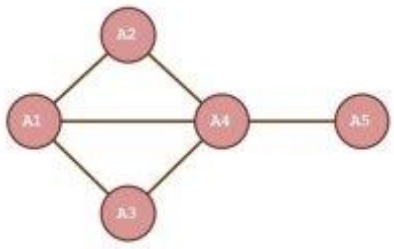


Ответ: Третья последовательность (A1 A4); (A4 A2); (A2 A1); (A1 A4); (A4, A5)

Путь называется простым, если он не проходит ни через одну из вершин графа более одного раза.

Задание 4. Определите, какие последовательности ребер являются путями, и какие из них простые. Если последовательность не является путем укажите почему.

1. (A1 A4); (A4 A5).
2. (A1 A2); (A2 A4); (A4 A5).
3. (A1 A4); (A4 A2); (A2 A1); (A1 A4); (A4, A5).
4. (A1 A4); (A4 A2); (A2 A1); (A1 A3); (A3 A4); (A4, A5).



Ответ: Первая, вторая и четвертая последовательности являются путями, а третья нет, т.к. ребро (A1, A4) повторяется.

Первая и вторая последовательность являются простыми путями, а четвертая нет, т.к. вершины A1 и A4 повторяются.

Понятие цикла в графе

Циклом называется путь, в котором совпадают его начальная и конечная вершины.

Простым циклом в графе называется цикл, не проходящий ни через одну из вершин графа более одного раза.

Задание 5. Назовите в графе циклы, содержащие

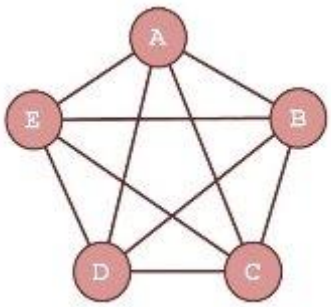
a) 4 ребра;

b) 6 ребер;

c) 5 ребер;

d) 10 ребер.

Какие из этих циклов являются простыми?



Ответ:

Решение:

a. (AB, BC, CE, EA), (CD, DA, AB, BC), (EB, BC, CD, DE) и т.д. – простые циклы.

b. (DB, BE, EA, AB, BC, CD), (EC, CA, AB, BC, CD, DE) ит.д. – циклы.

c. (AB, BC, CD, DE, EA), (AC, CE, EB, BD, DA) ит.д. – простыециклы.

(AC, CE, EB, BD, DA, AB, BC, CD, DE, EA), (EB, BD, DA, AC, CE, EA, AB, BC, CD, DE) и т.д. – циклы.

Задания по вариантам

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Список используемых источников
4. Выводы и предложения
5. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

Изобразите графически:

1. Неориентированное и ориентированное ребро;

2. Неориентированный граф G(V,E), заданный множеством

V={v0, v1, v2, v3, v4, v5}

E = {e1=(v0, v1), e2=(v0, v2}, e3=(v1, v2), e4=(v1,v4), e5=(v2, v5), e6=(v3, v4)}

3. Плоский граф;
4. Полный неориентированный граф на трех, четырех и пяти вершинах;
5. Неполный ориентированный граф на пяти вершинах;
6. Петлю графа.
7. Цикл графа.

Практическое занятие № 21 Способы задания графов. Алгоритм построения дерева решений

Цель занятия:

1. Научить задавать графы, выполнять вычисления степеней вершин графа;
2. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, таблицы, схемы, задания по вариантам.

Исходные данные:

Папка на РС «Практическое занятие №21 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам.

Теоретическая часть

1 Граф G - совокупность двух множеств: вершин V и ребер E, между которыми определено отношение инцидентности. Если |V(G)|=n, |E(G)|=m, то граф G есть (n,m) граф, где n - порядок графа, m - размер графа.

2 Каждое ребро e из E инцидентно ровно двум вершинам v', v'', которые оно соединяет. При этом вершина v' и ребро e называются инцидентными друг другу, а вершины v' и v'' называются смежными.

3 Ребро (v',v'') может быть ориентированным и иметь начало (v') и конец (v'') (дуга в орграфе).

4 Ребро (v,v) называется петлей (концевые вершины совпадают).

5 Граф, содержащий ориентированные ребра (дуги), называется орграфом.

6 Граф, не содержащий ориентированные ребра (дуги), называется неографом.

7 Ребра, инцидентные одной паре вершин, называются параллельными или кратными.

8 Конечный граф - число вершин и ребер конечно.

9 Пустой граф - множество ребер пусто (число вершин может быть произвольным).

10 Полный граф - граф без петель и кратных ребер, каждая пара вершин соединена ребром.

11 Локальная степень вершины - число ребер ей инцидентных.

12 Внеографе сумма степеней всех вершин равна удвоенному числу ребер (лемма о рукопожатиях). Петля дает вклад, равный 2 в степень вершины.

13В орграфе сумма входящих ребер всех вершин равна сумме исходящих ребер всех вершин и равна числу ребер графа.

14 Графы равны, если множества вершин и инцидентных им ребер совпадают.

15 Графы, отличающиеся только нумерацией вершин и ребер, называются изоморфными.

16 Способы задания графов: − явное задание графа как алгебраической системы; − геометрический; − матрица смежности; − матрица инцидентности

17 Матрица инцидентности: По вертикали указываются вершины, по горизонтали - ребра. aij=1 если вершина i инцидентна ребру j, в противном случае aij=0. Если ребро - петля, то aij=2. Матрицей инцидентности (инциденций) ориентированного графа называется матрица, для которой aij=1, если вершина является началом дуги ,aij= – 1, если является концом дуги, в остальных случаях aij=0.

18 Матрица смежности - квадратная симметричная матрица. По горизонтали и вертикали - все вершины.aij= число ребер, соединяющее вершины i,j. Матрицей смежности ориентированного графа называется матрица, для которой aij=1, если вершина является началом дуги, в остальных случаях aij=0.

19 Маршрут - последовательность ребер, в которых каждые два соседних ребра имеют общую вершину.

20 Маршрут, в котором начало и конец совпадают - циклический.

21 Маршрут в неографе, в котором все ребра разные - цепь.

22 Маршрут в орграфе, в котором все дуги разные - путь.

23 Вершины связанные, если существует маршрут из одной вершины в другую.

24 Связанный граф - если все его вершины связаны.

25 Плоский граф - граф с вершинами, расположенными на плоскости и непересекающимися ребрами.

26 Вершины графа, которые не принадлежат ни одному ребру, называются изолированными.

27 Дерево - связный граф без циклов.

Пример выполнения:

Исходные данные:

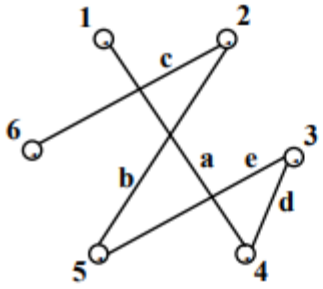
1 Задать неограф, представленный множеством вершин и ребер, графически и матрицами, преобразовать граф в плоский, вычислить степени его вершин.

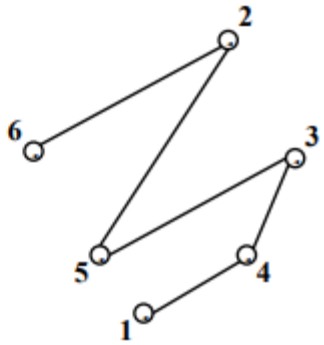
V = {1; 2; 3; 4; 5; 6};**E = {a; b; c; d; e}**

E = {(1; 4); (2; 5); (2; 6); (3; 4); (3; 5)}

Решение:

1Изобразим граф, соединив вершины: Ребро a соединяет вершины 1 и 4, b соединяет вершины 2 и 5 и т. д. Затем преобразуем этот граф в плоский:





2 Составим матрицу смежности. В первом столбце и первой строке выпишем вершины. Ребру a инцидентны вершины 1 и 4, следовательно в колонке 1 и строке 4 ставим 1, а также колонке 4 и строке 1 ставим 1. Ребру b инцидентны вершины 2 и 5, следовательно в колонке 2 в строке 5 и колонке 5 строке 2 ставим 1 и т.д. Остальные ячейки таблицы содержат нули.

3 Составим матрицу инцидентности. В первом столбце выпишем вершины, первой строке – ребра. Ребру a инцидентны вершины 1 и 4, следовательно в колонке a в строке 1 и строке 4 ставим 1. Ребру b инцидентны вершины 2 и 5, следовательно в колонке b в строке 2 и строке 5 ставим 1и т.д. Остальные ячейки таблицы заполняем нулями

Матрица смежности

	1	2	3	4	5	6
1	0	0	0	1	0	0
2	0	0	0	0	1	1
3	0	0	0	1	1	0
4	1	0	1	0	0	0
5	0	1	1	0	0	0
6	0	1	0	0	0	0

Матрица инцидентности

	a	b	c	d	e
1	1	0	0	0	0
2	0	1	1	0	0
3	0	0	0	1	1
4	1	0	0	1	0
5	0	1	0	0	1
6	0	0	1	0	0

4 Вычислим степени вершин:

ρ (1) = 1 ρ (2) = 2 ρ (3) = 2 ρ (4) = 2 ρ (5) = 2 ρ (6) = 1

ρ (1) + ρ (2) + ρ (3) + ρ (4) + ρ (5) + ρ (6) = 10 = 2 * q

q = 5 (ребер 5)

Исходные данные:

2Задать граф, представленный матрицей инцидентности, алгебраически, графически и матрицей смежности, преобразовать граф в плоский, вычислить степени его вершин

	a	b	c	d	e	f
1	-1	-1	0	0	0	0
2	1	0	-1	1	0	0
3	0	0	0	-1	0	0
4	0	0	1	0	1	0
5	0	0	0	0	-1	-1
6	0	1	0	0	0	1

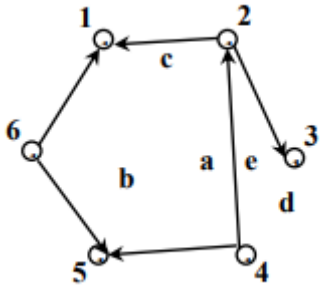
Решение:

1 Количество вершин – 6. V = {1; 2; 3; 4; 5; 6}.

2 Ребро a выходит из вершины 2, т.к. в ячейке (2; 1) стоит 1, а приходит в вершину 1 (в ячейке (1; 1) находится -1) и т.д.

Получим множество E = {(2; 1); (6; 1); (4; 2); (2; 3); (4; 5); (6; 5)}

3Изобразим граф, соединив вершины, этот граф уже плоский, т.к. ребра не пересекаются:



4 Составим матрицу смежности.

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	0	1	0	0	0
3	0	0	0	0	0	0
4	0	1	0	0	1	0
5	0	0	0	0	0	0
6	1	0	0	0	1	0

5 Вычислим степени вершин:

ρ1 (1) = 0

ρ1 (2) = 2

ρ1 (3) = 0

ρ1 (4) = 2

ρ1 (5) = 0

ρ1 (6) = 2

ρ2 (1) = 2

ρ2 (2) = 1

ρ2 (3) = 1

ρ2 (4) = 0

ρ2 (5) = 2

ρ2 (6) = 0

ρ1 (1) + ρ1 (2) + ρ1 (3) + ρ1 (4) + ρ1 (5) + ρ1 (6) = 6

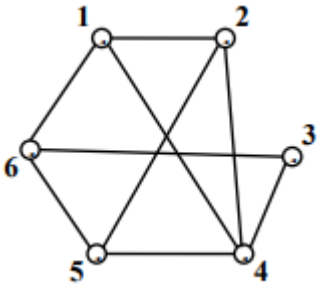
ρ2 (1) + ρ2 (2) + ρ2 (3) + ρ2 (4) + ρ2 (5) + ρ2 (6) = 6

q = 6(ребер 6)

Пример варианта задания

1. Орграф V = {1; 2; 3; 4; 5; 6}

E = {(1; 3); (1; 6); (2; 5); (3; 2); (3; 4); (4; 1); (4; 5); (5; 3); (6; 2)}



.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Список используемых источников
4. Выводы и предложения
5. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что такое граф?

2. Что такое инцидентное ребро или инцидентная вершина?

3 Что такое петля?

4 Какое ребро называется ориентированным?

5 Что такое кратные ребра?

6 Что такое неограф?

7 Что такое орграф?

Практическое занятие № 22 Решение логических задач с помощью графов. Анализ алгоритмов в профессиональной области.

Цель занятия:

1.обобщить и систематизировать знания о графах, их видах и свойствах;

2.рассмотреть типы задач, которые можно решить с использованием графов;

3.отработать навыки преобразования ориентированного графа в дерево;

4.сформировать навыки построение путей в графе, поиска кратчайшего пути;

5.Формировать ОК 01, ОК 02.

Исходные данные:

Папка на РС «Практическое занятие №22 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задание по вариантам.

Теоретическая часть

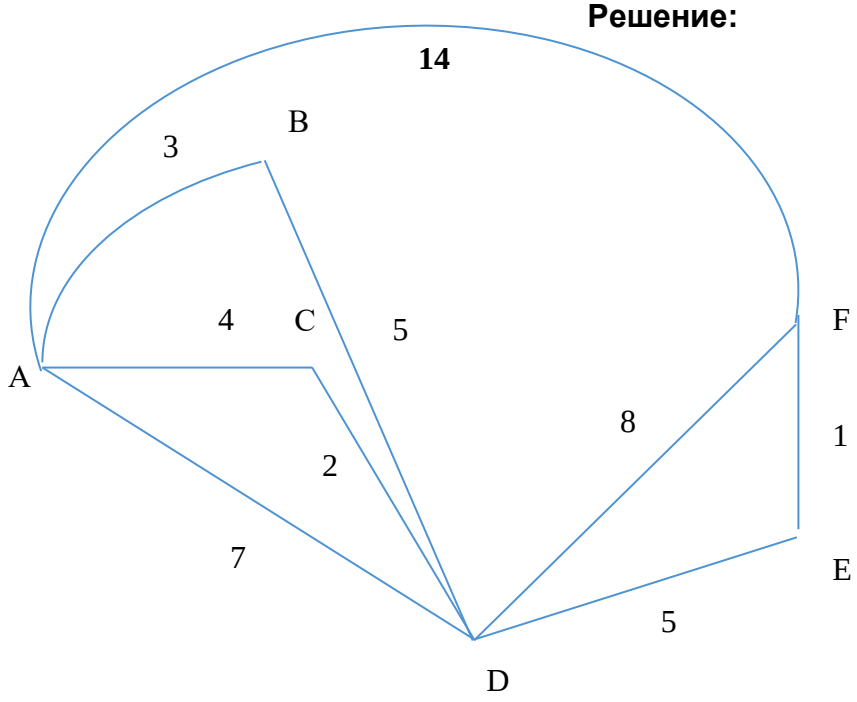
При решении логических задач по информатике часто приходится сталкиваться с заданиями, которые вызывает затруднение при решении. Если же построить граф, то задача существенно упрощается. Теория графов позволяет решать наиболее легким способом, наглядно многие логические задачи, которые способствуют развитию мышления и интеллекта. Слово «граф» означает картинку, где нарисовано несколько точек, некоторые из которых соединены линиями. Графами являются блок – схемы программ для ЭВМ, сетевые графики строительства, где вершины – события, означающие окончания работ на некотором участке, а ребра, связывающие эти вершины, - работы, которые возможно начать по совершении одного события и необходимо выполнить для совершения следующего.

Задача 1. Между населёнными пунктами A, B, C, D, E, F построены дороги, протяжённость которых приведена в таблице. Отсутствие числа в таблице означает, что прямой дороги между пунктами нет.

	A	B	C	D	E	F
A		3	4	7		14
B	3			5		
C	4			2		
D	7	5	2		5	8
E				5		1
F	14			8	1	

Определите длину кратчайшего пути между пунктами A и F при условии, что передвигаться можно только по указанным в таблице дорогам.

Решение:



AF=14

AF=AB+BD+DF=3+5+8=16

AF=AC+CD+DF=4+2+8=14

AF=AD+DE+EF=7+5+1=13

AF= AC+ CD+ DE+ EF=4+2+5+1=12

Ответ: 12

Задача 2

Сколькими способами можно рассадить в ряд на три стула трёх студентов? Выписать все возможные случаи.

Решение этой задачи удобнее всего представить в виде дерева. За его корневую вершину возьмём произвольную точку плоскости О.

На первый стул можно посадить любого из трёх студентов — обозначим их А, В и С. На схеме это соответствует трём ветвям, исходящим из точки О (рис. 1).

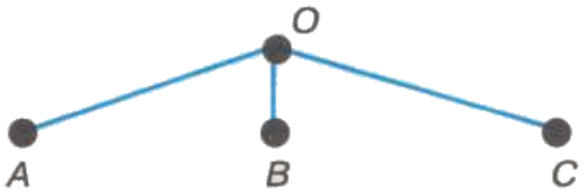


Рис. 1

Посадив на первый стул студента А, на второй стул можно посадить студента В или С. Если же на первый стул сядет В, то на второй можно посадить А или С. Если на первый стул сядет С, то на второй можно будет посадить А или В. Это соответствует на схеме двум ветвям, исходящим из каждой вершины первого уровня (рис. 2).

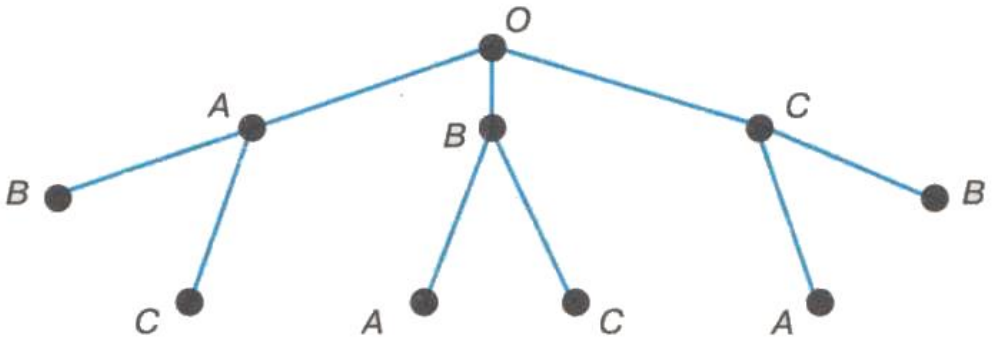


Рис. 2

Очевидно, что третий стул в каждом случае займёт оставшийся студент. Это соответствует одной ветви дерева, которая «вырастает» на каждой из предыдущих ветвей (рис. 3).

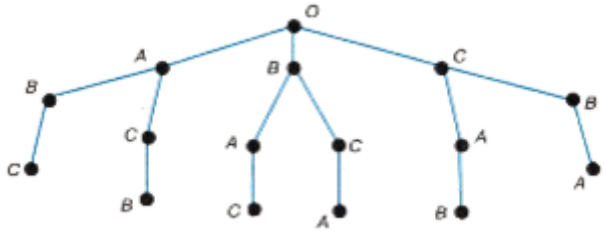


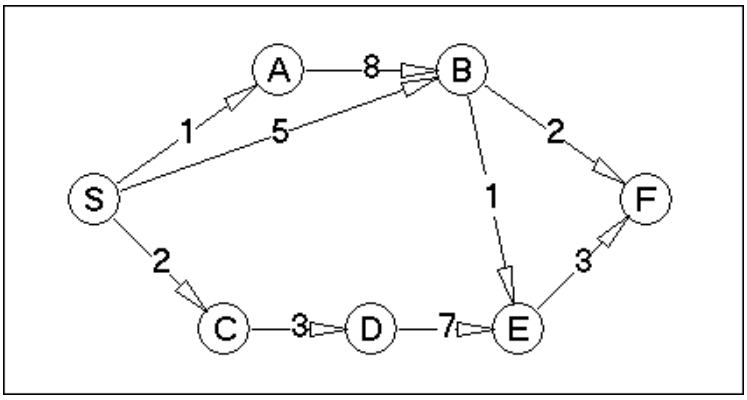
Рис. 3

Выпишем все пути от вершин первого уровня к вершинам третьего уровня: А-В-Су А-С-В, В-А-С, В-С-А, С-А-Б, С-В-А. Каждый из выписанных путей определяет один из вариантов рассаживания учеников на стулья. Так как других путей нет, то искомое число способов — 6.

Дерево можно не строить, если не требуется выписывать все возможные варианты, а нужно просто указать их число. В этом случае рассуждать нужно так: на первый стул можно усадить одного из трёх человек, на второй — одного из двух оставшихся, на третий — одного оставшегося: 3-2-1 = 6.

Пример варианта задания

Для данного графа найти наикратчайшее расстояние



Между населёнными пунктами A, B, C, D, E, F построены дороги, протяжённость которых приведена в таблице:

	A	B	C	D	E	F
A		6	4	2	1	
B	6		1			
C	4	1		3		2
D	2		3		2	
E	1			2		6
F			2		6	

Определите длину кратчайшего пути между пунктами A и F. Передвигаться можно только по дорогам, протяжённость которых указана в таблице.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Список используемых источников
5. Выводы и предложения
6. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Скажите, что такое граф?
2. Что является вершинами и ребрами графа?
3. Какой граф можно назвать взвешенным?
4. Какой граф можно назвать сематической сетью?

Тема 5.3 Этапы решения задач с помощью компьютера

Практическое занятие №23 Технология подготовки и решения задач с помощью компьютера

Цель занятия:

1. Познакомить с этапами решения задач с помощью компьютера;
2. Изучить технологию подготовки задачи к решению на компьютере;
3. Формировать ОК 01, ОК 02.

Исходные материалы: теоретический материал, компьютер.

Исходные данные:

Папка на РС «Практическое занятие №23 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы.

Теоретическая часть

Решение задач с помощью компьютера включает в себя следующие основные этапы,часть из которых осуществляется без участия компьютера.

1. Постановка задачи:
   - сбор информации о задаче;
   - формулировка условия задачи;
   - определение конечных целей решения задачи;
   - определение формы выдачи результатов;
   - описание данных (их типов, диапазонов величин, структуры и т.п.).
2. Анализ и исследование задачи, модели:
   - анализ существующих аналогов;
   - анализ технических и программных средств;
   - разработка математической модели;
   - разработка структур данных.
3. Разработка алгоритма:
   - выбор метода проектирования алгоритма;
   - выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
   - выбор тестов и метода тестирования;

• проектирование алгоритма.

4. Программирование:

• выбор языка программирования;

• уточнение способов организации данных;

• запись алгоритма на выбранном языке программирования.

5. Тестирование и отладка:

• синтаксическая отладка;

• отладка семантики и логической структуры;

• тестовые расчеты и анализ результатов тестирования;

• совершенствование программы.

6. Анализ результатов решения задачи и уточнение в случае необходимости

математической модели с повторным выполнением этапов 2 — 5.

7. Сопровождение программы:

• доработка программы для решения конкретных задач;

• составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Математическая модель — это система математических соотношений — формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта или явления. Всякое явление природы бесконечно в своей сложности.

Чтобы описать явление, необходимо выявить самые существенные его свойства,закономерности, внутренние связи, роль отдельных характеристик явления. Выделив наиболее важные факторы, можно пренебречь менее существенными. Наиболее эффективно математическую модель можно реализовать на компьютере в виде алгоритмической модели — так называемого "вычислительного эксперимента"

Итак, создавая математическую модель для решения задачи, нужно:

1. выделить предположения, на которых будет основываться математическая модель;

2. определить, что считать исходными данными и результатами;

3. записать математические соотношения, связывающие результаты с исходными данными.

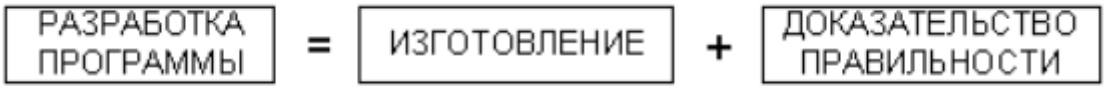
При построении математических моделей далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких

случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Существует не только математическое моделирование какого-либо явления, но и визуально-натурное моделирование, которое обеспечивается за счет отображения этих явлений средствами машинной графики, т.е. перед исследователем демонстрируется своеобразный "компьютерный мультфильм", снимаемый в реальном масштабе времени. Наглядность здесь очень высока.

Основные этапы содержит процесс разработки программ.

Процесс разработки программы можно выразить следующей формулой:



На начальном этапе работы анализируются и формулируются требования к программе, разрабатывается точное описание того, что должна делать программа и каких результатов необходимо достичь с ее помощью.

Затем программа разрабатывается с использованием той или иной технологии программирования (например, структурного программирования).

Полученный вариант программы подвергается систематическому тестированию — ведь наличие ошибок в только что разработанной программе это вполне нормальное закономерное явление. Практически невозможно составить реальную (достаточно сложную) программу без ошибок. Нельзя делать вывод, что программа правильна, лишь на том основании, что она не отвергнута машиной и выдала результаты. Все, что достигнуто в этом случае, это получение каких-то результатов, не обязательно правильных. В программе при этом может оставаться большое количество логических ошибок. Ответственные участки программы проверяются с использованием методов доказательства правильности программ.

Для каждой программы обязательно проводятся работы по обеспечению качества и эффективности программного обеспечения, анализируются и улучшаются временные характеристики.

Контроль текста программы до выхода на компьютер

Текст программы можно проконтролировать за столом с помощью просмотра, проверки и прокрутки.

• ***Просмотр.*** Текст программы просматривается на предмет обнаружения описок и расхождений с алгоритмом. Нужно просмотреть организацию всех циклов, чтобы убедиться в правильности операторов, задающих кратности циклов. Полезно посмотреть еще раз условия в условных операторах, аргументы в обращениях к подпрограммам и т.п.

• ***Проверка.*** При проверке программы программист по тексту программы мысленно старается восстановить тот вычислительный процесс, который определяет программа, после чего сверяет его с требуемым процессом. На время проверки нужно "забыть", что должна делать программа , и "узнавать" об этом по ходу её проверки. Только после окончания проверки программы можно "вспомнить" о том, что она должна делать и сравнить реальные действия программы с требуемыми.

• ***Прокрутка.*** Основой прокрутки является имитация программистом за столом выполнения программы на машине. Для выполнения прокрутки приходится задаваться какими-то исходными данными и производить над ними необходимые вычисления. Прокрутка — трудоемкий процесс, поэтому ее следует применять лишь для контроля логически сложных участков программ. Исходные данные должны выбираться такими, чтобы в прокрутку вовлекалось большинство ветвей программы.

Отладка и тестирование

Отладка программы — это процесс поиска и устранения ошибок в программе, производимый по результатам её прогона на компьютере.

Тестирование (англ. test — испытание) — это испытание, проверка правильности работы программы в целом, либо её составных частей.

Отладка и тестирование — это два четко различимых и непохожих друг на друга этапа:

• при отладке происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования;

• в процессе же тестирования проверяется работоспособность программы, не содержащей явных ошибок.

Тестирование устанавливает факт наличия ошибок, а отладка выясняет ее причину. Английский термин debugging ("отладка") буквально означает "вылавливание жучков". Термин появился в 1945 г., когда один из первых компьютеров — "Марк-1" прекратил работу из-за того, что в его электрические цепи попал мотылек и заблокировал своими останками одно из тысяч реле машины.

Отладка

В современных программных системах отладка осуществляется часто с использованием специальных программных средств,называемых отладчиками. Эти средства позволяют исследовать внутреннее поведение программы.

Программа-отладчик обычно обеспечивает следующие возможности:

• пошаговое исполнение программы с остановкой после каждой команды (оператора);

• просмотр текущего значения любой переменной или нахождение значения любого выражения, в том числе, с использованием стандартных функций; при необходимости можно установить новое значение переменной;

• установку в программе "контрольных точек", т.е. точек, в которых программа временно прекращает свое выполнение, так что можно оценить промежуточные результаты, и др.

При отладке программ важно помнить следующее:

• в начале процесса отладки надо использовать простые тестовые данные;

• возникающие затруднения следует четко разделять и устранять строго поочередно;

• не нужно считать причиной ошибок машину, так как современные машины и трансляторы обладают чрезвычайно высокой надежностью.

Тест и тестирование.

Как бы ни была тщательно отлажена программа, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на системе тестов.

Программу условно можно считать правильной, если её запуск для выбранной системы тестовых исходных данных во всех случаях дает правильные результаты.

Но, как справедливо указывал известный теоретик программирования Э. Дейкстра, тестирование может показать лишь наличие ошибок, но не их отсутствие. Нередки случаи, когда новые входные данные вызывают "отказ" или получение неверных результатов работы программы, которая считалась полностью отлаженной.

Для реализации метода тестов должны быть изготовлены или заранее известны эталонные результаты. Вычислять эталонные результаты нужно обязательно до, а не после получения машинных результатов. В противном случае имеется опасность невольной подгонки вычисляемых значений под желаемые, полученные ранее на машине.

Тестовые данные

Тестовые данные должны обеспечить проверку всех возможных условий возникновения ошибок:

- должна быть испытана каждая ветвь алгоритма;
- очередной тестовый прогон должен контролировать нечто такое, что еще не было проверено на предыдущих прогонах;
- первый тест должен быть максимально прост, чтобы проверить, работает ли программа вообще;
- арифметические операции в тестах должны предельно упрощаться для уменьшения объема вычислений;
- количества элементов последовательностей, точность для итерационных вычислений, количество проходов цикла в тестовых примерах должны задаваться из соображений сокращения объема вычислений;
- минимизация вычислений не должна снижать надежности контроля;

•тестирование должно быть целенаправленным и систематизированным, так как случайный выбор исходных данных привел бы к трудностям в определении ручным способом ожидаемых результатов; кроме того, при случайном выборе тестовых данных могут оказаться непроверенными многие ситуации;

усложнение тестовых данных должно происходить постепенно.

Этапов процесс тестирования

Процесс тестирования можно разделить на три этапа.

1. Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.

2. Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий — это граничные объемы данных, когда массивы состоят из слишком малого или слишком большого числа элементов.

3. Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений. Известно, что все программы разрабатываются в расчете на обработку какого-то ограниченного набора данных. Поэтому важно получить ответ на следующие вопросы:

— что произойдет, если программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?

— как будет вести себя программа, работающая с массивами, если количество их элементов певысит величину, указанную в объявлении массива?

— что произойдет, если числа будут слишком малыми или слишком большими?

Наихудшая ситуация складывается тогда, когда программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат.

Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно.

Характерные ошибки программирования

Ошибки могут быть допущены на всех этапах решения задачи — от ее постановки до оформления. Разновидности ошибок и соответствующие примеры приведены в таблице:

Вид ошибки	Пример
Неправильная постановка задачи	Правильное решение неверно сформулированной задачи
Неверный алгоритм	Выбор алгоритма, приводящего к неточному или эффективному решению задачи
Ошибка анализа	Неполный учет ситуаций, которые могут возникнуть; логические ошибки
Семантические ошибки	Непонимание порядка выполнения оператора
Синтаксические ошибки	Нарушение правил, определяемых языком программирования
Ошибки при выполнении операций	Слишком большое число, деление на ноль, извлечение квадратного корня из отрицательного числа и т. п.
Ошибки в данных	Неудачное определение возможного диапазона изменения данных
Опечатки	Перепутаны близкие по написанию символы, например, цифра 1 и буквы *I, l*
Ошибки ввода-вывода	Неверное считывание входных данных, неверное задание форматов данных

Примеры синтаксических ошибок:

Обычно синтаксические ошибки выявляются на этапе трансляции. Многие же другие ошибки транслятору выявить невозможно, так как транслятору неизвестны замыслы программиста.

Отсутствие сообщений машины о синтаксических ошибках является необходимым, но не достаточным условием, чтобы считать программу правильной.

- пропуск знака пунктуации;
- несогласованность скобок;

• неправильное формирование оператора;

• неверное образование имен переменных;

• неверное написание служебных слов;

• отсутствие условий окончания цикла;

• отсутствие описания массива и т.п.

Ошибки не обнаруживаемые транслятором

Существует множество ошибок, которые транслятор выявить не в состоянии, если используемые в программе операторы сформированы верно. Приведем примеры таких ошибок.

Логические ошибки:

• неверное указание ветви алгоритма после проверки некоторого условия;

• неполный учет возможных условий;

• пропуск в программе одного или более блоков алгоритма.

Ошибки в циклах :

• неправильное указание начала цикла;

• неправильное указание условий окончания цикла;

• неправильное указание числа повторений цикла;

• бесконечный цикл.

Ошибки ввода-вывода; ошибки при работе с данными:

• неправильное задание тип данных;

• организация считывания меньшего или большего объёма даных, чем требуется;

• неправильное редактирование данных.

Ошибки в использовании переменных:

• использование переменных без указания их начальных значений;

• ошибочное указание одной переменной вместо другой.

Ошибки при работе с массивами:

• массивы предварительно не обнулены;

• массивы неправильно описаны;

• индексы следуют в неправильном порядке.

Ошибки в арифметических операциях:

• неверное указание типа переменной (например, целочисленного вместо вещественного);

• неверное определение порядка действий;

• деление на нуль;

• извлечение квадратного корня из отрицательного числа;

• потеря значащих разрядов числа.

Все эти ошибки обнаруживаются с помощью тестирования.

Сопровождение программы

Сопровождение программ — это работы, связанные с обслуживанием программ в процессе их эксплуатации.

Многократное использование разработанной программы для решения различных задач заданного класса требует проведения следующих дополнительных работ:

• исправление обнаруженных ошибок;

•модификация программы для удовлетворения изменяющихся эксплуатационных требований;

• доработка программы для решения конкретных задач;

• проведение дополнительных тестовых просчетов;

• внесение исправлений в рабочую документацию;

• усовершенствование программы и т.д.

Применительно ко многим программам работы по сопровождению поглощают более половины затрат, приходящихся на весь период времени существования программы (начиная от выработки первоначальной концепции и кончая моральным ее устареванием) в стоимостном выражении.

Программа, предназначенная для длительной эксплуатации, должна иметь соответствующую документацию и инструкцию по её использованию.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;

2. Цель занятия;

3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

4. Список используемых источников;

5. Выводы и предложения;

6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки:

1. Какие основные этапы включает в себя решение задач на компьютере?
2. Какие этапы компьютерного решения задач осуществляются без участия

компьютера?

3.. Что называют математической моделью объекта или явления?

4. Какие способы моделирования осуществляются с помощью компьютера?

5. Из каких последовательных действий состоит процесс разработки программы?

6. Чем тестирование программы отличается от её отладки?

7. На какой стадии работы над программой вычисляются эталонные результаты тестов?

8. Назовите основные этапы процесса тестирования.

9. В чём заключается отличие синтаксических ошибок от семантических?

10. Какие разновидности ошибок транслятор не в состоянии обнаружить?

Практическое занятие № 24 Введение в язык программирования Python. Ввод и вывод данных. Типы данных

Цель занятия:

4. Познакомиться со средой разработки Python;
5. Уметь запускать среду программирования Python, сохранять и открывать программы в среде программирования;
6. Знать основные типы данных, команды ввода и вывода данных;
7. Формировать ОК 01, ОК 02.

Исходные материалы: теоретический материал, компьютер, среда программирования Python

Исходные данные:

Папка на РС «Практическое занятие №24 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Python — активно развивающийся язык программирования, новые версии с добавлением и изменением языковых свойств выходят примерно раз в два с половиной года. Он находит применение во множестве сфер человеческой деятельности.

В комплекте вместе с интерпретатором Python идет IDLE (интегрированная среда разработки). По своей сути она подобна интерпретатору, запущенному в интерактивном режиме с расширенным набором возможностей (подсветка синтаксиса, просмотр объектов, отладка и т.п.).

Скачать и установить интерпретатор Python можно совершенно бесплатно с официального сайта: http://python.org. Для работы нам понадобится интерпретатор **Python версии 3** или выше.

Для запуска **IDLE** в Windows необходимо перейти в папку **Python** в меню **“Пуск”** и найти там ярлык с именем “**IDLE** (Python 3.X XX-bit)”.

После установки программы запустите интерактивную графическую среду IDLE и дождитесь появления приглашения для ввода команд:

Type "copyright", "credits" or "license ()" for more information.

>>>

Три знака «больше» (>>>) называются *приглашением*. После приглашения можно вводить различные команды.

В интерактивном режиме IDLE можно с клавиатуры вводить математические выражения. После завершения набора выражения нажимается клавиша **Enter** для завершения ввода и вывода результата на экран. Такой режим работы Python называют интерактивным калькулятором.

Пример:

>>>1-5
- 4
>>> _+6 **Нижним подчеркиванием обозначается последний полученный результат.**
2

Любая Python-программа состоит из последовательности допустимых символов, записанных в определенном порядке и по определенным правилам.

Алфавит языка Python (набор допустимых символов) состоит из букв латинского алфавита (причём *заглавные и строчные буквы различаются*), цифр и специальных знаков (знаков препинания, арифметических и других). Русские буквы могут использоваться только при выводе текста на экран и в комментариях к программе.

Программа на языке Python включает в себя:

1)**комментарии**

обозначаются предваряющим их символом # и продолжаются до конца строки

2)**команды**

каждая команда пишется с новой строки или разделяется в одной строке ;

3)**знаки пунктуации**

в алфавит Python входит достаточное количество знаков пунктуации, которые используются для различных целей. Например, знаки "+" или "" могут использоваться для сложения и умножения, а знак запятой "," - для разделения параметров функций*

4)**идентификаторы**

идентификаторы в Python - это имена используемые для обозначения переменной, функции, класса, модуля или другого объекта.

5)**ключевые слова**

некоторые слова имеют в Python специальное назначение и представляют собой управляющие конструкции языка, ключевые слова, которые для Python имеют определенный смысл.

Величины в программе представлены в виде констант и переменных.

Константы – величины, не изменяющие своего значения при выполнении программы.

Переменные – величины, которые могут изменять свое значение при выполнении программы. Каждая переменная имеет имя, тип и значение.

Имя переменной (идентификатор) – любая отличная от служебных слов последовательность латинских букв, цифр и символа подчеркивания "_", не может начинаться с цифры.

N, N1, massa, massa_tela – правильно;

1N, масса, massa tela – неправильно.

Имена переменным придумывает программист, но есть несколько ограничений, связанных с наименованием. В качестве имен переменных нельзя использовать ключевые слова, которые для Python имеют определенный смысл (эти слова подсвечиваются в IDLE оранжевым цветом):

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

Значения переменных хранятся в ячейках оперативной памяти.

Тип переменной определяет способ хранения данных в памяти компьютера и допустимые операции над ними.

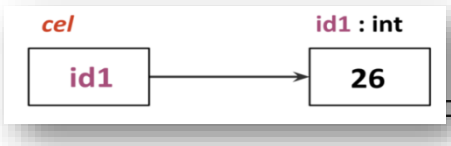
Типы данных

Название	Обозначение	Допустимые значения
Целочисленный	int («integer»)	Сколь угодно большие целые числа, размер ограничен оперативной памятью
Вещественный	float («floating point»)	Любые числа с дробной частью (с плавающей точкой)
Строковый	str («string»)	Произвольная последовательность символов из таблицы Unicode
Логический	bool («boolean»)	False («Ложь») или True («Истина»)

Команда присваивания

Знак = сопоставим в программировании с командой присваивания.

В момент выполнения присваивания




```
>>> cel = 26
>>>cel = 26 + 46
>>> cel
72
>>>
```

держат адреса объектов или можно сказать, что переменные ссылаются на объекты. Постоянно сохраняя в голове эту модель, для упрощения будем говорить, что переменная содержит значение.

В языке Python допускается множественное присваивание:

Запись оператора:	*Равносильная запись:*
a, b = 0, 1	a = 0 b = 1
a = b = 0	a = 0 b = 0

В математических выражениях в качестве операндов можно использовать целые числа (1, 4, -5) или вещественные (в программировании их еще называют числами с плавающей точкой): 4.111, -9.3. Если один из операндов является вещественным числом, то в результате получится вещественное число.

Простые арифметические операции над числами

оператор	описание
x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление

//	Деление с округлением вниз
**	Возведение в степень
%	Остаток от деления
abs(x)	Модуль числа

Пример:

- **>>> 5/3**
- **1.6666666666666667**
- **>>> 5//3**
- **1**
- **>>> 5%3**
- **2**
- **>>> 5**67**
- **677626357803440271254658000543713569641113281 25**
- **>>>**

Приоритеты выполнения операций

1)операции в скобках;

2) возведение в степень;

3) умножение и деление (в том числе // и %);

4) сложение и вычитание.

Операции одинакового приоритета выполняются в порядке записи слева направо.

- >>> -2**4
- -16
- >>> -(2**4)
- -16
- >>> (-2)**4
- 16

Если выражение слишком длинное и не помещается в одной строке, необходимо заключить всё выражение в скобки (перенос внутри скобок разрешён).

Ввод и вывод данных

Для ввода значений переменных с клавиатуры в процессе выполнения программы используется функция ввода

input («ввод»):

Формат записи функции

<имя_переменной> = **input**()

В скобках функции можно указать сообщение - комментарий к вводимым данным:

```
a = input ("Введите количество: ")
```

При выполнении оператора:

- компьютер переходит в режим ожидания данных;
- пользователь вводит с клавиатуры данные в виде строки символов;
- для завершения ввода пользователь нажимает клавишу Enter;

введенная строка записывается в указанную переменную.

Если вводится не строка, а число, необходимо выполнить преобразование типов с помощью функций **int** (для целых) и **float** (для вещественных).

Например:

```
print("Введите слово и два числа:")
x = input()
y = int(input())
z = float(input())
print(x, y, z)
```

На экране:

```
Введите слово и два числа:
ноль
1
2
ноль 1 2.0
```

Можно в одной строке ввести несколько значений через пробел. Для этого используется функция **split** («расщепить»). Затем данные необходимо преобразовать к нужному типу по отдельности.

Например:

```
a, b, c = input("Введите a,b,c через пробел: ").split()
a, b, c = int(a), int(b), int(c)
print (a, b, c)
```

На экране:

```
Введите a,b,c через пробел: 1 2 3
1 2 3
```

Вывод данных из оперативной памяти на экран осуществляется с помощью функции вывода

print («печатать»):

формат записи функции

print(<выражение1>, <выражение2>, ..., <выражениеN>)

- На экран выводятся значения переменных и выражений, строковые значения выводится на экран без кавычек.
- Выводимые значения разделяются пробелом (по умолчанию).
- После выполнения функции происходит автоматический переход на новую строку.

Например:

print ("Масса равна", m, "кг");

Для m=15 на экране появится:

Масса□равна□15□кг

Здесь и далее символом □ обозначен пробел.

Вместо пробела можно использовать другие символы в качестве разделителя, указав их после слова **sep** («separator»).

Чтобы убрать переход на новую строку после выполнения оператора, используется параметр **end**

Нужный вариант вывода	**Оператор**	**На экране**
По умолчанию	print (1, 20, 300)	1□20□300
Без разделителя	print (1, 20, 300, sep="")	120300
Через запятую и пробел	print (1, 20, 300, sep=", ")	1,□20,□300
Каждое с новой строки	print (1, 20, 300, sep="\n")	1 20 300
Без перехода на новую строку	print (1, end="") print (20)	120

Фрагмент программы	**На экране**

print ("{:3}{:3}{:3}".format(13, 7, 22))	□13□□7□22
a = 7 print ("{:4d}{:4d}".format(a, a*a))	□□□7□□49
a = 1/3; b = 1/9 print ("{:7.3f}{:7.4f}".format(a, b))	□□0.333□0.1111
a = 1/3 print ("{:10.3e}".format(a))	□3.333e-01

Пример

Напишите программу, которая запрашивала бы у пользователя:

Вариант 0 **Решение**

- ФИО ("Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваше имя"

"Ваш возраст"

"Вы живете в"

```
a=input('Введите ваши фамилию, имя, отчество ')
b=input('Сколько вам лет? ')
c=input('Где вы живёте? ')
print('Ваше имя ',a)
print('Ваш возраст ',b)
print('Вы живете в ',c)
```

```
Введите ваши фамилию, имя, отчество Иванов Иван Иванович
Сколько вам лет? 15
Где вы живёте? Уссурийск
Ваше имя  Иванов Иван Иванович
Ваш возраст  15
Вы живете в  Уссурийск
```

Пример варианта задания

1) В режиме интерактивного калькулятора найти значение выражения

$$\left(5{,}1\cdot 10^{-3}\right)^2-\left(\frac{1}{7}\right)^2+\sqrt{5}.$$

2) Напишите программу, которая запрашивала бы у пользователя:

Имя, Фамилия, Возраст, Место жительства

- фамилия, имя ("Ваши фамилия, имя?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваши фамилия, имя"

"Ваш возраст"

"Вы живете в"

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;

2. Цель занятия;

3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

4. Список используемых источников;

5. Выводы и предложения;

6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1.Как в языке Python называются указания компьютеру, определяющие, какие операции выполнит компьютер над данными?

2.Определите порядок выполнения операций в указанной инструкции?

```
a = 3 - 5 * 4 ** (-3 + 2)
```

3. Символ # в Python обозначает **…**

4.345, 6.8, КМРК - **…** типы данных.

5.Функция input() – предназначена для …

Практическое занятие № 25 Оператор присваивания. Математические операции с целыми и вещественными числами.

Цель занятия:

1. Изучить основные математические операциями в Python;
2. Уметь писать коды программ для решения математических задач;
3. Формировать ОК 01, ОК 02.

Исходные материалы: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №25 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Язык Python, благодаря наличию огромного количества библиотек для решения разного рода вычислительных задач, сегодня является конкурентом таким пакетам как Matlab и Octave. Запущенный в интерактивном режиме, он, фактически, превращается в мощный калькулятор.

Задания для самостоятельной работы

Задание №1

Рассчитайте на калькуляторе Python

$3{,}95+\left(\frac{1}{20}\right)^2+\frac{1}{900}$, *Ответ*: 3,95.....

$\frac{2{,}75\cdot 10^2}{\sqrt{14{,}3+\pi}}$, *Ответ*: 65,842...

$6{,}7\left(\frac{1}{\pi+5{,}04}+10^{0{,}198\sqrt{3{,}2}}\right)-0{,}36\cdot 10^{-3}$, *Ответ*: 15,96

$\left(\sqrt{\frac{0{,}079}{3{,}5}-6{,}1\cdot 10^{-5}}+0{,}4\right)\cdot\left(e^{\frac{0{,}7}{9}+0{,}8}-1{,}5\right)$, *Ответ*: 0,49...

Ответы даны для того, чтобы вы убедились в правильности ваших действий в системе программирования.

В этом практическом занятии речь пойдет об арифметических операциях, доступных в данном языке. Арифметические операции изучим применительно к числам.

Если в качестве операндов некоторого арифметического выражения используются только целые числа, то результат тоже будет целое число. Исключением является операция деления, результатом которой является вещественное число. При совместном использовании целочисленных и вещественных переменных, результат будет вещественным.

Целые числа (int)

Числа в Python 3 поддерживают набор самых обычных математических операций:

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)
x ** y	Возведение в степень
pow(x, y[, z])	x : Число, которое требуется возвести в степень. y : Число, являющееся степенью, в которую нужно возвести первый аргумент. Если число отрицательное или одно из чисел "x" или "y" не целые, то аргумент "z" не принимается. z : Число, на которое требуется произвести деление по модулю. Если число указано, ожидается, что "x" и "y" положительны и имеют тип int.

Пример применения выше описанных операций над целыми числами

```
x = 5
y = 2
z = 3
x+y = 7
x-y = 3
x*y = 10
x/y = 2.5
x//y = 2
x%y = 1
-x= -5
abs(-x) = 5
divmod(x,y) = (2, 1)
x**y = 25
pow(x,y,z) = 1
```

Вещественные числа (float)

Вещественные числа поддерживают те же операции, что и целые. Однако (из-за представления чисел в компьютере) вещественные числа неточны, и это может привести к ошибкам.

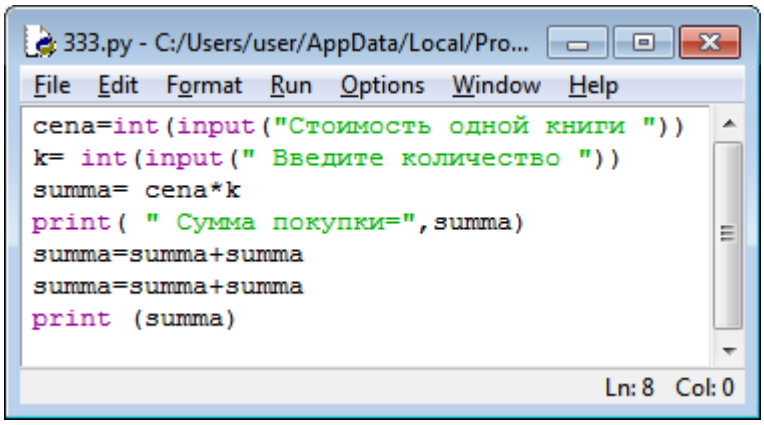
Пример применения вышеописанных операций над вещественными числами

```
x = 5.5
y = 2.3
x+y = 7.8
x-y = 3.2
x*y = 12.649999999999999
x/y = 2.3913043478260874
x//y = 2.0
x%y = 0.9000000000000004
-x= -5.5
abs(-x) = 5.5
divmod(x,y) = (2.0, 0.9000000000000004)
x**y = 50.44686540422945
```

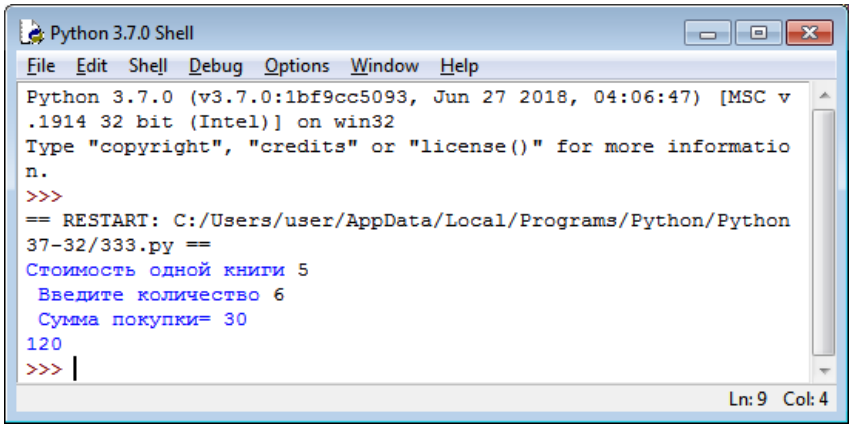
Пример составления программ для решения математической задачи

Задается стоимость одной книги и количество покупаемых книг. Написать программу вычисления стоимости покупки с выводом на экран.

Дописать в программу операторы присваивания, которые позволили бы просуммировать несколько значений стоимости покупки



Решение



Примерный вариант задания.

1. Составить программу, меняющую местами значения двух переменных
2. Составить программу для вычисления площади треугольника по известным длинам его сторон.

Формула Герона: $$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$$

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;
2. Цель занятия;
3. Вариант задания;
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

5. Список используемых источников;
6. Выводы и предложения;
7. Дата и подпись курсанта и преподавателя;

Вопросы для самопроверки

1 Перечислите математические операции для целых чисел.

2 Операция 46%10 – это

3 Операция 3**4 - это

4 Что получается в результате выполнения операции divmod(x, y)

5 Дан код программы:

```
x=1
x=4
print x,x
```

Что в результате работы программы появится на экране?

6 Как надо изменить программу выше, чтобы первое число не терялось?

7 Возможно ли в математике такое равенство?

```
x=x+1
```

Какой смысл оператора x=x+1 в программировании?

Практическое занятие № 26 Стандартные функции. Математический модуль math.

Цель занятия:

1. Изучить стандартные функции в Python;
2. Познакомиться с функциями математического модуля math;
3. Уметь писать коды программ для расчета математических функций;
4. Формировать ОК 01, ОК 02.

Исходные материалы: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №26 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Большинство стандартных функций языка Python разбиты на группы по назначению, каждая группа записана в отдельном файле, который называется модулем.

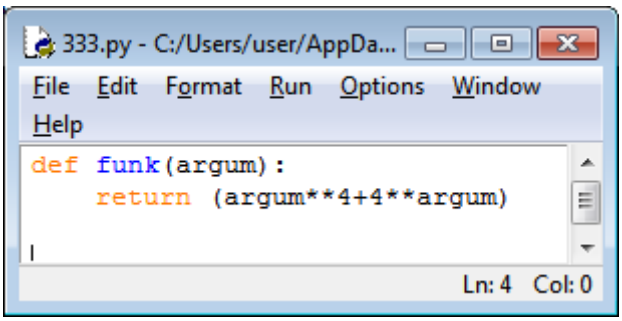
К примеру, вы написали несколько полезных функций, которые часто используете в своих программах. Чтобы к ним быстро обращаться, удобно все эти функции поместить в отдельный файл и загружать их оттуда.

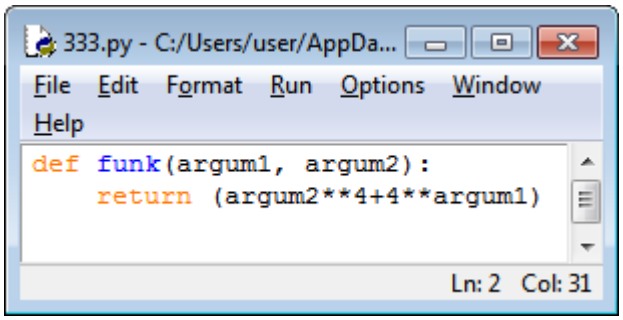
Пример

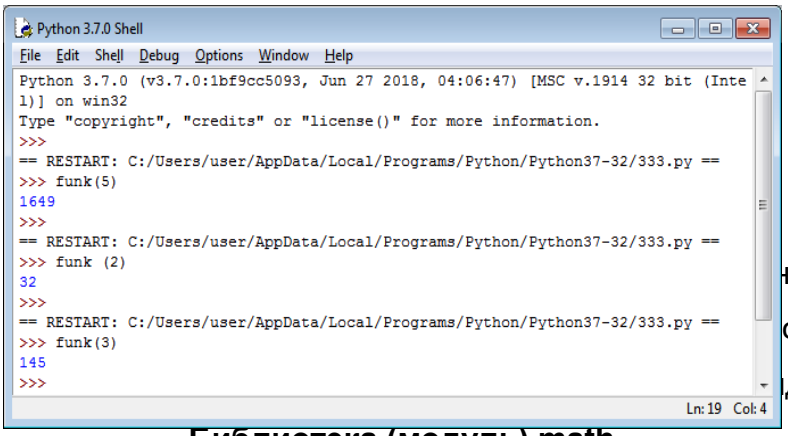
1. Создайте собственные функции для вычисления следующих выражений

$x^4 + 4^x$ $y^4 + 4^x$

Решение





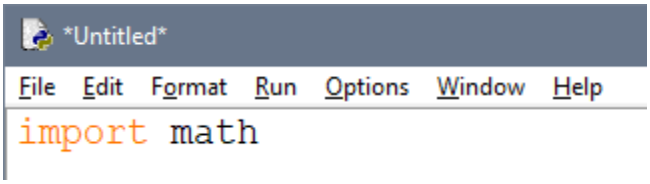


нкций и на дулями. Для орые нахо модуле, его ды import:

Библиотека (модуль) math

В стандартную поставку Python входит библиотека math, в которой содержится большое количество часто используемых математических функций.

Для работы с данным модулем его предварительно нужно импортировать.




```
        # подключаем все функции из модуля math
                from math import *
```

```
from math import *

>>> math.sqrt(9)
 3.0

>>>
```

Рассмотрим наиболее часто используемые функции модуля math

math.ceil(x)	Возвращает ближайшее целое число большее, чем x
math.fabs(x)	Возвращает абсолютное значение числа x
math.factorial(x)	Вычисляет факториал x
math.floor(x)	Возвращает ближайшее целое число меньшее, чем x
math.exp(x)	Вычисляет e**x
math.log2(x)	Логарифм по основанию 2
math.log10(x)	Логарифм по основанию 10
math.log(x[, base])	По умолчанию вычисляет логарифм по основанию e, дополнительно можно указать основание логарифма
math.pow(x, y)	Вычисляет значение x в степени y
math.sqrt(x)	Корень квадратный от x

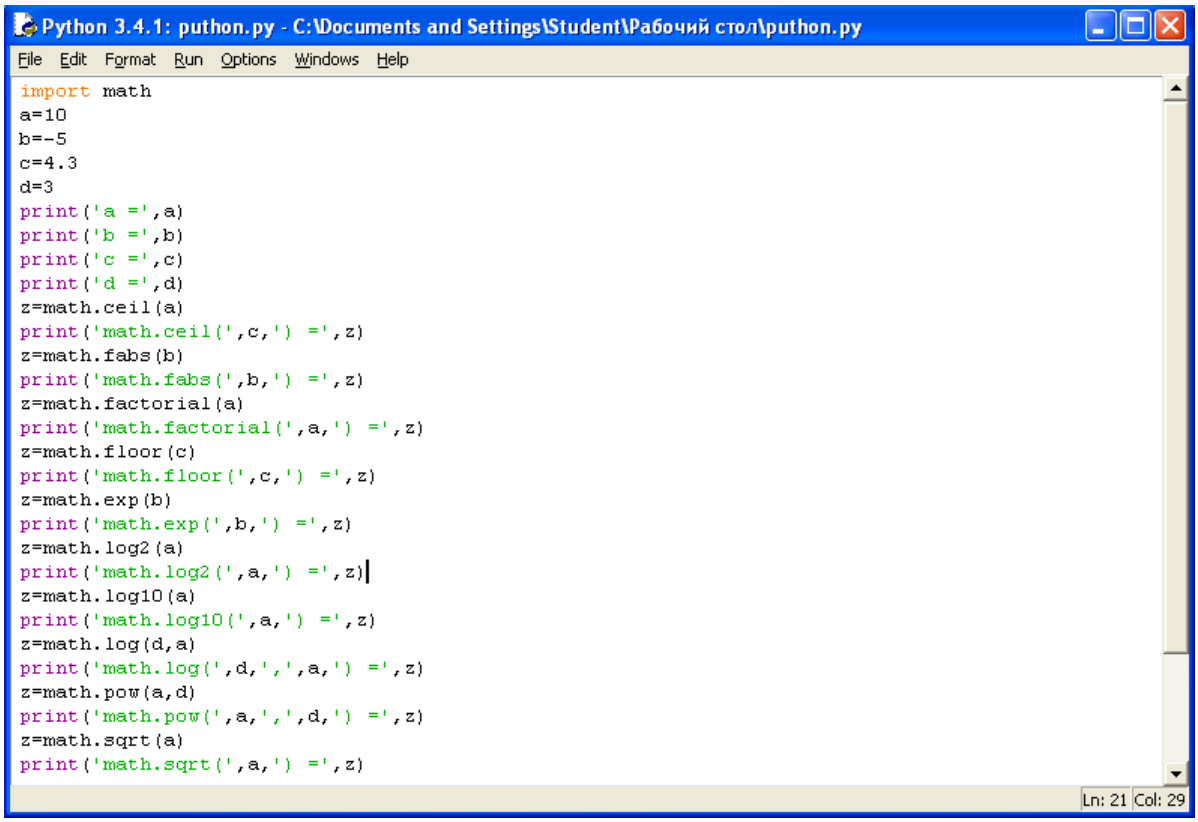
Пример применения вышеописанных функций над числами

В программе определены 4 переменные - a, b, c, d, каждая из которых является либо целым числом, либо вещественным, либо отрицательным.

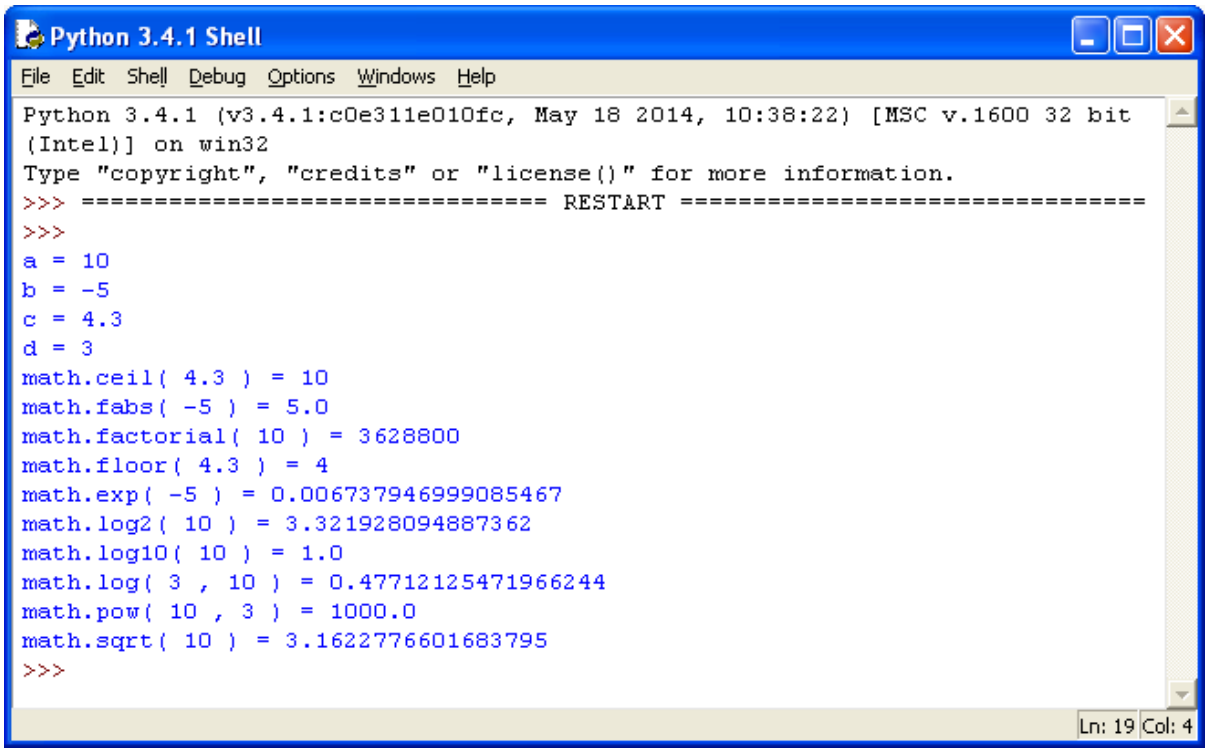
Командой print() выводится значение каждой переменной на экран при выполнении программы.

В переменную z помещается результат выполнения функции модуля math.

Затем командой print() выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.



Пример программы на Python



Результат выполнения программы с применением функций модуля math

Тригонометрические функции модуля math

math.cos(x)	Возвращает cos числа X
math.sin(x)	Возвращает sin числа X
math.tan(x)	Возвращает tan числа X
math.acos(x)	Возвращает acos числа X
math.asin(x)	Возвращает asin числа X
math.atan(x)	Возвращает atan числа X

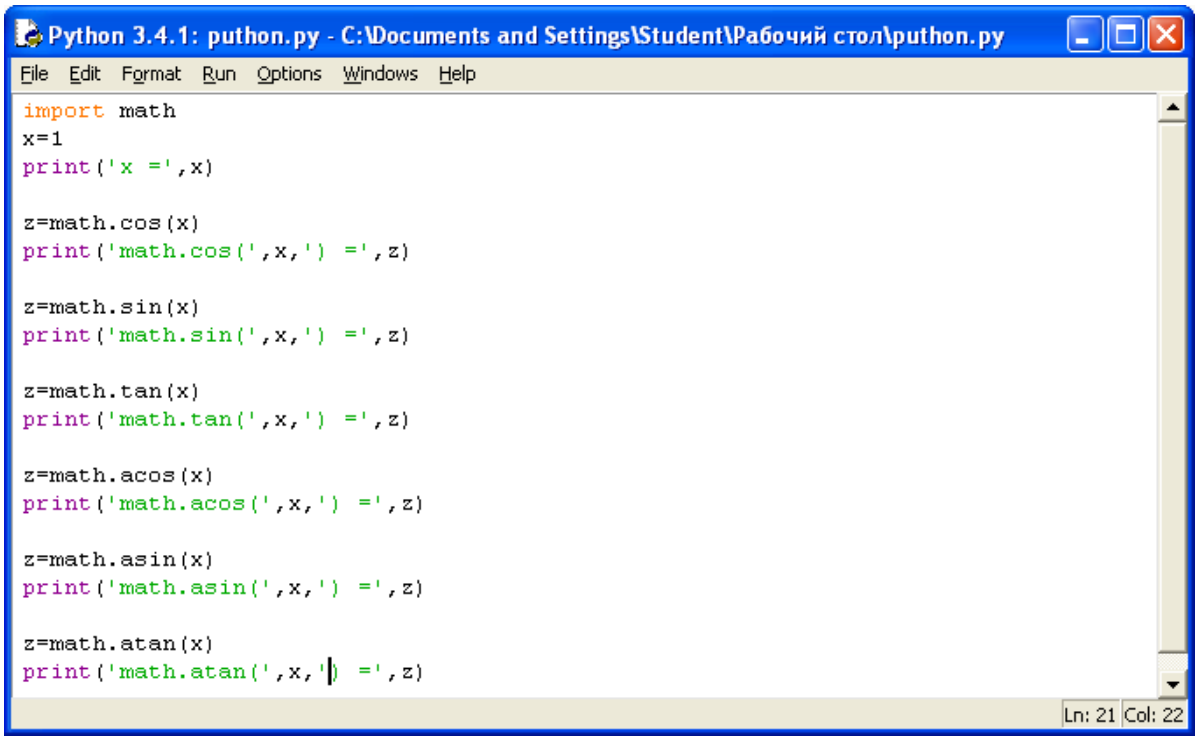
Пример применения вышеописанных функций над числами

В программе определена переменная x, содержащая целое число.

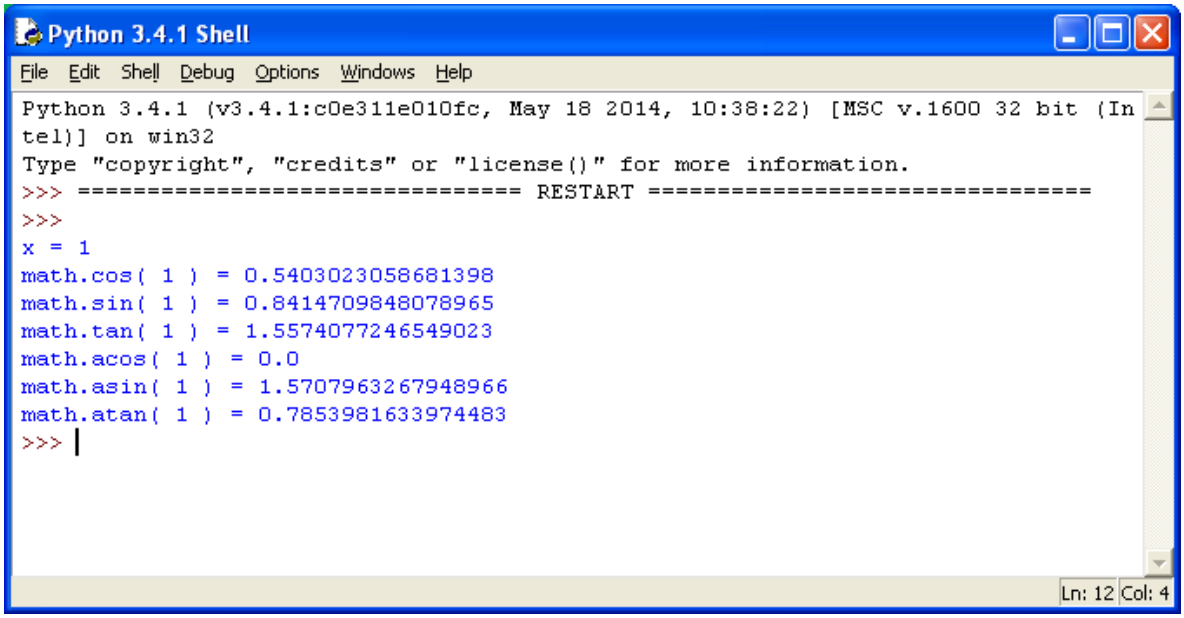
Значение переменной выводится командой print() на экран.

В переменную z помещается результат выполнения тригонометрической функции модуля math.

Затем командой print() выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.



Пример программы с использованием тригонометрических функций модуля math



Результат выполнения программы с применением тригонометрических функций модуля math

Константы:

- **math.pi** - число Pi.
- **math.e** - число e (экспонента).

Пример

Напишите программу, которая бы вычисляла заданное арифметическое выражение при заданных переменных. Ввод переменных осуществляется с клавиатуры. Вывести результат с 2-мя знаками после запятой.

Пример варианта задания

$$Z = \frac{9\pi t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^{x}$$

x=10; t=1

Решение

Сначала импортируем модуль math. Для этого воспользуемся командой importmath.

Затем следует ввести значения двух переменных целого типа x и t.

Для ввода данных используется команда input, но так как в условии даны целые числа, то нужно сначала определить тип переменных: x=int(), t=int().

Определив тип переменных, следует их ввести, для этого в скобках команды int() нужно написать команду input().

Для переменной x это выглядит так: x=int(input("сообщение при вводе значения")).

Для переменной t аналогично: t=int(input("сообщение при вводе значения")).

Следующий шаг - это составление арифметического выражения, результат которого поместим в переменную z.

Сначала составим числитель. Выглядеть он будет так: 9*math.pi*t+10*math.cos(x).

Затем нужно составить знаменатель, при этом обратим внимание на то, что числитель делится на знаменатель, поэтому и числитель, и знаменатель нужно поместить в скобки (), а между ними написать знак деления /.

Выглядеть это будет так: (9*math.pi*t+10*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t))).

Последним шагом является умножение дроби на экспоненту в степени x.

Так как умножается вся дробь, то следует составленное выражение поместить в скобки (), а уже потом написать функцию math.pow(math.e,x).

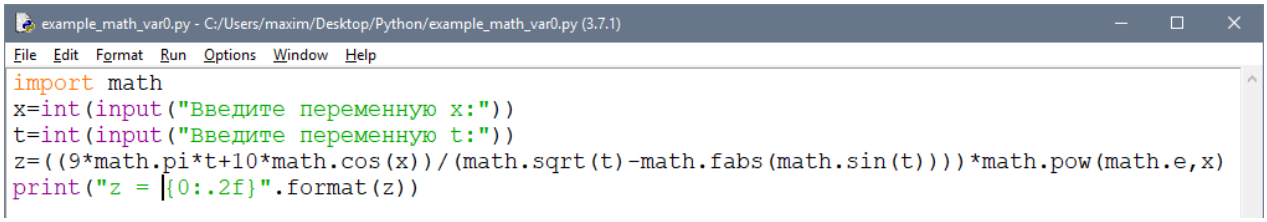
В результате выражение будет иметь вид:
z=((9*math.pi*t+10*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t))))*math.pow(math.e,x).

При составлении данного выражения следует обратить внимание на количество открывающихся и закрывающихся скобок.

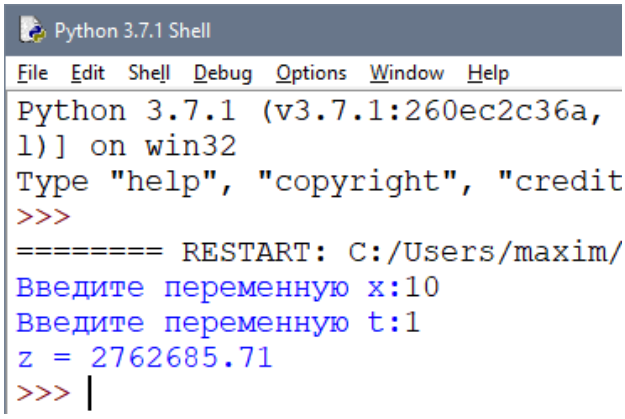
Командой print() выведем значение переменной, отформатировав его командой format.

Сам формат записывается в апострофах в фигурных скобках {}.

В задаче требуется вывести число с двумя знаками после запятой, значит вид формата будет выглядеть следующим образом: {0:.2f}, где 2 - это количество знаков после запятой, а f указывает на то, что форматируется вещественное число. При этом перед 2 нужно поставить точку, указав тем самым на то, что форматируем именно дробную часть числа.



Результат



Выводы и предложения о проделанной работе

Содержание отчета:

8 Наименование практического занятия;

9 Цель занятия;

10 Вариант задания;

11 Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

12 Список используемых источников;

13 Выводы и предложения;

14 Дата и подпись курсанта и преподавателя;

Вопросы для самопроверки

1 Перечислите математические операции для целых чисел.

2 Перечислите часто используемые функции модуля math.

3 Перечислите тригонометрические функции модуля math.

4 Как создать свою собственную функцию?

Практическое занятие № 27 Понятие логических выражений и операций. Таблица истинности.

Цель занятия:

1. Познакомиться с логическими выражениями и операциями;

2. Освоить комбинацию логических высказываний;

3. Уметь представлять в виде таблицы истинности комбинации логических высказываний.

4. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №27 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Логическое выражение — конструкция языка программирования, результатом вычисления которой является «истина» или «ложь».

Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо логическое True.

Числовые и строковые величины можно сравнивать. В Python для этого есть следующие операции сравнения:

>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Не равно

В интерактивном режиме можно сравнивать числа:

```
>>> 6>5
True
>>> 7<1
False
>>> 7==7   # не путайте == и =
True
>>> 7 != 7
False
```

Python возвращает
True (Истина == 1), когда сравнение верное
False (Ложь == 0) , когда сравнение неверное
True и False относятся к логическому (булевому) типу данных **bool**.

Логическим высказыванием (предикатом- это понятие можно посмотреть здесь http://book.kbsu.ru/theory/chapter5/1_5_1.html) будем называть любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Например, высказывание: «6 — четное число». Истинно или ложно? Очевидно, что истинно. А высказывание: «6 больше 19» Высказывание ложно.

Высказывания можно комбинировать. Высказывания «Петров – врач», «Петров – шахматист» можно объединять с помощью связок И, ИЛИ.

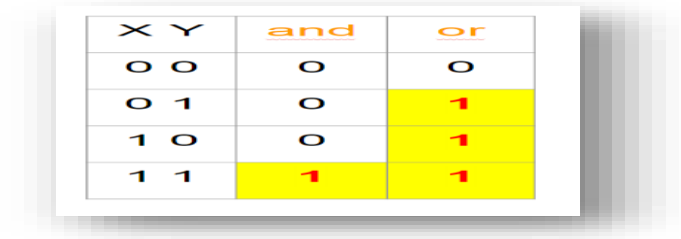
«Петров – врач И шахматист». Это высказывание истинно, если ОБА высказывания «Петров – врач» И «Петров – шахматист» являются истинными.

«Петров – врач ИЛИ шахматист». Это высказывание истинно, если истинным является ОДНО ИЗ высказываний «Петров – врач» ИЛИ «Петров – шахматист».

Как это используется в Python? Рассмотрим пример комбинаций из высказываний:

```
>>> 2>4
False
>>> 45>3
True
>>> 2>4 and 45>3 # комбинация False and (И) True вернет False False
False
>>> 2>4 or 45>3# комбинация False or (ИЛИ) True вернет True
 True
```

Все, что мы сказали про комбинацию логических высказываний, можно объединить и представить в виде таблицы истинности, где 0 – False, а 1 – True.



X	Y	and	or
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Для Python истинным или ложным может быть не только логическое высказывание, но и объект. Так, что же такое истина в Python?

В Python любое число, не равное нулю, или непустой объект интерпретируется как истина.

Числа, равные нулю, пустые объекты и специальный объект None интерпретируются как ложь.

Рассмотрим пример:

```
>>> ‘ ' and 2
« «      # False and True
>>> ‘ ' or 2
2        # False or True
>>>
```

Мы выполнили логическую операцию and (И) для двух объектов: пустого строкового объекта (он будет ложным) и ненулевого числового объекта (он будет истинным). В итоге Python вернул нам пустой строковый объект. В чем тут дело? Затем мы выполнили аналогично операцию or (ИЛИ). В результате получили числовой объект. Будем разбираться.

У Python есть три логических оператора and, or, not. not из них самый простой:

```
>>> y = 6>8
>>> y
False
>>> not y
True
>>> not None
 True
>>> not 2
 False
```

Результатом применения логического оператора not (НЕ) произойдет отрицание операнда, т.е. если операнд истинный, то not вернет – ложь, если ложный, то – истину.

Логический оператор and (И) вернет True (истину) или False (ложь), если его операндами являются логические высказывания.

Если операндами оператора and являются объекты, то в результате Python вернет объект:

Посмотрим на столбец and таблицы истинности. Какая закономерность? Если среди операндов (X,Y) есть ложный, то получим ложное значение, но вместо ложного значения для операндов-объектов Python возвращает первый ложный операнд, встретившийся в выражении, и дальше вычисления НЕ производит. Это называется вычислением по короткой схеме. Если Python не удается найти ложный объект-операнд, то он возвращает крайний правый операнд.

```
>>> 0 and 3 # вернет первый ложный объект-операнд
0
>>> 5 and 4 # вернет крайний правый объект- операнд
4
>>>
```

X	Y	and	or
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Логические выражения можно комбинировать:

```
>>> 1+3 > 7      # приоритет + выше, чем >
False
>>> (1+3) > 7 # скобки способствуют наглядности и избавляют от ошибок
False
>>> 1+(3>7)      # подумайте, что вернет выражение и почему
1
>>>
```

В Python можно проверять принадлежность интервалу:

```
>>> x=0
>>> -5< x <10 # эквивалентно: x > -5 and x<10
True
>>>
```

Строки в Python тоже можно сравнивать по аналогии с числами. Начнем издалека. Символы, как и все остальное, представлено в компьютере в виде чисел. Есть специальная таблица, которая ставит в соответствие каждому символу некоторое число. Определить, какое число соответствует символу можно с помощью функции ord():

```
>>> ord ('L')
76
>>> ord ('Ф')
1060
>>> ord ('A')
65
>>>
```

Для сравнения строк Python их сравнивает посимвольно:

Теперь сравнение символов сводится к сравнению чисел, которые им соответствуют:

```
 >>> 'Aa' > 'Ll'
False
>>>
```

Следующий полезный оператор, с которым мы познакомимся – in. Он проверяет наличие подстроки в строке:

```
>>> 'a' in 'abc'
True
>>> 'A' in 'abc'          # большой буквы A нет
 False
>>> "" in 'abc'           # пустая строка есть в любой строке
 True
>>> '' in ''
True
>>>
```

Освоив логические операции, перейдем к их использованию в следующем практическом занятии.

Пример варианта задания

1)Решение в интерактивном режим

1. В чем разница между выражениями: True == 1 и True is 1? Какие результаты получим при их вычислении?

Решение

```
>>> True == 1
True
>>> True is 1 # Сравниваются ячейки памяти
False
>>> id(True), id(1)
(2078150976, 2078275504) # Как видим – разные адреса в памяти
```

2)

Если сравнивать 2 логических объекта оператором or, то какие значения могут быть получены в итоге? Проиллюстрируйте ответ.

Решение:

```
>>> True or True
True
```

>>> False or True
True
>>> False or False
False
>>> True or False
True

3) После изучения темы стала задача написать функцию divider(a, b), принимающую любые 2 числовых параметра.

Задача функции: разделить a на b.

Если в знаменателе введут ноль, то результат будет следующим: «Нули в знаменателе не приветствуются».

В противном случае выводится итог деления чисел, возведенный в куб.

Решите задание применяя свойства логических операторов.

Для решения задачи необходимо вспомнить о свойствах операторов and, or.

Если решать задание «в лоб», то оно очень простое.

Решение

```
def divider(a, b):
    return b and (a / b) ** 3 or 'Нули в знаменателе не приветствуются'

# Тесты
print(divider(10, 4))
print(divider(10, 0))
print(divider(-12.2, 2))
print(divider(-6.4, 0))
```

Результат

15.625
Нули в знаменателе не приветствуются
-226.98099999999997
Нули в знаменателе не приветствуются

4) Теперь вы без труда сможете разобраться в работе следующего кода:

```
>>> x = 5 < 10   # True
>>> y = 2 > 3    # False
>>> x or y
True
>>> (x or y) + 6
7
>>>
```

Почему так?

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;

2. Цель занятия;

3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

4. Список используемых источников;

5. Выводы и предложения;

6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Какие знаки используются в Питоне для сравнения?

2. Какие величины можно сравнивать?.

3. Дайте определение понятию «логическое выражение»

4. Что из себя представляет таблица истинности логических выскываний?

5. С помощью каких связок можно комбинировать высказывания?.

Практическое занятие № 28 Проверка условий в Python. Синтаксис If,If-else,if-elif-else. Составление программ с проверкой условий.

Цель занятия:

1. Познакомиться со структурой ветвление (if, if-else, if-elif-else);

2. Научиться работать с числами и строками используя структуру ветвление;

3. Уметь писать коды программ с использованием структуры ветвления;

4. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №28 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

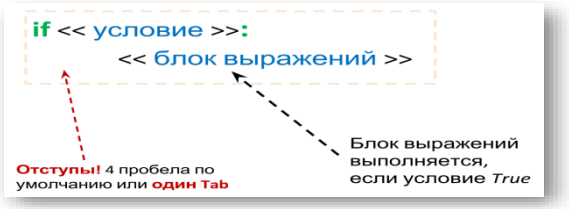
Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Наиболее часто логические выражения используются внутри условной инструкции **if:**



Условный оператор ветвления if, if-else, if-elif-else

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

1. Конструкция if

Синтаксис оператора if выглядит так:

if логическое выражение:

команда_1

команда_2

...

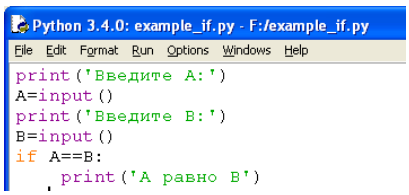
команда_n

После оператора if записывается **логическое выражение**.

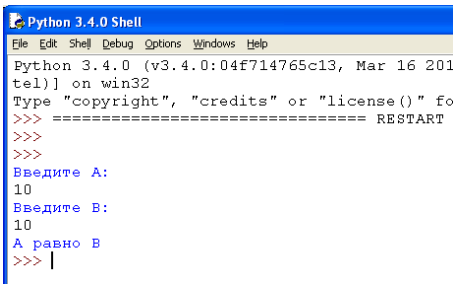
Напомним, **Логическое выражение** — конструкция языка программирования, результатом вычисления которой является «истина» или «ложь».

Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. После условного выражения нужно поставить двоеточие “**:**”.

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение.



Пример программы на Python



Результат выполнения программы с использованием условного оператора if

2. Конструкция if – else

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else.

Синтаксис оператора if – else выглядит так:

if логическое выражение:

команда_1

команда_2

...

команда_n

else:

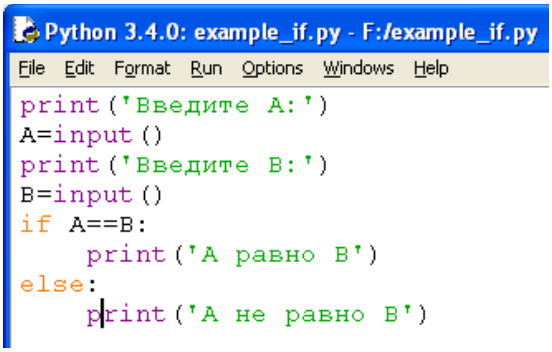
команда_1

команда_2

...

команда_n

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение. В противном случае выводится сообщение, что числа не равны.



Пример программы на Python

```
>>> ================================ RESTART
>>>
Введите A:
10
Введите B:
5
A не равно B
>>>
```

Результат выполнения программы с использованием условного оператора if-else

3. Конструкция if – elif – else

Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.

Синтаксис оператора if – elif – else выглядит так:

if логическое выражение_1:

команда_1

команда_2

...

команда_n

elif логическое выражение_2:

команда_1

команда_2

...

команда_n

elif логическое выражение_3:

команда_1

команда_2

...

команда_n

else:

команда_1

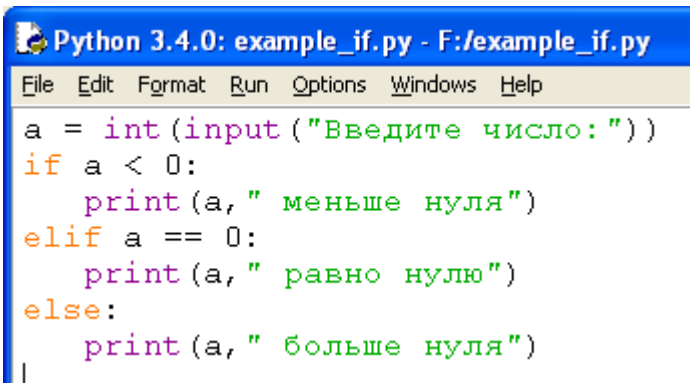
команда_2

...

команда_n

Программа запрашивает число у пользователя и сравнивает его с нулём a<0. Если оно меньше нуля, то выводится сообщение об этом. Если первое

логическое выражение не истинно, то программа переходит ко второму - a==0. Если оно истинно, то программа выведет сообщение, что число равно нулю, в противном случае, если оба вышеуказанных логических выражения оказались ложными, то программа выведет сообщение, что введённое число больше нуля.



Пример программы на Python

```
Введите число:41
41  больше нуля
>>> ================================ RESTART =
>>>
Введите число:-5
-5  меньше нуля
>>> ================================ RESTART =
>>>
Введите число:0
0  равно нулю
>>>
```

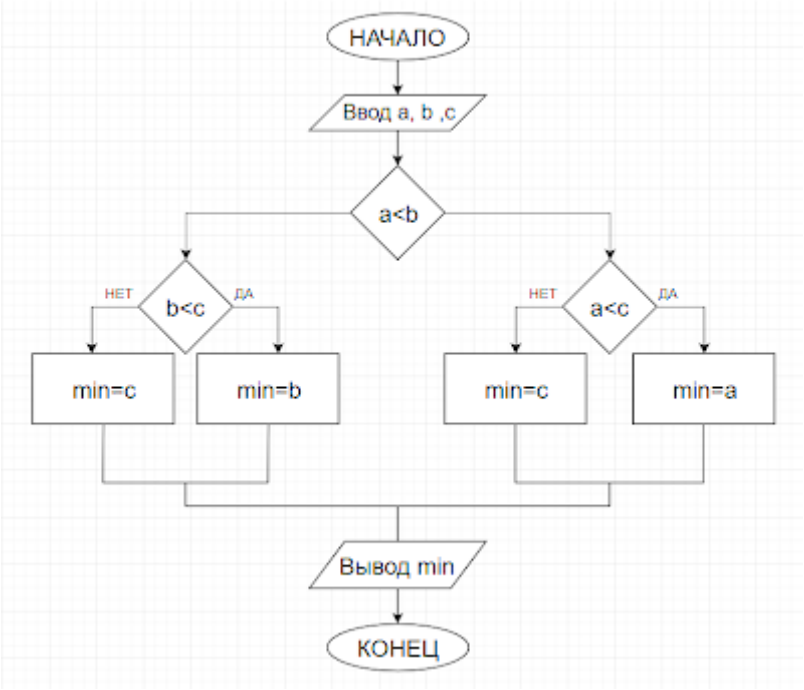
Результат выполнения программы с использованием условного оператора if-elif-else

Пример варианта задания

Дано 3 числа. Найти минимальное среди них и вывести на экран.

Решение

Для простоты построим блок-схему задачи.



Командами

a=input('')

b=input('')

c=input('')

введём три числа, присвоив значения переменным a, b, c.

Условной конструкцией if-else проверим на истинность логическое выражение a<b. Если оно истинно, то переходим на проверку логического выражения a<c. Если оно истинно, то переменной "y" присвоим значение переменной "a", т.е. "a" будет минимальным, а иначе "y" присвоится значение переменной "c".

Если в начале логическое выражение a<b оказалось ложным, то переходим на проверку другого логического выражения b<c.

Если оно истинно, то "y" присвоится значение переменной "b", иначе "c".

Командой print() выводим минимальное значение.

Пример программы:

```
#нахождение минимального из 3-х чисел
a=input('Введите целое число \n')
b=input('Введите целое число \n')
c=input('Введите целое число \n')
if a<b:
    if a<c:
        y=a
    else:
        y=c
else:
    if b<c:
        y=b
    else:
        y=c
print('Минимальное:',y)
```

Результат выполнения программы:

```
Введите целое число
2
Введите целое число
5
Введите целое число
1
Минимальное: 1
```

Выводы и предложения о проделанной работе

Содержание отчета:

7. Наименование практического занятия;
8. Цель занятия;
9. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;
10. Список используемых источников;
11. Выводы и предложения;
12. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Какой вычислительный процесс называется разветвляющимся?
2. Опишите синтаксис оператора if.
3. Дайте определение понятию «логическое выражение»
4. Опишите конструкцию if-elif-else.
5. Что позволяет выполнить оператор ветвления if.

Практическое занятие № 29 Реализация циклических алгоритмов в Python. Синтаксис цикла с предусловием и постусловием

Цель занятия:

1. Познакомиться с циклическими конструкциями;
2. Уметь писать коды программ, содержащие циклические конструкции с while;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №29 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

В Python существуют два типа цикличных выражений:

- Цикл while с предусловием и можно организовать while с постусловием
- Цикл for – (изучаем на следующем практическом занятии).

1. Цикл while в Python *Цикл с предусловием (с заданным условием продолжения работы, цикл «ПОКА»)*

Инструкция while в Python повторяет указанный блок кода до тех пор, пока указанное в цикле логическое выражение будет оставаться истинным.

Синтаксис цикла while:

while логическое выражение:

команда 1

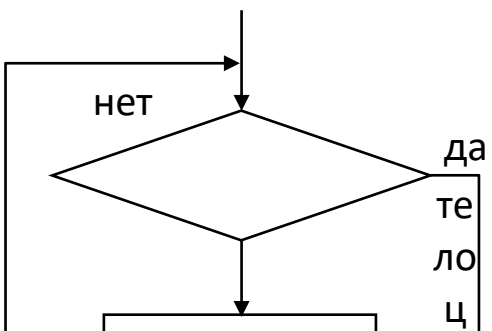
команда 2

...

команда n

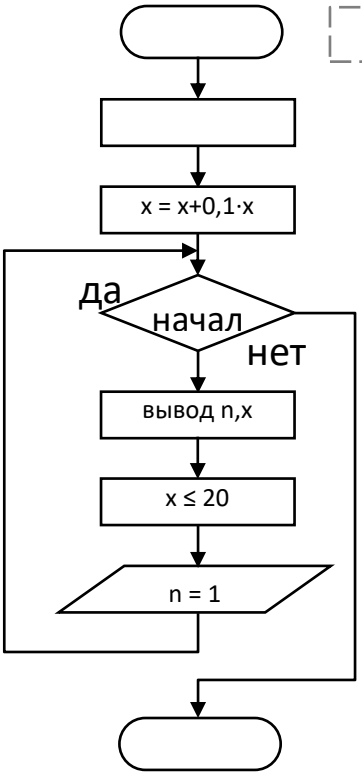
После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

```
while <условие>:
    <блок_операторов>
```



Задача

Лыжник в первый день тренировок пробежал 10 км. Каждый следующий день он увеличивал пройденное расстояние на 10% от пройденного в предыдущий день. В какой день он пробежит больше 20 км?

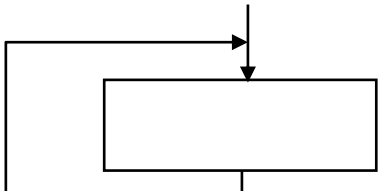


x = 10

```
# Лыжная тренировка
n = 1                  # день номер 1
x = 10                 # расстояние в 1 день
while x<=20:           # пока x<=20 повторять:
    n = n+1            # номер следующего дня
    x = x+0.1*x        # расстояние в следующий день
    print("{:3}".format(n), "{:6.1f}".format(x))
```

2. **Цикл while в Python** *Цикл с постусловием (с заданным условием окончания работы, цикл «ДО»)*

В языке Python нет оператора цикла с постусловием, но его можно организовать с помощью оператора while («пока») с условием True («истина»). Такой цикл будет выполняться бесконечно. Выход из цикла произойдет при истинности условия в операторе ветвления с помощью специального оператора break («прервать»).



Используется в тех случаях, когда требуется, чтобы тело цикла выполнилось хотя бы один раз.

Пример варианта задания

Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

Решение:

Для начала введём целое число командой int(input(текст сообщения)).

Затем зададим переменной i значение 2. Переменная i выполняет роль счётчика. Если задать ей значение 1, то условие задачи не будет выполнено, а результатом всегда будет 1.

В цикле while в качестве логического выражения используется команда n%i сравниваемая с нулём. Таким образом, если остаток от деления введённого числа на текущее значение i не равно нулю, то счётчик увеличивается на 1, а если равно нулю цикл заканчивается и командой print() выводится сообщение и значение i.

Пример программы с циклом while

```
n = int(input('Введите целое число не меньшее 2\n'))
i = 2
while n%i != 0:
    i+=1
print('наименьший натуральный делитель:',i)
```

```
Введите целое число не меньшее 2
49
наименьший натуральный делитель: 7
```

Выводы и предложения о проделанной работе

Содержание отчета:

1.Наименование практического занятия;

2.Цель занятия;

3.Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

4.Список используемых источников;

5.Выводы и предложения;

6.Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Какие типы цикличных выражений существуют в Python?

2. Как организовать цикл с постусловием в Python?

3. Напишите синтаксис цикла while с предусловием .

Практическое занятие № 30 Функция range. Синтаксис цикла с параметром.

*Цель занятия***:**

1. Познакомиться с циклической конструкцией- цикл с параметром;
2. Познакомиться с функцией range;
3. Уметь писать коды программ, содержащие циклические конструкции с for;
4. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №30 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

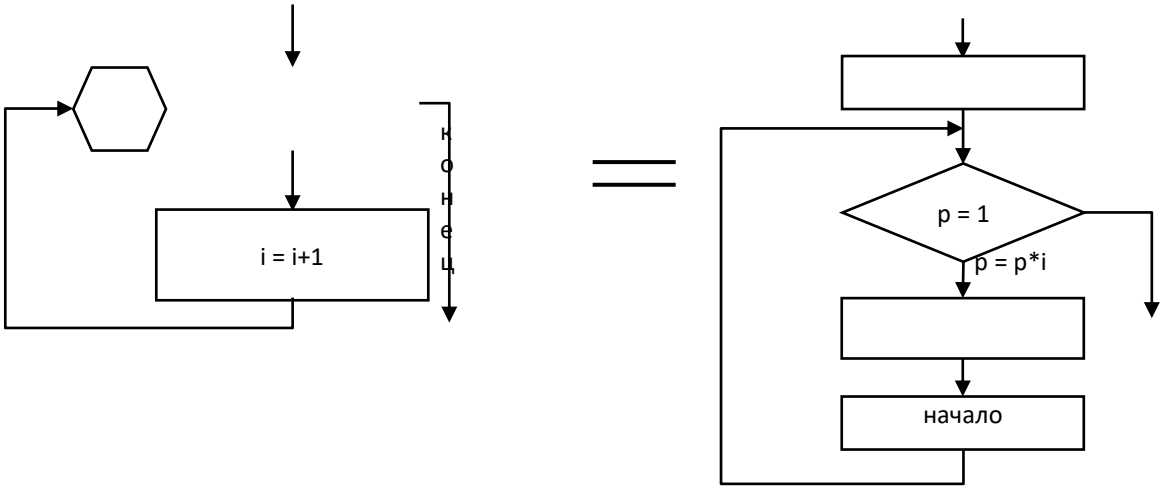
2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Цикл for в Python:

Цикл for в Python обладает способностью перебирать элементы любого комплексного типа данных (например, строки или списка).

Цикл с параметром (с заданным числом повторений, цикл «ДЛЯ»)



Синтаксис цикла for:

```
k < 13
```

Тело цикла повторяется *фиксированное число раз* **для** каждого значения параметра. Параметр – переменная целого типа. Функция **range** («диапазон») задаёт количество повторов тела цикла и содержит от одного до трёх чисел.

range(старт, стоп, шаг) - так выглядит стандартный вызов функции range() в Python. По умолчанию старт равняется нулю, шаг единице.

Одно число **(k)** – параметр цикла изменяется *от **0** до **k-1** с шагом **1***.

Два числа **(n, k)** – параметр цикла изменяется *от **n** до **k-1** с шагом **1***.

Три числа **(n, k, s)** – параметр цикла изменяется *от **n** до **k-1** с шагом **s***.

Возможно изменение параметра от большего значения к меньшему. В этом случае **n** должно быть больше **k**, а **s** – отрицательное.

Функция range()

Достаточно часто при разработке программ необходимо получить последовательность (диапазон) целых чисел:

Для решения этой задачи в Python предусмотрена функция range(), создающая последовательность (диапазон) чисел. В качестве аргументов функция принимает: начальное значение диапазона (по умолчанию 0), конечное значение (не включительно) и шаг (по умолчанию 1). Если вызвать функцию, то результата мы не увидим:

```
>>> range(0,10,1)
range(0, 10)
>>> range(10)
range(0, 10)
```

Дело в том, что для создания диапазона чисел необходимо использовать цикл for:

```
>>> for i in range(0, 10, 1): print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
>>> for i in range(10):
print(i, end=' ')
```

0 1 2 3 4 5 6 7 8 9

Синтаксис цикла for:

for int **in** range():

команда 1

команда 2

...

команда

Переменной int присваивается значение первого элемента функции range(), после команда 2 чего выполняются команды. Затем переменной int присваивается следующее по порядку значение и так далее до тех пор, пока не будут перебраны все элементы команда функции range().

Функция range() является универсальной функцией Python для создания списков (list) содержащих арифметическую прогрессию. Чаще всего она используется в циклах for.

Пример варианта задания

1. Найти сумму n элементов следующего ряда чисел: 1 -0.5 0.25 -0.125 ... n. Количество элементов (n) вводится с клавиатуры. Вывести на экран каждый член ряда и его сумму. Решить задачу используя циклическую конструкцию for.

Решение:

В данном случае ряд чисел состоит из элементов, где каждый следующий меньше предыдущего в два раза по модулю и имеет обратный знак. Значит, чтобы получить следующий элемент, надо предыдущий разделить на -2.

Какой-либо переменной надо присвоить значение первого элемента ряда (в данном случае это 1). Далее в цикле добавлять ее значение к переменной, в которой накапливается сумма, после чего присваивать ей значение следующего элемента ряда, разделив текущее значение на -2. Цикл должен выполняться n раз.

```
n=int(input('Введите количество элементов последоваетльности: '))
x=1
s=0
print(x)
for i in range(n):
    s+=x
    x/=-2
    print(x)
print('Сумма ряда:',s)
```

Пример программы с циклом for

```
Введите количество элементов последоваетльности: 5
1
-0.5
0.25
-0.125
0.0625
-0.03125
Сумма ряда: 0.6875
```

Результат выполнения программы

Выводы и предложения о проделанной работе

Содержание отчета:

7.Наименование практического занятия;

8.Цель занятия;

9.Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

10. Список используемых источников;

11. Выводы и предложения;

12. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Какие типы цикличных выражений существуют в Python?

2. Какой способностью обладает цикл for в Python?

3. Как работает функция range?

Практическое занятие № 31 Работа со строками. Понятие списка в Python. Создание и считывание списков. Функции и методы списков.

Цель занятия:

1. Изучить методы работы со строками;
2. Овладеть умениями по созданию и считыванию списков;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №31 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Строка — базовый тип, представляющий из себя неизменяемую последовательность символов; str от «string» — «строка».

Строковая константа (строка) – произвольная последовательность символов из таблицы Unicode, заключенная в одинарные или двойные кавычки (тип **str** – «string»).

Например:

'Это строка'

"Это тоже строка“

Вызовем функцию type() и передадим ей на вход целочисленный аргумент:

```
>>> type(0)
<class 'int'>
>>>
```

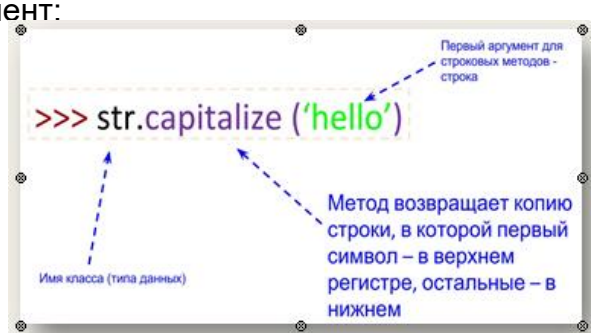
Функция сообщила нам, что объект 0 относится к классу 'int', т.е. тип данных является классом (тип данных и класс – синонимы).

Класс будем представлять, как некий аналог модуля, т.е. набор функций и переменных, содержащихся внутри класса. Функции, которые находятся внутри класса, называются методами.

Рассмотрим пример вызова строкового метода:

```
>>> str.capitalize('hello')
'Hello'
>>>
```

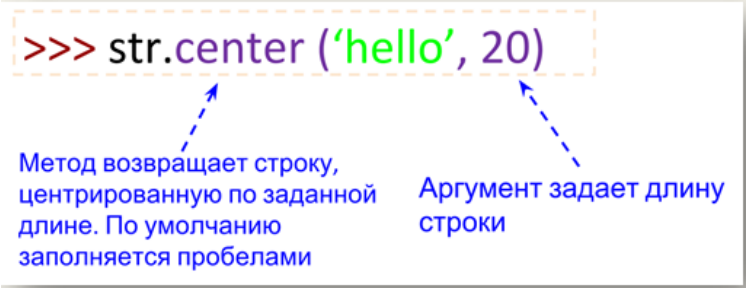
По аналогии с вызовом функции из модуля указываем имя класса – str, затем через точку пишем имя строкового метода capitalize(), который принимать один строковый аргумент:



Метод – это обычная функция, расположенная внутри класса. Вызовем еще один метод:

```
>>>
str.center('hello',20)
'       hello        '
>>>
```

тот метод принимает два аргумента – строку и число:



Форма вызова метода через обращение к его классу через точку называется полной формой.

Постоянно писать имя класса перед вызовом каждого метода быстро надоест, поэтому чаще всего используют сокращенную форму вызова метода:

```
>>>
'hello'.capitalize()
'Hello'
>>>
```

В примере мы вынесли первый аргумент метода и поместили его вместо имени класса:

Вынесенныйиз метода первый строковый аргумент может быть выражением, возвращающим строку:

```
>>> ('TTA' +
'G'*3).count('T') 2
>>>
```

Не сложно догадаться, что делает метод count().

Python содержит интересный метод format()38:

```
>>> '{0} и {1}'.format('труд', 'май')
 'труд и май'
>>>
```

Вместо {0} и {1} подставятся аргументы методы format(). Поменяем их местами:

```
>>> '{1} и {0}'.format('труд',
'май') 'май и труд'
```

```
>>>
```

Формат вывода методы format() может варьироваться:

```
>>> n = 10
>>> '{:b}'.format(n)    # вывод в двоичной системе
счисления '1010'
>>> '{:c}'.format(n)    # вывод в формате
Unicode '\n'
>>> '{:d}'.format(n)    # по снованию
10 '10'
>>> '{:x}'.format(n)    # по основанию
16 'a'
>>>
```

В Python есть полезные строковые методы, которые возвращают (True) истину или (False) ложь:

```
>>> 'spec'.startswith('a')
False
>>>
```

Метод startswith() проверяет, начинается ли строка с символа, переданного в качестве аргумента методу.

Метод swapcase() возвращает строку с противоположными регистрами символов:

```
>>>
'Hello'.swapcase()
'hELLO'
>>>
```

Специальные строковые методы

Длина строки – количество символов в строке .

Пустая строка – строка с нулевой длиной (не содержит ни одного символа, обозначается **""**).

1. Присваивание значения строковой переменной

s = "Привет"

2. Ввод строки с клавиатуры

n = input("Введите имя: ")

Примечание: при вводе значения строки кавычки не вводятся.

3. Вывод строки на экран

print (n)

Примечание: при выводе строки кавычки не выводятся.

4. Объединение строк (конкатенация)

Соединяет несколько строк в одну строку. Обозначается знаком **+**.

Например:

"КОМ"+"ПЬЮ"+"ТЕР" # "КОМПЬЮТЕР"

"10"+"2" # "102"

Операции или методы работы со строковыми величинами вы можете посмотреть в таблице ниже.

Функции и методы работы со строками

Функция или метод	Назначение
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.join(список)	Соединение строк из последовательности str через разделитель, заданный строкой
S1.count(S[, i, j])	количество вхождений подстроки s в строку s1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска *j*
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
s.title()	Возвращает строку, первый символ которой в верхнем регистре
s.count('e', 1, 5)	Возвращает количество подстрок в интервале либо -1
s.isalpha()	Проверяет, состоит ли строка только из букв
s.isdigit()	Проверяет, состоит ли строка только из чисел
s.isupper()	Проверяет, написаны ли все символы в верхнем регистре
s.islower()	Проверяет, написаны ли все символы в нижнем регистре
s.istitle()	Проверяет, начинается ли строка с большой буквы
s.isspace()	Проверяет, состоит ли строка только из пробелов

Каждый из перечисленных ниже методов запустить в интерактивном режиме на примере различных строк.

Предположим, что переменная s содержит некоторую строку, тогда применим к ней методы:

При работе со строковыми величинами можно выполнять :

1.Выделение части строки (подстроки)

s[n:k]

Выделяет из строки **s** часть строки от позиции **n** до **k-1**.

s[:k]

Выделяет из строки **s** часть строки от начала до **k-1**.

s[n:]

Выделяет из строки **s** часть строки от позиции **n** до конца.

Примечание: символы нумеруются, начиная с 0.

Например:

```
s = "ИНФОРМАТИКА"
print(s[2:7])      # "ФОРМА"
```

2.Преобразование типов

При необходимости можно преобразовать число в строку или наоборот.

int(s) – преобразует строку в целое число.

float(s) – преобразует строку в вещественное число.

str(n) – преобразует целое или вещественное число в строку.

Примечание: если в строке содержатся символы, не допустимые для чисел, возникнет ошибка.

Например:

```
s1 = "123"          # строка цифр
i = int(s1)           # i=123
f = float(s1)         # f=123.0
s2 = str(f)           # s2="123.0"
print(i, f, len(s2)) # 123 123.0 5
```

3.*Операции с кодами символов*

ord(s) – возвращает код символа **s**.

chr(k) – возвращает символ с кодом **k**.

Примечание: коды из кодовой таблицы Unicode.

Например:

```
print (ord("И"))                  # 1048
```

print (chr(1048)+chr(1050)+chr(1058)) # ИКТ

4.*Сравнение строк*

Операции отношения: **==, !=, <, >, <=, >=.**

Сравнение строк производится слева направо до первого несовпадающего символа. Строка считается больше, если первый несовпадающий символ имеет больший код в кодовой таблице (пробел, цифры, латинские заглавные, латинские строчные, русские заглавные, русские строчные). Строки равны, если они совпадают по длине и содержат одни и те же символы.

"1STR" < "STR" < "Str" < "str" < "ШАГ" < "Шаг" < "шаг"

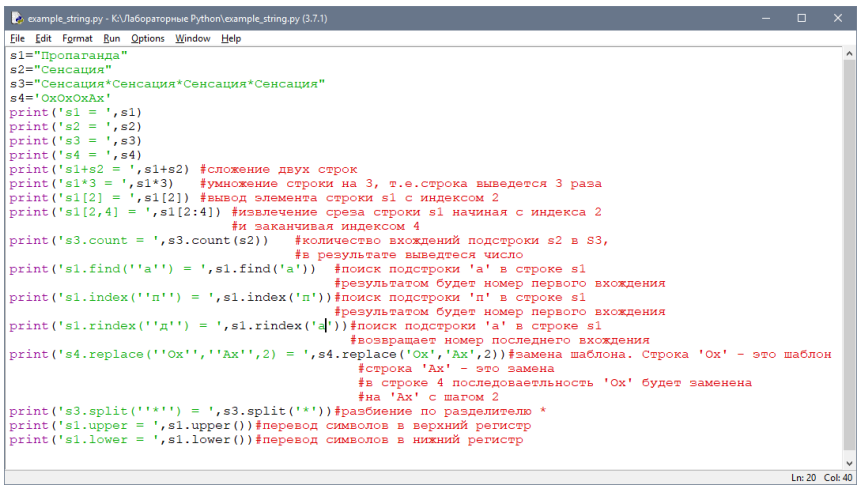
5.*Перебор символов строки*

Часто в программах требуется перебирать по одному все символы строки для проведения с ними каких-либо действий. Для этого удобно использовать следующий цикл:

for C in S: # перебор символов в строке S

  <операторы> # очередной символ в перем. C

Ниже приведена программа, демонстрирующая использование функций и методов работы со строками.



Пример программы на Python

```
Python 3.7.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.19
1)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
>>>
============= RESTART: K:\Лабораторные Python\example_string.py =
s1 =  Пропаганда
s2 =  Сенсация
s3 =  Сенсация*Сенсация*Сенсация*Сенсация
s4 =  OxOxOxAx
s1+s2 =  ПропагандаСенсация
s1*3 =  ПропагандаПропагандаПропаганда
s1[2] =  о
s1[2,4] =  оп
s3.count =  4
s1.find(а) =  4
s1.index(п) =  3
s1.rindex(д) =  9
s4.replace(Ox,Ax,2) =  AxAxOxAx
s3.split(*) =  ['Сенсация', 'Сенсация', 'Сенсация', 'Сенсация']
s1.upper =  ПРОПАГАНДА
s1.lower =  пропаганда
```

Результат выполнения программы с использованием функций и методов работы со строками

Список (list) – это структура, состоящая из элементов, расположенных в определенном порядке. Каждому элементу соответствует номер (или индекс), по которому к нему можно обратиться. Для создания списка вквадратных скобках ([]) через запятую перечисляются все его элементы. Например, создадим список членов своей семьи:

```
>>>myfamily=['father','mother','sister','brother']
```

Программа на Python	Результат вывода на экран

`myfamily = ['father', 'mother', 'sister', 'brother']`
`for item in myfamily:`
`  print('Hello', item)`
	Hello father
	Hello mother
	Hello sister
	Hello brother

В данном случае наш список будет храниться в переменной myfamily.

Когда список создан, можно написать программу для работы с этим списком. К примеру, напишем приветствие для каждого из членов семьи, используя цикл:

Каждый элемент списка имеет свой номер (индекс). Нумерация элементов начинается с нуля:

a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]

0	1	2	3	4
Андрей	Вера	Даша	Коля	Юра
a[0]	a[1]	a[2]	a[3]	a[4]

С каждым элементом списка можно работать отдельно:

```
a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]
print(a[2])
```

Вывод: **Даша**

Список может содержать разные типы объектов. В один и тот же список одновременно можно включать строки, числа, объекты других типов данных:

```
objects=[1,2.6,'Hello',True]
```

Списки можно складывать, тогда новый список будет содержать элементы из обоих списков:

x = [1, 2, 3, 4]

y = [5, 6, 7, 8]

z = x + y print (z)

Результат: [1, 2, 3, 4, 5, 6, 7, 8]

Со списками можно делать много разных операций:

x in A	Проверить, содержится ли элемент **x** в списке **A**. Возвращает True или False	`A = [1, 2, 3, 4, 5, 6, 7, 8]`
		`print (2 in A)`
		`Результат:`
		`True`

min(A)	Найти наименьший элемент в списке **A.**	`A = [1, 2, 3, 4, 5, 6, 7, 8]`
		`print (min(A))`
		`Результат:`
		`1`

max(A)	Найти наибольший элемент в списке **A.**	`A = [1, 2, 3, 4, 5, 6, 7, 8]`
		`print (max(A))`
		`Результат:`
		`8`

Функция нахождения длины списка len(a):

```
a = ["Яблоко", "Банан", "Груша"]
x = len(a)
print(x)
```

Результат:3

Сортировка списка. Функция sorted(a):

По возрастанию:

```
animals = ["кот", "еж", "собака", "барсук"]
animals = sorted(animals)
print(animals)
```

Вывод:

['барсук', 'еж', 'кот', 'собака']

Поубыванию:

```
a = [5, 65, 14, 700, 8]
a = sorted(a, reverse = True)
print(a)
```

Вывод:

[700, 65, 14, 8, 5]

Способы вывода списков

1)*С помощью функции*print():

```
b = [17, 409, 88]
s=[‘f’,’g’,’r’,’a’]
print(b)
print(s)
```

Вывод:

[17, 409, 88]

[f,g,r,a]

2) *Вывод каждого элемента списка по отдельности*:

```
a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]
fori in range(5):
print(a[i])
```

Вывод:

Андрей

Вера

Даша

Коля

Юра

3)Вывод каждого элемента списка поотдельности в одной строке:

```
a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]
fori in range(5):
print(a[i], end = " ")
```

Вывод

Андрей Вера Даша Коля Юра

4) Вывод элементов списка без обращения к индексам элементов:

```
fruits = ["Яблоко", "Банан", "Груша"]
for x in fruits:
print(x, end = " ")
```

Вывод

Яблоко Банан Груша

Способы вывода списков

1)С помощью функции print():

```
b = [17, 409, 88]
s=[‘f’,’g’,’r’,’a’]
print(b)
print(s)
```

Вывод:

```
 [17, 409, 88]
 [f,g,r,a]
```

2) Вывод каждого элемента списка по-отдельности:

```
a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]
for i in range(5):
    print(a[i])
```

Вывод:

```
 Андрей
Вера
Даша
```

Коля
Юра
3)Вывод каждого элемента списка по-отдельности в одной строке:

```
a = ["Андрей", "Вера", "Даша", "Коля", "Юра"]
for i in range(5):
    print(a[i], end = " ")
```

Вывод
Андрей Вера Даша Коля Юра

4) Вывод элементов списка без обращения к индексам элементов:

```
fruits = ["Яблоко", "Банан", "Груша"]
for x in fruits:
    print(x, end = " ")
```

Вывод
Яблоко Банан Груша

Пример варианта задания

Проверить, будет ли строка читаться одинаково справа налево и слева направо (т. е. является ли она палиндромом).

Решение

Сначала введём строку командой: s=input('Введите строку ').

Затем определим логическую переменную flag и присвоим ей значение 1: flag=1.

Для начала в введённой строке нужно удалить пробелы. Для этого воспользуемся циклической конструкцией for, которая выполнится столько раз, какую имеет длину строка. Длину строки определим функцией len(s).

В теле цикла будем проверять следующее условие: s[i]!=' '. Данное логическое выражение будет истинно в том случае, если i-ый элемент строки не будет равен пробелу, тогда выполнится команда следующая после двоеточия: string+=s[i].

К строке string, которая была объявлена в начале программы, будет добавляться посимвольно строка s, но уже без пробелов.

Для проверки строки на "палиндром" воспользуемся циклической конструкцией for.

Длина половины строкинаходится делением нацело на 2.Если количество символов нечетно,то стоящий в середине не учитывается,т.к. его сравниваемая пара - он сам.

Количество повторов цикла равно длине половины строки. Длину строки определим функцией len(s), где аргумент введённая нами строка s. Зная длину строки, можно вычислить количество повторов цикла. Для этого целочисленно разделим длину строки на 2: len(s)//2.

Для задания диапазона для цикла используем функцию range(), в которой аргументом будет являться половина длины строки: range(len(s//2)).

for i in range(len(s//2)).

Если символ с индексом i не равен "симметричному" символу с конца строки (который находится путем индексации с конца)

if s[i] != s[-1-i],

то переменной flag присваивается значение 0 и происходит выход из цикла командой break.

Далее, при помощи условной конструкции if-else в зависимости от значения flag либо - 0, либо -1 выводится сообщение, что строка палиндром, либо нет.

```
s=input('Введите строку \n')
flag=1
string=''
for i in range(len(s)):
    if s[i]!=' ':
        string+=s[i]
print(string)
for i in range(len(s)//2):
    if string[i]!=string[-i-1]:
        flag=0
        break
if flag: print('Палиндром')
else: print('не палиндром')
```

Пример программы на Python

```
Введите строку
а роза упала на лапу азора
арозаупаланалапуазора
Палиндром
```

Результат выполнения программы

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;

2. Цель занятия;

3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;

4. Список используемых источников;

5. Выводы и предложения;

6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Что называется строкой в Python

2. Перечислите функции и методы работы со строками.

3. Для чего служит функция S.split**?**

4. Для чего используется функция range()?

Практическое занятие № 32 Понятие картежа и словаря. Создание словарей и кортежей. Методы словарей.

Цель занятия:

1. Изучить понятие кортежей и словарей в Python;
2. Овладеть операциями и методами над кортежами и словарями;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №32 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Часто нам нужно держать много однообразных данных в одном файле, например, список учеников колледжа или номера телефонов в справочнике. В Python такие наборы данных можно организовывать в списки, кортежи и словари.

Кортеж (**tuple**), как и список, представляет собой последовательность элементов. Однако хранящиеся в нем элементы нельзя изменять, добавлять или удалять. Для создания кортежа используются круглые скобки, в которые помещаются его значения, разделенные запятыми:

```
user=('Timur',23,1.10.1998)
print(user)
```

В кортежах удобно хранить свойства объектов, например, имя, возраст, дату рождения. Если вдруг кортеж состоит из одного элемента, то после единственного элемента кортежа необходимо поставить запятую:

```
user=('Tom',)
```

Некоторые операции над кортежами:

```
>>> ()     #  создание пустого
кортежа ()
>>> (4)    # это не кортеж, а целочисленный
объект! 4
>>> (4,)   # а вот это – кортеж, состоящий из одного
элемента! (4,)
>>> b=('1',2,'4')  # создаем кортеж
>>> b
('1', 2, '4')
>>> len(b)  # определяем длину
кортежа 3
>>> t=tuple(range(10)) # создание кортежа с помощью функции range()
>>> t+b   # слияние кортежей
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, '1', 2, '4')
>>> r=tuple([1,5,6,7,8,'1']) # кортеж из списка
>>> r
(1, 5, 6, 7, 8, '1')
>>>
```

С помощью кортежей можно присваивать значения одновременно двум переменным:

```
>>> (x,y)=(10,5)
>>
>
x
10
>>
>
y
5
>>> x,y = 1,3 # если убрать круглые скобки, то результат не изменится
```

```
>>
>
x
1
>>
>
y
3
>>>
```

Поменять местами содержимое двух переменных:

```
>>> x, y = y, x
>>
>
x
3
>>
>
y
1
>>>
```

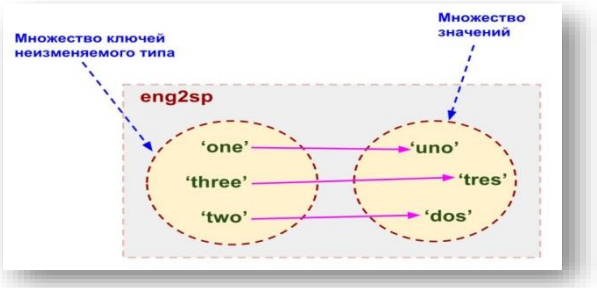
Кортеж нельзя изменить, но можно изменить, например, список, входящий в кортеж:

```
>>> t = (1,[1,3],'3')
>>>
t[1]
[1, 3]
>>> t[1][0] = '1'
>>> t
(1, ['1', 3], '3')
>>>
```

Словари (**dictionary**) – это структура данных, в которой каждый элемент вместо индекса имеет уникальный ключ. Элементы словаря можно изменять. Для создания словаря используются фигурные скобки ({}):

1. Словари создаются с помощью фигурных скобок.
2. Пары из ключа и значения разделяются запятыми.
3. Ключи и значения разделяются между собой двоеточием
4. Ключи в словаре могут быть только строками, целыми числами или числами с плавающей точкой. А вот значения могут быть любого типа
5. Важно не забывать использовать кавычки для строки-ключа

Пример создания словаря, который каждому слову на английском языка будет ставить в соответствие слово на испанском языке.



Фактически, словарь – это отображение двух множеств: множества ключей и множества значений.

Некоторые из методов и функций словарей:

- `.keys()` — используется для вывода ключей словаря.
- `.items()` — используется для создания кортежей с ключами и значениями.
- `.get()` — метод для получения значения по ключу.
- `.clear()` — очистить словарь.
- `.copy()` — скопировать весь словарь.
- `len()` — получить длину словаря.
- `type()` — узнать тип.

К словарям применим оператор in:

```
>>> eng2sp
{'three': 'tres', 'one': 'uno', 'two': 'dos'}
>>> 'one' in eng2sp  # поиск по множеству
КЛЮЧЕЙ True
>>>
```

Часто словари используются, если требуется найти частоту встречаемости элементов в последовательности (списке, строке, кортеже).

Функция, которая возвращает словарь, содержащий статистику встречаемости элементов в последовательности:

```
def histogram(s):
    d = dict()
    for c in s:
        if c not in
d:
            d[c]=1
        else:
            d[c]=d[c]+1 # или d[c] +=
1
    return d
```

Результат вызова функции histogram() для списка, строки, кортежа соответственно:

```
>>> histogram([2,5,6,5,4,4,4,4,3,2,2,2,2])
{2: 5, 3: 1, 4: 4, 5: 2, 6: 1}
>>> histogram("ywte3475eryt3478e477477474")
{'4': 6, '8': 1, 'e': 3, '3': 2, '7': 7, '5': 1, 'r': 1, 'y':
2, 'w':
1, 't': 2}
>>> histogram((5,5,5,6,5,'r',5))
{5: 5, 6: 1, 'r': 1}
>>>
```

Начиная с версии Python 3.9, в языке появились новые операторы, которые облегчают процесс слияния словарей.

1. Merge (|): этот оператор позволяет объединять два словаря с помощью одного символа |.
2. Update (|=): с помощью такого оператора можно обновить первый словарь значением второго (с типом dict)

Вот основные отличия этих двух операторов:

- «|» создает новый словарь, объединяя два, а «|=» обновляет первый словарь.
- Оператор merge (|) упрощает процесс объединения словарей и работы с их значениями.
- Оператор update (|=) используется для обновления словарей.

```
dictionary={ключ1:значение1,ключ2:значение2,..
.}
```

Создадим словарь под именем mycollege:

```
mycollege = {'4kurs':'Анна, Кирилл, Pavel',
'3kurs':'Игорь,Олег,Ольга'}
```

В этом словаре в качестве ключей используются названия классов, а в качестве значений – имена тех, кто учится в этих классах.

В словарь можно добавить значение, пометив его новым ключом:

```
mycollege['2kurs']='Elena,Алиса,Дима'
print(mycollege)
```

Результат:

```
{'4kurs': 'Анна, Кирилл, Pavel', '3kurs':
'Игорь,Олег,Ольга','2kurs':'Elena,Алиса,Дима}
```

Чтобы изменить значение элемента, нужно придать его ключу новое значение:

```
mycollege = {'4kurs':'Анна, Кирилл, Pavel',
'3kurs':'Игорь,Олег,Ольга'}
mycollege['3kurs']='Matvei,Таня,Света'
print(mycollege)
```

Результат:

```
{'4kurs':Анна,Кирилл,Pavel','3kurs':'Matvei,Таня,Света'}
```

Используя циклfor,можно вывести на экран только ключи словаря:

```
foriinmycollege:
    print(i)
```

Результат:

```
4 kurs
3 kurs
```

Или вывести только значения словаря:

```
foriinmycollege:
    print(mycollege[i])
```

Результат:

```
Анна,Кирилл,Pavel
Matvei, Таня, Света
```

Пример варианта задания

Создать список своих любимых фильмов. Вывести список тремя способами: а) в строчку; б) в столбик; в) в строчку через запятую.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
4. Список используемых источников
5. Выводы и предложения
6. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки

1. Что такое список, кортеж, словарь в Питоне?
2. Как создаются списки?
3. Какие операции можно производить со списками?
4. Какие существуют способы вывода списка?
5. Как записывается функция определения длины списка?
6. Как отсортировать список?

Практическое занятие № 33 Применение списков и словарей в различных задачах

Цель занятия:

1. Научиться составлять программы с применением списков и словарей;
2. Закрепить умение применять методы списков и словарей в задачах.
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №33 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

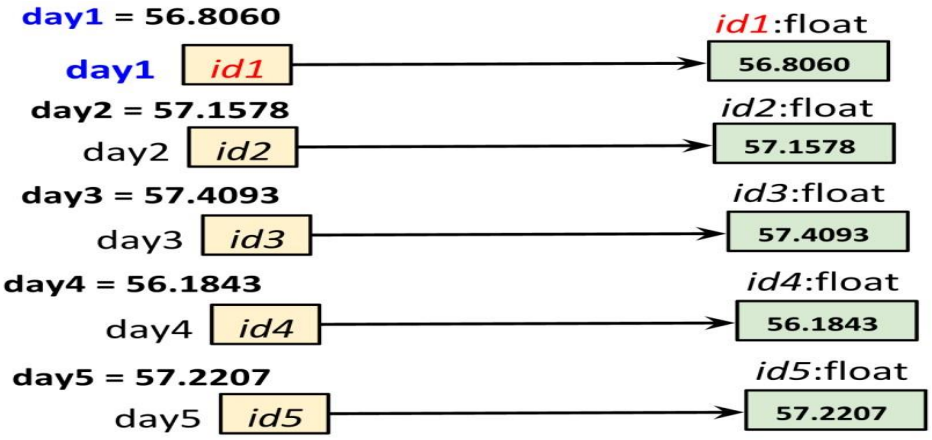
Предположим, что нам необходимо обработать информацию о курсах валют

Дата	Доллар США USD	ЕВРО EUR
16.05.2015	50.0115	56.9881
15.05.2015	50.0774	57.1383
14.05.2015	49.5366	55.7138
13.05.2015	50.9140	57.1102
09.05.2015	50.7511	56.8971
08.05.2015	50.3615	57.2207
07.05.2015	49.9816	56.1843
06.05.2015	51.7574	57.4093
01.05.2015	51.1388	57.1578
30.04.2015	51.7029	56.8060
29.04.2015	52.3041	56.9016
28.04.2015	51.4690	55.8747
25.04.2015	50.2473	54.6590
24.04.2015	51.6011	55.1255
23.04.2015	53.6555	57.7226
22.04.2015	53.9728	57.5998

Мы можем курс валюты на каждый день поместить в отдельную переменную:

```
>>> day1 = 56.8060
>>> day2 = 57.1578
>>>
```

СХЕМАТИЧНО ЭТО ВЫГЛЯДИТ ТАК



Тут нам на помощь приходят списки. Их можно рассматривать как аналог массива, о которых мы будем говорить на следующих занятиях, за исключением важной особенности – списки в качестве своих элементов могут содержать любые объекты. Но обо всем по порядку.

Список (list) в Python является объектом, поэтому может быть присвоен переменной (переменная, как и в предыдущих случая, хранит адрес объекта класса список).

Представим список для нашей задачи с курсом валют:

```
>>> e = [56.8060, 57.1578, 57.4093, 56.1843, 57.2207]
>>> e
[56.806, 57.1578, 57.4093, 56.1843, 57.2207]
>>>
```

Список позволяет хранить разнородные данные, обращаться к которым можно через имя списка (в данном случае переменную e).

Обращаться к отдельным элементам списка можно по их индексу (позиции), начиная с нуля:

```
>>> e=[56.8060, 57.1578, 57.4093, 56.1843, 57.2207]
>>> e[0]
 56.806
>>> e[1]
57.1578
>>> e[-1]          # последний элемент
 57.2207
>>>
```

Обращение по несуществующему индексу вызовет ошибку:

```
 >>> e[100]
Traceback (most recent call last):
File "<pyshell#10>", line 1, in <module> e[100]
IndexError: list index out of range
>>>
```

Списки можно изменить. Проведем эксперимент:

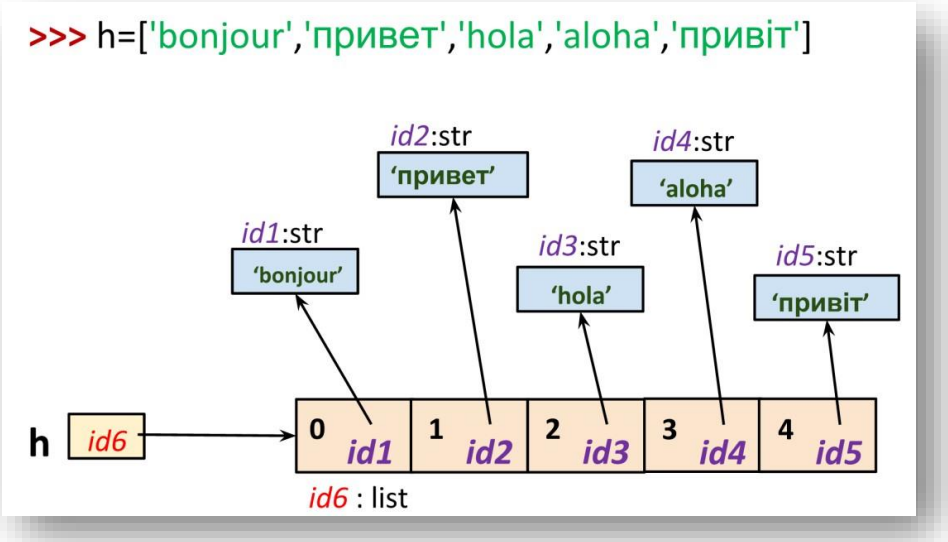
```
 >>> h=['Hi', 27 , -8.1, [1,2]]
>>> h[1]='hello'
>>> h
['Hi', 'hello', -8.1, [1, 2]]
>>> h[1]
'hello'
>>>
```

В примере мы создали список и изменили элемент, находящийся в позиции 1. Видим, что список изменился.

Рассмотрим еще один пример и покажем, что происходит в памяти:

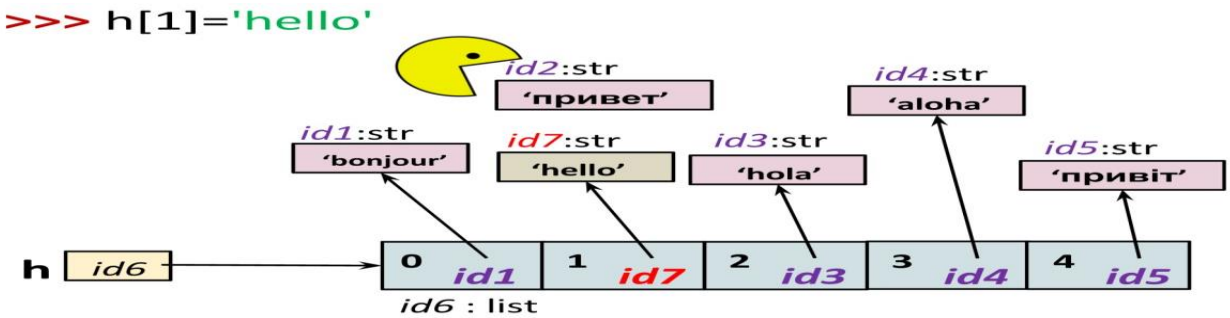
```
>>> h=['bonjour','привет','hola','aloha','привіт']
>>>
```

В памяти:



Производим изменения списка:

```
>>> h[1]='hello'
>>> h
['bonjour', 'hello', 'hola', 'aloha', 'привіт']
>>> h[1]
'hello'
>>>
```



В момент изменения списка в памяти создается новый строковый объект 'hello'. Затем адрес на этот объект (*id*7) помещается в первую ячейку списка (вместо *id*2). Python увидит, что на объект по адресу *id*2 нет ссылок, поэтому удалит его из памяти.

Список (list), наверное, наиболее часто встречающийся тип данных, с которым приходится сталкиваться при написании программ. Это связано со встроенными в Python функциями, которые позволяют легко и быстро обрабатывать списки:

Рассмотрим интересный пример, напишем функцию, объединяющую два списка.

Получится следующее:

```
>>> def f(x,y):
return x+y
>>> f([1,2,3],[4,5,6])
[1, 2, 3, 4, 5, 6]
>>>
```

Теперь передадим в качестве аргументов две строки:

```
>>> f("123", "456")
'123456'
>>>
```

Передадим два числа:

```
>>> f(1,2)
3
```

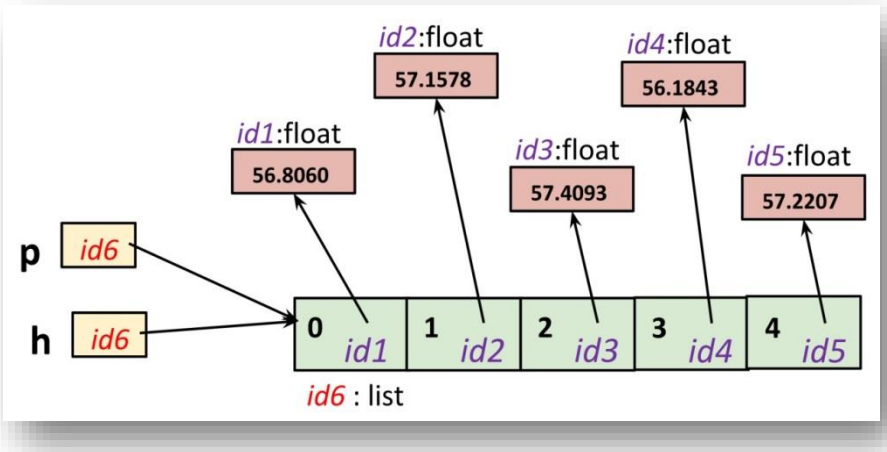
Вернемся к инструкции del и удалим с помощью среза подсписок:

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4]   # удаление подсписка начиная со 2 по 4
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a []
```

```
>>>
```

Рассмотрим важную особенность списков. Выполним следующий код:

```
>>> h
['bonjour', 7, 'hola', -1.0, 'привет']
>>> p=h          # содержат указатель на один и тот же список
>>> p
['bonjour', 7, 'hola', -1.0, 'привет']
>>> p[0]=1 # модифицируем одну из переменных
>>> h    # изменилась другая переменная!
[1, 7, 'hola', -1.0, 'привет']
>>> p
[1, 7, 'hola', -1.0, 'привет']
>>>
```



В Python две переменные называются *псевдонимами*, когда они содержат одинаковые адреса памяти.

На схеме видно, что переменные p и h указывают на один и тот же список:

Пример варианта задания

Пример

Задача «Скрабл»

В настольной игре Скрабл (Scrabble) каждая буква имеет определенную ценность. В случае с английским алфавитом очки распределяются так:

- A, E, I, O, U, L, N, S, T, R – 1 очко;

- D, G – 2 очка;
- B, C, M, P – 3 очка;
- F, H, V, W, Y – 4 очка;
- K – 5 очков;
- J, X – 8 очков;
- Q, Z – 10 очков.

А русские буквы оцениваются так:

- А, В, Е, И, Н, О, Р, С, Т – 1 очко;
- Д, К, Л, М, П, У – 2 очка;
- Б, Г, Ё, Ь, Я – 3 очка;
- Й, Ы – 4 очка;
- Ж, З, Х, Ц, Ч – 5 очков;
- Ш, Э, Ю – 8 очков;
- Ф, Щ, Ъ – 10 очков.

Напишите программу, которая вычисляет стоимость введенного пользователем слова. Будем считать, что на вход подается только одно слово, которое содержит либо только английские, либо только русские буквы.

Пример ввода:
ноутбук

Пример вывода:
12

Решение:

Сначала напишем функцию isCyrillic(), которая возвращает True, если введенное слово содержит кириллические символы, и False, если буквы – латинские.

Затем создадим словари, где ключами будут очки, а значениями – строки из букв. Метод items() позволяет обращаться к ключам и значениям одновременно – если очередная буква в слове входит в одно из значений, генератор добавит ключ в список, а метод списка sum() подсчитает стоимость всего слова:

```
import re
def isCyrillic(text):
          return bool(re.search('[а-яА-Я]', text))
points_en = {1:'AEIOULNSTR',
          2:'DG',
          3:'BCMP',
          4:'FHVWY',
          5:'K',
          8:'JZ',
          10:'QZ'}
points_ru = {1:'АВЕИНОРСТ',
          2:'ДКЛМПУ',
          3:'БГЁЬЯ',
          4:'ЙЫ',
```

```
            5:'ЖЗХЦЧ',
            8:'ШЭЮ',
            10:'ФЩЪ'}
text = input().upper()
if isCyrillic(text):
            print(sum([k for i in text for k, v in points_ru.items() if i in v]))
else:
            print(sum([k for i in text for k, v in points_en.items() if I in v]))
```

Выводы и предложения о проделанной работе

Содержание отчета:

7. Наименование практического занятия
8. Цель занятия
9. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
10. Список используемых источников
11. Выводы и предложения
12. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки

7. Что такое список, кортеж, словарь в Питоне?
8. Как создаются списки?
9. Какие операции можно производить со списками?
10. Какие существуют способы вывода списка?
11. Как записывается функция определения длины списка?
12. Как отсортировать список?

Практическое занятие №34 Подпрограммы :процедуры и функции

Цель занятия:

1. Изучить процедуры и функции в Python;

2. Знать - синтаксис процедур и функций, процедур с параметром, локальные и глобальные переменные;

3. Уметь - применять синтаксис процедур и функций при составлении программы;

4. Владеть - основными навыками работы с функциями и процедурами;

8. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №34 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Ранее вы уже использовали встроенные в интерпретатор функции:

print() – выводит на печать данные, заключенные в круглые скобки;

str() – преобразует данные к строковому типу;

int() – преобразует данные к целому числу;

float() – преобразует целые числа в дробный тип;

round() – округляет число в большую по модулю сторону.

Кроме них мы можем создавать свои собственные функции для выполнения тех или иных задач. Для этого в Python предусмотрена возможность, когда некоторый повторяющийся вспомогательный алгоритм (или его фрагмент) оформляется в отдельную функцию.

Вспомогательный алгоритм – это алгоритм решения какой-либо подзадачи, который может вызываться из основного алгоритма.

В программировании вспомогательные алгоритмы называют подпрограммами. В языке Python существуют два вида подпрограмм: процедуры и функции.

Процедура – это подпрограмма, которая выполняет некоторые действия после вызова её из основной программы или другой процедуры. Каждая процедура имеет уникальное имя, может иметь произвольное количество входных параметров. При вызове процедуры указываются фактические значения параметров.

Для этого новой функции необходимо присвоить имя и описать его алгоритм. В дальнейшем, при упоминании в программе имени функции, запускается соответствующий вспомогательный алгоритм со списком входных и выходных данных. После выполнения функции работа продолжается с той команды, которая непосредственно следует за вызовом подпрограммы (функции).

Предположим, что в нескольких местах программы требуется выводить на экран сообщение об ошибке: «Ошибка в программе». Это можно сделать, например, так:

```
print ('Ошибка в программе')
```

Вставка этого оператора вывода везде, где нужно вывести сообщение об ошибке, приведет к загруженности памяти и увеличению кода программы. Если потребуется поменять текст сообщения, то нужно будет искать эти операторы вывода по всей программе. Именно для таких случаев используются функции – вспомогательные алгоритмы (подпрограммы), к которым можно обратиться с другого места программы.

Запишем функцию `error:`

```
def error():
    print ('Ошибка в программе')
    n = int (input())
if n < 0:
    error()
```

Мы ввели новую функцию error.

Как видите процедура (подпрограмма) начинается служебным словом -

def (define – «определить»).

Формальные параметры процедуры перечисляются через запятую. Операторы, входящие в тело процедуры, записываются с отступом. Процедура должна быть определена до первого её вызова.

def <имя>(<параметры>):

<операторы>

После имени функции ставятся скобки, в которых можно передавать параметры функции и двоеточие. Тело функции записывается с отступом.

Вызов процедуры осуществляется по её имени с указанием фактических параметров (аргументов).

<имя>(<аргументы>)

Для того чтобы функция заработала в другом месте программы, необходимо ее вызвать по имени (не забыв скобки).

Например:

```
error().
```

Примечание: Между формальными и фактическими параметрами должно

быть соответствие по количеству, порядку следования и типу.

Использование функций сокращает код, если какие-то операции выполняются несколько раз в разных местах программы. Иногда большую програм- му разбивают на несколько функций для удобства и упрощения, оформляя в виде функций отдельные этапы сложного алгоритма. Такой подход делает всю программу более понятной.

Функции и их аргументы

Функциям можно передавать аргументы – дополнительные данные для изменения выполняемых действий.

Предположим, что требуется многократно вывести на экран символ, например, чтобы нарисовать таблицу или разделитель. Программу, решающую эту задачу для переменной n, можно записать так:

```
n = 125 #количество раз
s = '_' #символ
```

ЗАПОМНИ

Имена функций должны состоять из строчных букв, а слова разделяться символами подчеркивания – это делает

```
while n > 0:
      print (s, end = '')
      n -= 1;
```

Обратим внимание на вызов функции print с именованным аргументом end завершающий символ (по умолчанию символ «новая строка»).

Вспомогательный алгоритм цикла вывода повторяющегося символа можно оформить в виде функции. Этой функции нужно передать аргументы – символ и число, сколько раз его повторить. Программа получается такая:

```
def pr_char(s, n): #имя функции с аргументами
      k = n
      while k > 0:
            print (s, end = '')
            k -= 1
pr_char ('-', 10)    #аргументы
```

Основная программа содержит всего одну команду – вызов функции pr_char. В скобках указаны аргументы функции, указывающие, что символ тире («-») нужно вывести 10 раз.

Глобальные и локальные переменные

Во многих случаях функции используют для обработки данные. Эти данные могут находиться в глобальных либо локальных переменных.

Локальные переменные – это переменные, определённые в процедуре, они доступны только внутри процедуры.

Глобальные переменные – это переменные, определённые в основной программе. Они доступны внутри процедуры только для чтения, а для изменения требуется объявить их в процедуре после служебного слова global.

Например:

```
def summa(a, b):
    global c
    c = a+b
summa(2)
```

print(c)

Локальные переменные передаются в функцию через аргументы, указанные в круглых скобках после имени функции. Локальные переменные находятся в «зоне видимости» только этой функции и недоступны для всей остальной программы. **Глобальные переменные** доступны во всей программе. К ним можно обратиться по имени и получить связанное с ними значение.

Пример варианта задания

1.Рассмотрим типы переменных на примере данной программы:

```
def rectangle():
    a = float(input('Ширина: '))
    b = float(input('Высота: '))
    s = a*b
    print('Площад ь: ', s)
def triangle():
    a = float(input('Основание: '))
    h = float(input('Высота: '))
    s = 0.5*a*h
    print('Площадь: ', s)
figure = input('1-прямоугольник, 2-треугольник: ')
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()
```

В этой задаче 5 переменных, где глобальная только figure. Переменные a и b из функции rectangle(), и a и h из функции triangle() – локальные. При этом локальные переменные в разных функциях являются разными переменными.

Возврат значений из функции

Функция – это вспомогательный алгоритм, который всегда возвращает в основной алгоритм значение-результат.

Каждая функция выдает определенный результат. Если даже вы не указываете на возврат значения в качестве результата, она, тем не менее, выдаст результат None (ничего).

Для того чтобы указать, чему равно значение функции, используют оператор return (англ. вернуть), после которого записывают значение-результат.

Результатом может быть число, символ, символьная строка или любой другой объект.

Описание функции

def <имя>(<параметры>):

<операторы>

return <результат>

2. Составим функцию, которая вычисляет сумму всех цифр числа (н-р, для числа 147 нужно сложить цифры: 1+4+7=12). Сложение цифр на- чинаем с последней, в нашем примере это цифра 7.

1 Чтобы получить последнюю цифру числа, нужно взять остаток от деле- ния числа на 10 (147%10=7).

2 Полученный остаток добавляем к «сумме», которая изначально равна нулю (sum=0). Теперь сумма равна 7.

3 Затем мы «отсекаем» последнюю цифру числа 147, используя оператор целочисленного деления (147//10=14).

4 Поскольку цифра 14 > 0, то мы возвращаемся в начало цикла. Цикл продолжается до тех пор, пока значение n не становится равно нулю.

5 В нашем примере мы снова отсекаем цифру 4 (14%10= 4) и добавляем ее к **сумме (4+7=11).**

6 Отделяем ее от всего числа (14//10=1).

7 Последнее однозначное число прибавляем к сумме (11+1).

Результат: 12

Таким образом, наша программа будет выглядеть так:

```
n = int(input('Введите число: '))

def digits_sum (n):
    total = 0
    while n > 0:
        total += n%10
        n = n // 10
    return total #основная программа
print (digits_sum(n))
```

3. Напишем программу, которая определяет: делится ли сумма цифр заданного числа на 3. Причем, если в полученной сумме больше одной цифры, то необходимо повторить операцию, пока сумма не будет состоять из единственной цифры.

Например, сумма цифр числа 123456789 равняется 45. Число двузначное – опять находим сумму цифр: 4 + 5 = 9. Получили девятку, которая делится на три (9%3==0). Значит, исходное число 123456789 делится на три. Исходя из этого алгоритма, получим следующую программу:

```
def sum(n):
    sum = 0
    while n>0:
        sum += n % 10
        n = n // 10
    return sum #основная программа
k = int(input('Введите число: '))
while k > 9:  #пока сумма цифр не будет одной цифрой
    k = sum(k)                    #вызываем функцию
if k%3==0:
    print ('Число делится на 3')
else:
    print ('Число не делится на 3')
```

Функцию можно вызывать не только из основной программы, но и из другой функции. Например, функция, которая находит среднее из трех различ- ных чисел (то есть число, заключенное между двумя остальными), может быть определена так:

```
def middle ( a, b, c ):
 mi = min ( a, b, c )
    ma = max ( a, b, c )
    return a + b + c - mi - ma
```

Она использует встроенные функции min и max. Идея решения состоит в том, что если из суммы трех чисел вычесть минимальное и максимальное, то получится как раз третье число.

Функция может возвращать несколько значений. Например, возврат сразу и частного, и остатка от деления двух чисел можно написать так:

```
def divmod ( x, y ):
d = x // y
m = x % y
return d, m
```

Результат функции можно записать в две различные переменные:

```
a, b = divmod ( 7, 3 )
print ( a, b ) # 2 1
```

Если указать только одну переменную, мы получим кортеж – набор элементов, который заключается в круглые скобки:

```
q = divmod ( 7, 3 )
print ( q ) # (2, 1)
```

В случаях, когда необходимо создать функцию, которая нужна в программе только один раз и выполняет несложную операцию с несколькими аргументами, используют lambda-функции. Lambda-функция – это анонимная функция, то есть не имеет своего названия, как в случая с def. Запись функции начинается со слова lambda, затем через пробел указываются аргументы функции и сразу после двоеточия указывается операция, результат которой будет возвращен функцией. Создадим такую функцию на примере умножения двух чисел:

```
multiple = lambda x, y: x * y #lambda-функция с 2-мя
аргументами
print (multiple (2, 5)) #результат 10
```

В примере выше мы присвоили lambda-функцию к переменной multiple. Хотя такую функцию можно также записать одной строкой, не присваивая к переменной:

```
print ((lambda x, y: x * y)(2, 6))
```

Одно из главных умений в программировании – это не писать один и тот же код несколько раз. Как только это происходит, нужно понимать, что этот повторяющийся кусок кода должен идти в функцию.

Практическое занятие №35 Понятие данных, больших данных. Массивы одномерные и двумерные.

Цель занятия:

1. Изучить понятия больших данных, одномерных и двумерных массивов в Python.

2. Знать операции над массивами и способы сортировки массивов;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №35 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы.

Теоретическая часть:

Основное предназначение современных компьютеров – обработка большого количества данных. При этом надо как-то обращаться к каждой из тысяч (или даже миллионов) ячеек с данными. В этой ситуации имя дают не ячейке, а группе ячеек, в которой каждая ячейка имеет собственный номер. Такая область памяти называется массивом (или таблицей).

Массив — упорядоченный набор данных, предназначенный для хранения данных одного типа, идентифицируемых с помощью одного или нескольких индексов.

Массивы используются для обработки большого количества однотипных данных, например, последовательностей (одномерные массивы) и таблиц (двухмерные массивы).

Массив характеризуется:

- размерностью;
- базовым типом элементов;
- типом индекса (может быть только порядковым типом
- множеством значений для индекса.

Элемент массива — отдельная переменная, входящая в массив.

Индекс элемента массива — номер элемента в этом массиве.

Размерность массива — количество элементов, которое содержит массив.

Массив, состоящий из некоторого ряда элементов, называется векторным или одномерным. Если массив включает несколько рядов, он называется матричным или многомерным.

[1, 2, 5, 7, 8] — векторный массив, состоящий из пяти элементов.

$$\begin{bmatrix} 3 & 5 & 3 \\ 4 & -1 & 2 \\ 2 & -3 & 1 \\ 10 & 2 & 4 \end{bmatrix}$$

- Массив является многомерным(в данном случае-двумерным)размерности 3x4 и включает 12 элементов.

Массивы могут состоять не только из чисел, но и из строк. Элементами матрицы могут являться алгебраические символы или математические функции.

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами.

Двумерные массивы или матрицами называются массивы элементов, представленные в виде прямоугольных таблиц, для которых определены правила математических действий. Матрица в Python – это прямоугольный двумерный массив, в котором данные хранятся в строках и столбцах. Матрица может хранить данные любого типа, такие как числа, строки, выражения и т. д.

Массив [“пример”, “использования”, “строк”] состоит из трёх строк, для каждой из которых выделено определённое количество информации.

Способы заполнения массива данными

- Непосредственное присваивание значений элементам.
- Заполнение массива произвольными элементами, случайными числами.
- Ввод значений элементов с клавиатуры или чтение из файла.

Количество элементов в массиве задаётся изначально и обозначается латинской буквой, например n. Доступ к элементам производится по числовому индексу, записывается как mydogs[i], где mydogs — имя масси- ва, [i] — числовой индекс элемента. Так, для обращения к символу под номером 5 запишем: mydogs[5]. В многомерном массиве также используются латинские буквы, например j и k. В этом случае доступ к массиву осуществляется по записи вида inf[5,6], где inf — имя массива, числа обозначают, что запрошенный элемент находится на пересечении строки 5 и столбца 6. Доступ к массиву произвольный, поэтому можно обращаться к элементам под индексами 0, 33, 2020, и независимо от того, какой из эле- ментов мы запросим, он сразу появится.

ДЕЙСТВИЯ НАД МАССИВАМИ И ИХ ЭЛЕМЕНТАМИ

Для того чтобы выбрать конкретный элемент массива, надо после идентификатора массива указать его индекс или индексы, поэтому элементы массива часто называют индексированными переменными. Элемент одномерного массива имеет один индекс, двухмерного — два и т. д. Над элементами массива могут производиться те же действия, что и над простыми переменными того же типа.

Если два массива и более описаны одинаково, то с ними можно проводить следующие операции:

Проверка на равенство. *A*=*B*. Результатом является значение «истина», если каждый элемент массива *A* равен соответствующему элементу массива *B*.

Проверка на неравенство. *A*<>*B*. Результатом является значение «истина», если значение хотя бы одного элемента массива *A* не равно значению соответствующего элемента массива *B*.

Присваивание. *A*:=*B*. Все значения элементов массива *B* присваиваются элементам массива *A*. Значения элементов массива *B* остаются без изменения.

Инициализация массива. Инициализация — это присваивание каждому элементу массива какого-либо значения.

Ввод и вывод элементов массива. Массив выводится на экран компьютера.

Копирование элементов из одного массива в другой. Для выполнения этого действия необходимо, чтобы массивы, участвующие в данной операции, были описаны одинаково.

Поиск в массиве элементов, удовлетворяющих некоторым условиям. Пусть нам зада- но некоторое условие, по этому условию намнужно найти элементы в массиве.

Поиск минимального (максимального) эле- мента. Осуществляется отбор минимального или максимального элемента среди всех эле- ментов массива.

Вычисление суммы (произведения) эле- ментов массива. В результате получаем одно число.

Перестановка элементов массива. Осуществляется с использованием вспомогательной переменной того же типа, что и базовый тип массивов.

Суммирование элементов матриц (двух- мерного массива) по строкам или столбцам. В результате получаем одномерный массив.

Поиск минимального (максимального) эле- мента строки (столбца). В результате выводим минимальный или максимальный элемент по строке или столбцу.

Сортировка массивов. Изменение массива в соответствии с заданным условием.

СОРТИРОВКА МАССИВА

Сортировкой называется процесс упорядочивания набора данных одного типа по возрастанию или убыванию значения какого-либо признака.

Алгоритм сортировки — система последовательных операций для упорядочивания элементов в списке.

ПАРАМЕТРЫ ОЦЕНКИ АЛГОРИТМОВ

Время сортировки

Основной параметр, характеризующий быстродействие алгоритма.

Память

Ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. При оценке используемой памяти не будет учитываться место, которое занимает исходный массив, и не зависящие от входной последовательности затраты, например на хранение кода программы.

Устойчивость

Устойчивая сортировка не меняет взаимного расположения равных элементов. Такое свойство может быть очень полезным, если элементы состоят из нескольких полей, а сортировка происходит по одному из них.

Естественность поведения

Определяется эффективностью метода при обработке уже отсортированных или частично отсортированных данных. Алгоритм ведёт себя естественно, если учитывает эту характеристику входной последовательности и работает лучше.

Сортировка массива может проходить по возрастанию или убыванию. Сортировка строк часто ранжируется по алфавиту. Различают обменную сортировку (пузырьком), сортиров- ку выбором, сортировку вставками, сортировку слиянием.

Сортировка обмена (пузырьком)

Идея метода: сортировка массива от элемента a[0] к элементу a[i]с попарным изменением местами двух элементов массива при определённых условиях.

Первый проход

1) Проходим снизу вверх (слева направо) по массиву.

2) Просматриваем пары соседних элементов.

3) Меняем местами элементы, если они не удовлетворяют условию сортировки.

После первого прохода по массиву вверху оказывается самый лёгкий элемент — отсюда аналогия с пузырьком.

Второй и последующие проходы

1) Следующий проход делаем до второго сверху элемента, таким образом, второй по величине элемент поднимается на правильную позицию.

2) Делаем проходы по все уменьшающейся нижней части массива до тех пор, пока в ней не останется только один элемент.

На этом сортировка заканчивается, т. к. последовательность упорядочена по возрастанию.

Сортировка выбором

Идея метода: создавать отсортированную последовательность путём присоединения к ней одного элемента за другим в правильном по- рядке. Строим готовую последовательность, начиная с левого конца массива.

Алгоритм состоит из n последовательных шагов, начиная от нулевого и заканчивая (n – 1)-м.

1) На первом шаге просмотрим весь данный массив и найдём наименьший элемент — k.

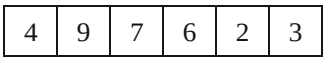
2) Меняем первый элемент и k-й (наименьший). Таким образом, на первом шаге по- лучим, что на месте нулевого элемента a[0] стоит наименьший.

3) На втором шаге находим наименьший эле- мент среди оставшихся a[1]...a[n – 1] и меня- ем его с a[1].

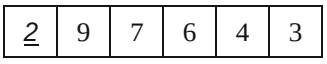
4) На i-м шаге выбираем наименьший из элементов a[i]...a[n – 1] и меняем его места- ми с a[i].

5) Продолжаем шаги до тех пор, пока не по- меняем последние элементы.

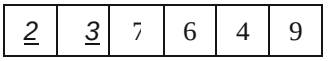
Последовательность шагов при n = 5 изображена на рисунке ниже.



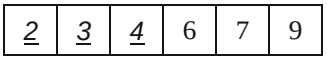
исходная последовательность



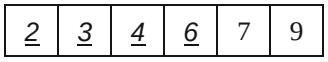
шаг 1: 2 4



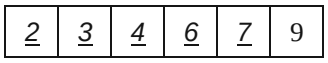
шаг 2: 3 9



шаг 3: 4 7



шаг 4: 6 6



шаг 5: 7 7

Вне зависимости от номера текущего шага i, последовательность a[0]...a[i] (выделена на рисунке курсивом) является упорядоченной. Таким образом, на (n – 1)-м шаге вся последовательность оказывается отсортированной.

Быстрая сортировка (вставками)

Этот вид сортировки является наиболее при- меняемым и эффективным.

Идея метода: разделение массива на под- массивы и сортировка этих подмассивов.

1) Выбираем из массива некоторый опорный элемент a[i].

2) Запускаем процедуру разделения массива, которая перемещает все ключи, меньшие либо равные a[i], влево от него, а все ключи, большие либо равные a[i], — вправо. Теперь массив состоит из двух подмножеств, причём левое меньше правого либо равно ему.

d *a*[*i*]	*a*[*i*]	□ *a*[*i*]

3)Если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру. Производим эту операцию для обоих подмассивов.

4)Продолжаем до тех пор, пока не получим полностью отсортированный массив.

Сортировка слиянием

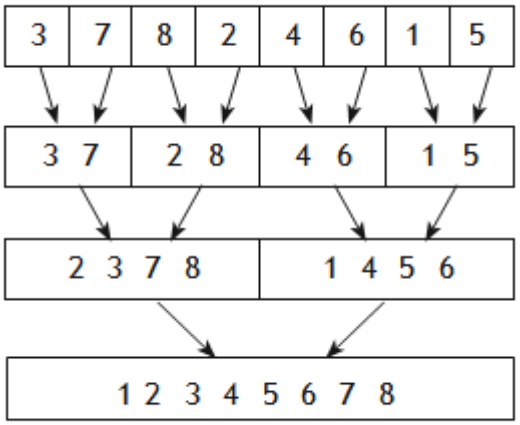
Идея метода: вместо разделения по опор- ному элементу, как в принципе сортировки вставками, массив просто делится пополам.

Массив разделён до последовательностей длины один

Слияние до упорядоченных пар

Слияние пар в упорядоченные четвёрки

Слияние четвёрок в общий массив



Часто для работы с одномерными массивами используются списки.

Список (list) – это структура данных для хранения объектов различных типов.

Списки являются упорядоченными последовательностями, которые состоят из различных типов данных, заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятой.

Для работы с матрицами в Python также используются списки. Каждый элемент списка-матрицы содержит вложенный список.

Таким образом, получается структура из вложенных списков, количество которых определяет количество столбцов матрицы, а число элементов внутри каждого вложенного списка указывает на количество строк в исходной матрице.

Примеры представления одномерного и двумерного массива в Python :

Одномерные массивы


```
A[0] = -5,  A[1] = -2, . . . A[9] = -3
```

Индексом может быть не только целое число, но и целое значение переменной или арифметического выражения.

Например: **A[2*i-1] = -2** (при **i=1**).

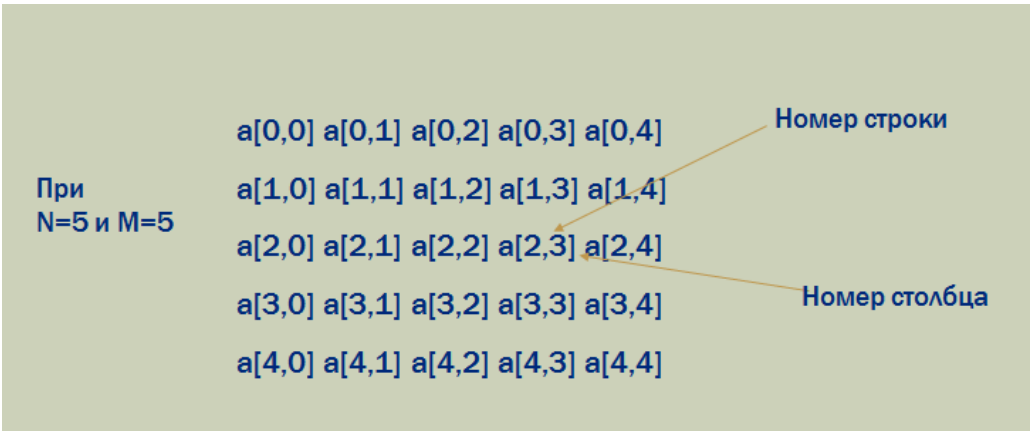
Перед использованием в программе массив необходимо создать. Обращение к несуществующему элементу вызовет ошибку.

Количество элементов в массиве определяется с помощью функции **len** (length – «длина») . Например: N = len(A)

Двумерные массивы

```
a = [[1, 2, 3], [4, 5, 6]]
```

Где n*m – размер матрицы, в которой каждый элемент характеризуется номером строки-i и номером столбца-j.



Задания для самостоятельного выполнения

Задание №1

Создайте конспект теоретического материала по плану

1)Что называют массивом и для чего они используются.

2) Как характеризуется массивы.

3) Что понимают под именем, индексом и размерностью массива.

4) Виды массива

5) Какие величины могут быть элементами массива

6) Способы заполнения массива

7)Действия над массивами и их элементами.

8) Виды сортировки массива.

9)Приведите пример записи одномерного и двумерного массива в Python

10) Какая функция определяет длину массива.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
4. Список используемых источников

5. Выводы и предложения
6. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки

1. Что такое массивы?
2. Как создаются списки?
3. Назовите идеи методов сортировок массива

Практическое занятие №36 Работа с одномерными массивами. Создание и вывод одномерного массива

Цель занятия:

1. Изучить способы создания и вывода одномерных массивов в Python;
2. Владеть способами по созданию одномерных массивов и вывода их на экран;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №36 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Способы создания одномерного массивов

1 способ. Создание массива указанием значений элементов

Массив создается перечислением элементов через запятую в квадратных скобках.

```
A = [1, -2, -3, 5, 7]
```

Если все элементы одинаковые, используется следующий оператор.

```
# массив из 5 элементов
# заполненный нулями
A = [0] * 5
```

Получение списка через присваивание конкретных значений.

Так выглядит в коде Python пустой список:

s = [] # Пустой список

Примеры создания списков со значениями:

```
l=[5,75,-4,7,-51]# список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8]# список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"]# список из строк
l=["Москва", "Иванов", 12, 124]   # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]]   # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Списки можно складывать (конкатенировать) с помощью знака «+»:

```
l=[1, 3]+[4,23]+[5]
print('l=[1, 3]+[4,23]+[5] =',l)
```

Результат:

```
>>>
l=[1, 3]+[4,23]+[5] = [1, 3, 4, 23, 5]
>>>
```

2 способ. Ввод с клавиатуры (при небольшом количестве элементов)

```
N = 5                       # размер массива в переменной
B = [0] * N                 # заполнение массива нулями
print ("Введите", N, "элементов массива:")
for i in range(N):          # перебор индексов
    B[i] = int(input())     # ввод числа с клавиатуры
```

Можно в цикл добавить подсказку с индексом вводимого элемента.

```
for i in range(N):                      # перебор индексов
    print ("B[", i, "] = ", end="")     # вывод подсказки
    B[i] = int(input())                 # ввод числа
```

На экране:

```
B[ 0 ] = 1
B[ 1 ] = 2
B[ 2 ] = 3
B[ 3 ] = 4
B[ 4 ] = 5
```

Генераторы списков.

В Python создать список можно также при помощи генераторов.

Первый способ.

Сложение одинаковых списков заменяется умножением:

Список из 10 элементов, заполненный единицами

l = [1]*10

Пример 1.

l = [i for i in range(10)]

```
Генераторы списков.
Первый способ.
l = [1]*10:
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Второй способ. Пример 1.
l = [i for i in range(10)]:
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Примеры использования генераторов списка.

Пример 1.

Заполнить список квадратами чисел от 0 до 9, используя генератор списка.

Решение:

l = [i*i for i in range(10)]

Пример 2.

Заполнить список числами, где каждое последующее число больше на 2.

l = [(i+1)+i for i in range(10)]

print(l)

```
Заполнить список квадратами чисел от 0 до 9, используя генератор списка.
l = [i*i for i in range(10)]:
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Заполнить список числами, где каждое последующее число больше на 2.
l = [(i+1)+i for i in range(10)]:
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

3 способ. Вычисление элементов по формуле (функция от индекса)

```
N = 5                 # размер массива в переменной
C = [0] * N           # заполнение массива нулями
for i in range(N):    # перебор индексов
    C[i] = i**2       # индекс в квадрате
print (C)             # вывод массива
```

На экране:

```
[0, 1, 4, 9, 16]
```

4 способ. Заполнение случайными числами

Функция **randint(a, b)** создаёт случайное целое число из отрезка [a, b].

```
N = 5                          # размер массива в переменной
D = [0] * N                    # заполнение массива нулями
from random import randint     # подключение функции randint
for i in range(N):             # перебор индексов
    D[i] = randint(-5, 5)      # случайные числа от -5 до 5
print (D)                      # вывод массива
```

Возможный результат на экране:

```
[0, -4, -2, 1, 5]
```

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

random.randint(A, B) - случайное целое число N, A ≤ N ≤ B.

random.random() - случайное число от 0 до 1.

Случайные числа в списке:

10 чисел, сгенерированных случайным образом в диапазоне (10,80)

from random import randint

l = [randint(10,80) for x in range(10)]

10 чисел, сгенерированных случайным образом в диапазоне (0,1)

l = [random() for i in range(10)]

```
from random import *
l = [randint(10,80) for i in range(10)]
print('10 чисел, сгенерированных случайным образом в диапазоне (10,80).')
print('l = [randint(10,80) for x in range(10)]:')
print(l)
print()

l = [random() for i in range(10)]
print('10 чисел сгенерированных в диапазоне от 0 до 1.')
print('l = [random() for i in range(10):')
for i in range(len(l)):
    print ('{:.2f}'.format(l[i]), end = " " )
```

```
10 чисел, сгенерированных случайным образом в диапазоне (10,80).
l = [randint(10,80) for x in range(10)]:
[70, 33, 79, 61, 34, 27, 11, 55, 52, 31]

10 чисел сгенерированных в диапазоне от 0 до 1.
l = [random() for i in range(10):
0.66 0.97 0.87 0.57 0.54 0.83 0.57 0.65 0.04 0.07
```

Ввод списка (массива) в языке Python.

Для ввода элементов списка используется цикл for и команда range ():

fori in range(N):

x[i] = int(input())

Более простой вариант ввода списка:

x = [int(input()) for i in range(N)]

```
print('Ввод списка. Пример 1:')
x=[]
for i in range(4):
    x.append(int(input()))
print(x)

x=[]
print('Ввод списка. Пример 2:')
x = [ int(input()) for i in range(4) ]
print(x)
```

```
Ввод списка. Пример 1:
45
4
85
2
[45, 4, 85, 2]
Ввод списка. Пример 2:
4
5
7
8
[4, 5, 7, 8]
```

Функция int здесь используется для того, чтобы строка, введенная пользователем, преобразовывалась в целые числа.

Способы вывода одномерных массивов на экран

1 способ. Весь массив выводится как один объект в квадратных скобках, элементы разделяются запятыми.

```
print (A)
```

На экране:

```
[1, 2, 3, 4, 5]
```

2 способ. Вывод элементов с помощью цикла в одной строке через пробел.

```
for i in range(len(A)):
    print (A[i], end=" ")
print()  # переход на новую строку
```

На экране:

```
1 2 3 4 5
```

3 способ. Вывод элементов с помощью цикла в столбик.

```
for i in range(len(A)):
    print (A[i])
```

На экране:

```
1
2
3
4
5
```

4 способ. Вывод элементов с помощью цикла в столбик с указанием индексов.

```
for i in range(len(A)):
    print ("A[", i, "]=", A[i])
```

На экране:

```
A[ 0 ]= 1
A[ 1 ]= 2
A[ 2 ]= 3
A[ 3 ]= 4
A[ 4 ]= 5
```

Вывод целого списка (массива):

print (L)

Поэлементный вывод списка (массива):

for i inrange(N):

  print(L[i], end = " ")

```
'''
Вывод целого списка (массива)
[1, 56, 6, 3, 6, 7, 3, 37, 7, 37, 37]

Поэлементный вывод списка (массива)
1 56 6 3 6 7 3 37 7 37 37
```

2. Методы списков.

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет перед i-ым элементом значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до

	end)
list.count(x)	Возвращает количество элементов со значением x
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Ниже приведена программа, демонстрирующая методы работы списков.

```
a=[0,2,2,2,4] #список a
b=[5,6,7,2,9] #список b
print('Исходный список a:',a)
print('Исходный список b:',b)
x=99
y=5

a.append(x)
print('a.append(x):',a)

a.extend(b)
print('a.extend(b):',a)

a.insert(3,x)
print('a.insert(3,x):',a)

a.remove(x)
print('a.remove(x):',a)

print('a.pop(5):',a.pop(5))
print(a)

print('a.index(y,0,len(a)):',a.index(y,0,len(a)))

print('a.count(2):',a.count(2))

a.reverse()
print('a.reverse():',a)

z=a.copy()
print('z=a.copy():',z)

z.clear()
print('z.clear():')
print('z =',z)
```

Пример программы на Python

```
Исходный список a: [0, 2, 2, 2, 4]
Исходный список b: [5, 6, 7, 2, 9]
a.append(x): [0, 2, 2, 2, 4, 99]
a.extend(b): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.insert(3,x): [0, 2, 2, 99, 2, 4, 99, 5, 6, 7, 2, 9]
a.remove(x): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.pop(5): 99
[0, 2, 2, 2, 4, 5, 6, 7, 2, 9]
a.index(y,0,len(a)): 5
a.count(2): 4
a.reverse(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z=a.copy(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z.clear():
z = []
```

Результат выполнения программы

Примерный вариант задания

1. Из массива X длиной n, среди элементов которого есть положительные, отрицательные и равные нулю, сформировать новый массив Y, взяв в него только те элементы из X, которые больше по модулю заданного числа M. Вывести на экран число M, данный и полученные массивы.

Решение:

```
n=int(input('Введите длину массива\n'))
m=int(input('Введите число M\n'))
x=[]
y=[]
for i in range(n):
    print('Введите ',i,'элемент:')
    x.append(int(input()))
for i in range(n):
    if abs(x[i])>m:
        y.append(x[i])
print('Введённое число M:',m)
print('Массив X:',x)
print('Массив Y:',y)
```

```
Введите длину массива
5
Введите число M
20
Введите  0 элемент:
21
Введите  1 элемент:
22
Введите  2 элемент:
5
Введите  3 элемент:
6
Введите  4 элемент:
8
Введённое число M: 20
Массив X: [21, 22, 5, 6, 8]
Массив Y: [21, 22]
```

2. В массиве целых чисел все отрицательные элементы заменить на положительные. Вывести исходный массив и полученный.

Решение:

```
n=int(input('Введите длину массива:'))
a=[]
for i in range(n):
    print('Введите',i,'элемент:')
    a.append(int(input()))
print('Исходный массив:',a)
for i in range(n):
    if a[i]<0:
        a[i]=-a[i]
print('Полученный массив:',a)
```

```
Введите длину массива:5
Введите 0 элемент:
-5
Введите 1 элемент:
-4
Введите 2 элемент:
-6
Введите 3 элемент:
5
Введите 4 элемент:
-7
Исходный массив: [-5, -4, -6, 5, -7]
Полученный массив: [5, 4, 6, 5, 7]
```

Выводы и предложения о проделанной работе

Содержание отчета:

7. Наименование практического занятия
8. Цель занятия
9. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
10. Список используемых источников
11. Выводы и предложения
12. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки

4. Что такое массивы?
5. Как создаются списки?
6. Как происходит ввод списка (массива)?
7. Как происходит вывод списка (массива)?

Практическое занятие №37 Работа с двумерными массивами. Создание и вывод двумерного массива.

Цель занятия:

1. Изучить двумерные массивы в Python.

2. Знать - способ описания двумерного массива, способы ввода элементов двумерного массива;

3. Уметь - вводить массивы, получать списки через присваивание конкретных значений, применять функции;

4. Владеть - основными навыками создания и вывода двумерных массивов;

5.Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №37 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Матрицами называются массивы элементов, представленные в виде прямоугольных таблиц, для которых определены правила математических действий. Элементами матрицы могут являться числа, алгебраические символы или математические функции.

Для работы с матрицами в Python также используются списки. Каждый элемент списка-матрицы содержит вложенный список.

Таким образом, получается структура из вложенных списков, количество которых определяет количество столбцов матрицы, а число элементов внутри каждого вложенного списка указывает на количество строк в исходной матрице.

Способы создания двумерного массивов

1) Через функцию range(n):

Вывод: [0, 1, 2, 3, 4]

```
a = range(5)
print (a)
```

2) Создание массива заполненного нулями с помощью цикла:

Ввести с клавиатуры количество строк n и количество столбцов m. Необходимо создать список размером n×m, заполненный нулями

3) Путём создания пустого списка(n раз добавить новый элемент, который является списком-строкой):

```
n = 3
m = 4
a = []
for i in range(n):
    a.append([0]*m)
    print(a)
```

Вывод: [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

4) Путём создания вложенного списка (двумерного массива) по значениям:

```
a = [[1,2,3,4],[5,6,7,8]]
for row in a:
    for elem in row:
        print(elem, end = ' ')
    print()
```

Вывод:

```
1 2 3 4
5 6 7 8
```

Способы вывода двумерных массивов на экран

1) С помощью функции print():

```
a = [[-1, 0, 1],
     [-1, 0, 1],
     [0, 1, -1],
     [1, 1, -1]]
print(a)
```

```
b = [[-1, 0, 1], [-1, 0, 1],
    [0, 1, -1], [1, 1, -1]]
print(b)
```

Вывод: [[-1, 0, 1], [-1, 0, 1], [0, 1, -1], [1, 1, -1]]

2) С помощью генератора чисел:

```
import random
n, m = 3, 3
a = [[random.randint(1, 10) for j in range(m)] for i in
range(n)]
print(a)
```

1. Создание списка

Пусть даны два числа: количество строк n и количество столбцов m. Необходимо создать список размером n×m, заполненный нулями.

Очевидное решение оказывается неверным:

A = [[0] *m]*n

В этом легко убедиться, если присвоить элементу A[0][0] значение 1, а потом вывести значение другого элемента A[1][0] — оно тоже будет равно 1! Дело в том, что [0] * m возвращает ссылку на список из m нулей. Но последующее повторение этого элемента создает список из n элементов, которые являются ссылкой на один и тот же список (точно так же, как выполнение операции B = A

для списков не создает новый список), поэтому все строки результирующего списка на самом деле являются одной и той же строкой.

Таким образом, двумерный список нельзя создавать при помощи операции повторения одной строки.

Первый способ.

Сначала создадим список из n элементов (для начала просто из n нулей). Затем сделаем каждый элемент списка ссылкой на другой одномерный список из m элементов:

```
A = [0] * n
for i in range(n):
    A[i] = [0] * m
```

```
n=3
m=3
A=[0]*n
for i in range(n):
    A[i]=[0]*m
print('A:',A)
```

```
A: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
```

Второй способ.

Создать пустой список, потом n раз добавить в него новый элемент, являющийся списком-строкой:

```
A = []
for i in range(n):
    A.append([0] * m)
```

```
n=3
m=4
A = []
for i in range(n):
    A.append([0]*m)
print(A)
```

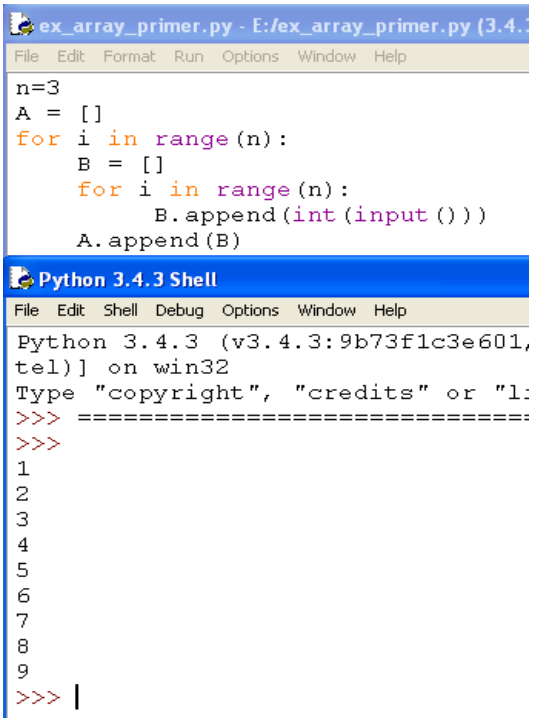
```
A: [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
>>>
```

2. Ввод вложенного списка (двумерного массива)

Пример:

n=5

```
A = []
for i in range(n):
    b = input()
        for i in range(len(row)):
            row[i] = int(row[i])
A.append(row)
```



3. Вывод вложенного списка (двумерного массива)

Для обработки и вывода списка как правило используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
for i in range(n):
    for j in range(n]):
        print(A[i][j], end = ' ')
    print()
```

```
n=3
A = []
#ввод массива
for i in range(n):
    B = []
    for i in range(n):
        B.append(int(input()))
    A.append(B)
#вывод массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end=' ')
    print()
```

```
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
```

То же самое, но циклы не по индексу, а по значениям списка:

```
for row in A:
    for elem in row:
        print(elem, end = ' ')
    print()
```

```
A=[[1,2,3,4],[5,6,7,8]]
for row in A:
    for elem in row:
        print(elem, end = ' ')
    print()
```

Python 3.7.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.1 (v3.7.1:260ec2c36a
l)] on win32
Type "help", "copyright", "cred
>>>
========== RESTART: K
1 2 3 4
5 6 7 8
>>>
```

Для вывода одной строки можно воспользоваться методом join. Использовав этот метод в цикле for можно

```
for row in A:
    print(' '.join(list(map(str, row))))
```

4. Обработка и вывод вложенных списков

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк,

каждый элемент которого является в свою очередь списком, например, чисел. Например, создать числовую таблицу из двух строк и трех столбцов можно так:

A = [[1, 2, 3], [4, 5, 6]]

Здесь первая строка списка A[0] является списком из чисел [1, 2, 3].

Тоесть

A[0][0]= 1,

A[0][1]= 2,

A[0][2]= 3,

A[1][0]=4,

A[1][1]=5,

A[1][2]=6.

Используем два вложенных цикла для подсчета суммы всех чисел в списке:

```
S = 0
for i in range(len(A)):
    for j in range(len(A[i])):
        S += A[i][j]
```

Или то же самое с циклом не по индексу, а по значениям строк:

```
S = 0
for row in A:
    for elem in row:
        S += elem
```

```
A=[[1, 2, 3,4],[ 5, 6,7,8]]
#вывод при помощи цикла for и метода join
print('Массив A:')
for i in A:
    print(' '.join(list(map(str, i))))
#Пример 1. Подсчёт суммы всех элементов
S = 0
for i in range(len(A)):
    for j in range(len(A[i])):
        S += A[i][j]
print('Пример 1. Сумма элементов:',S)
#Пример 2. Подсчёт суммы всех элементов
S = 0
for row in A:
    for elem in row:
        S += elem
print('Пример 2. Сумма элементов:',S)
```

```
Массив A:
1 2 3 4
5 6 7 8
Пример 1. Сумма элементов: 36
Пример 2. Сумма элементов: 36
```

5. Пример сложной обработки массива

Пусть дана квадратная матрица из n строк и n столбцов. Необходимо элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присвоить значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2. То есть получить такой массив (пример для n==3):

0 1 1

2 0 1

2 2 0

Рассмотрим несколько способов решения этой задачи.

Первый способ.

Элементы, которые лежат выше главной диагонали – это элементы A[i][j], для которых i<j, а для элементов ниже главной диагонали i>j. Таким образом, мы можем сравнивать значения i и j и по ним определять значение A[i][j]. Получаемследующийалгоритм:

```
for i in range(n):
    for j in range(n):
        if i< j:
            A[i][j] = 0
        elifi> j:
            A[i][j] = 2
        else:
            A[i][j] = 1
```

Ниже приведён пример программы, в котором квадратная матрица 3x3 заполняется элементами со значением 9, а затем элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присваивается значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2.

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(n):
        if i < j:
            A[i][j] = 1
        elif i > j:
            A[i][j] = 2
        else:
            A[i][j] = 0
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Второй способ.

Данный алгоритм плох, поскольку выполняет одну или две инструкции if для обработки каждого элемента. Если мы усложним алгоритм, то мы сможем обойтись вообще без условных инструкций.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

for i in range(n):

A[i][i] = 1

Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам A[i][j] для j=i+1, ..., n-1. Здесь нам понадобятся вложенные циклы:

foriinrange(n):

forjinrange(i + 1, n):

A[i][j] = 0

Аналогично присваиваем значение 2 элементам A[i][j] для j=0, ..., i-1:

```
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
```

Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:

```
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
    A[i][i] = 1
    for j in range(i + 1, n):
        A[i][j] = 0
```

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
    A[i][i] = 0
    for j in range(i + 1, n):
        A[i][j] = 1
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Третий способ.

А вот такое решение использует операцию повторения списков для построения очередной строки списка. i-я строка списка состоит из i чисел 2, затем идет одно число 1, затем идет n-i-1 число 0:

```
for i in range(n):
```

A[i] = [2] * i + [1] + [0] * (n - i - 1)

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    A[i] = [2] * i + [0] + [1] * (n - i - 1)
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Примерный вариант задания

1. Дан двумерный массив размером 3x3. Определить максимальное значение среди элементов третьего столбца массива; максимальное значение среди элементов второй строки массива. Вывести полученные значения.

Решение:

```
n=3
a=[]
for i in range(n):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент')
        b.append(int(input()))
    a.append(b)
#вывод массива
for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

#максимальное значение среди элементов третьего столбца
maximum=a[0][2]
for i in range(n):
    for j in range(n):
        if maximum<a[i][2]:
            maximum=a[i][2]
print('Максимальный в 3 столбце:',maximum)

#максимальное значение среди элементов второй строки
maximum=a[1][0]
for i in range(n):
    for j in range(n):
        if maximum<a[1][j]:
            maximum=a[1][j]
print('Максимальный во второй строке:',maximum)
```

```
Введите [ 0 , 0 ] элемент
1
Введите [ 0 , 1 ] элемент
2
Введите [ 0 , 2 ] элемент
3
Введите [ 1 , 0 ] элемент
4
Введите [ 1 , 1 ] элемент
5
Введите [ 1 , 2 ] элемент
6
Введите [ 2 , 0 ] элемент
7
Введите [ 2 , 1 ] элемент
8
Введите [ 2 , 2 ] элемент
9
1 2 3
4 5 6
7 8 9
Максимальный в 3 столбце: 9
Максимальный во второй строке: 6
```

2. Дан двумерный массив размером mxn. Сформировать новый массив заменив положительные элементы единицами, а отрицательные нулями. Вывести оба массива.

Решение:

```
m=int(input('Введите количество строк'))
n=int(input('Введите количство столбцов'))
a=[]
for i in range(m):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент')
        b.append(int(input()))
    a.append(b)
#вывод массива
print('Исходный массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

for i in range(m):
    for j in range(n):
        if a[i][j]<0: a[i][j]=0
        elif a[i][j]>0: a[i][j]=1

#вывод массива
print('Изменённый массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

```
Введите количество строк:3
Введите количество столбцов:4
Введите [ 0 , 0 ] элемент
-1
Введите [ 0 , 1 ] элемент
5
Введите [ 0 , 2 ] элемент
4
Введите [ 0 , 3 ] элемент
-5
Введите [ 1 , 0 ] элемент
-2
Введите [ 1 , 1 ] элемент
-1
Введите [ 1 , 2 ] элемент
0
Введите [ 1 , 3 ] элемент
4
Введите [ 2 , 0 ] элемент
-5
Введите [ 2 , 1 ] элемент
4
Введите [ 2 , 2 ] элемент
5
Введите [ 2 , 3 ] элемент
-5
Исходный массив:
-1 5 4 -5
-2 -1 0 4
-5 4 5 -5
Полученный массив:
0 1 1 0
0 0 0 1
0 1 1 0
```

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;
2. Цель занятия;
3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;
4. Список используемых источников;
5. Выводы и предложения;
6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Дайте определение понятию «матрицы».
2. Как создаются списки?
3. Как осуществляется вывод вложенного списка?
4. Как создать вложенный список (синтаксис)?

Практическое занятие №38 Обработка массивов

***Цель занятия*:**

1. Изучить способов обработки массивов в Python;
2. Овладеть основными навыками создания программ обработки массивов;
3.Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №38 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретический материал.

Обработка массивов

1) Сложение элементов массива:

```
matrix = [[1, 2, 3],
          [4, 5, 6]]
s = 0
for row in matrix:
```

2) Произведение элементов массива:

```
matrix = [[1, 2, 3],
          [4, 5, 6]]
n=2
m=3
p = 1
```

3) Смена мест столбцов матрицы (2 и 4 столбца):

```
a = [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]]
n = 2
for i in range(n):
    c = a[i][2]
    a[i][2] = a[i][4]
    a[i][4] = c
```

4) Смена мест строк матрицы (2 и 3 строки):

```
a = [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10],
     [11, 12, 13, 14, 15],[16, 17, 18, 19, 20]]
m = 5
for j in range(m):
    c = a[2][j]
    a[2][j] = a[3][j]
    a[3][j] = c
    print(a)
```

Вывод:
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [16, 12, 13, 14, 15], [11, 17, 18, 19, 20]]
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [16, 17, 13, 14, 15], [11, 12, 18, 19, 20]]

5) Удаление k строки матрицы:

```
a=[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10],
   [11, 12, 13, 14, 15],[16, 17, 18, 19, 20]]
n=2
k=0
m=5
for i in range (k,n-1):
    for j in range (m):
        a[i][j]=a[i+1][j]
    print(a)
```

Вывод: [[6, 7, 8, 9, 10], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20]]

Размер исходного массива уменьшается

6) Удаление k столбца матрицы:

```
a=[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10],
   [11, 12, 13, 14, 15],[16, 17, 18, 19, 20]]
n=2
```

7) Сортировка массива

```
# Сортировка элементов массива (метод пузырька по неубыванию)
N = 10; A = [0]*N           # создание массива
from random import randint  # подкл. генератора случайных чисел
for i in range(N):          # заполнение массива
    A[i] = randint(0, 99)   # случайными числами от 0 до 99
print (A)                   # вывод массива
for k in range(1, N-1):     # номер прохода
    for i in range(N-k):    # просмотр за один проход
        if A[i] > A[i+1]:   # если соседние неупорядочены
            A[i], A[i+1] = A[i+1], A[i]    # меняем их местами
    print (A)               # вывод текущих значений массива
```

```
[92, 47, 84, 49, 24, 73, 98, 19, 65, 90]
[47, 84, 49, 24, 73, 92, 19, 65, 90, 98]
[47, 49, 24, 73, 84, 19, 65, 90, 92, 98]
[47, 24, 49, 73, 19, 65, 84, 90, 92, 98]
[24, 47, 49, 19, 65, 73, 84, 90, 92, 98]
[24, 47, 19, 49, 65, 73, 84, 90, 92, 98]
[24, 19, 47, 49, 65, 73, 84, 90, 92, 98]
[19, 24, 47, 49, 65, 73, 84, 90, 92, 98]
[19, 24, 47, 49, 65, 73, 84, 90, 92, 98]
```

Примерный вариант задания

1. В одномерном числовом массиве D длиной n вычислить сумму элементов с нечетными индексами. Вывести на экран массив D, полученную сумму.

2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

3. Дана целая квадратная матрица n-го порядка. Определить, является ли она магическим квадратом, т. е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы.

Выводы и предложения о проделанной работе

Содержание отчета:

1. Наименование практического занятия;
2. Цель занятия;

3. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»;
4. Список используемых источников;
5. Выводы и предложения;
6. Дата и подпись курсанта и преподавателя.

Вопросы для самопроверки

1. Какой метод используется для сортировки массива?
2. Как создаются массивы?
3. Какие виды обработки массивов существуют?
4. Как создать вложенный список (синтаксис)?

Раздел 5 Алгоритмизация и программирование. Аналитика и визуализация данных на Python

Тема 5.6 Графика в Python. Визуализация данных

Практическое занятие № 39 Использование процедур в графике.

Цель занятия:

1. Изучить понятие графики Python;
2. Овладеть операциями и методами над процедурами в графике ;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №39 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

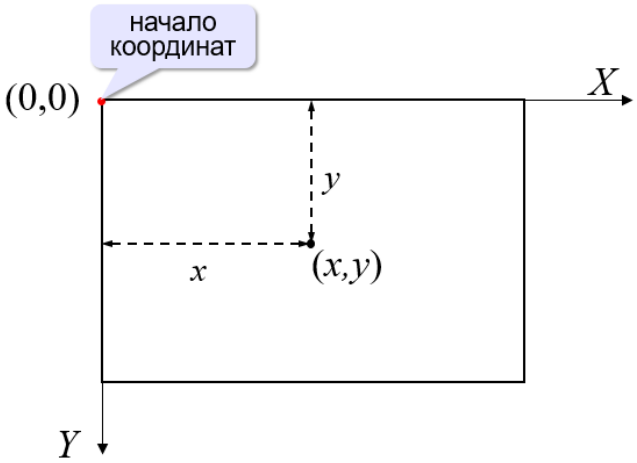
2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Подпрограммы прежде всего необходимы в ситуации, когда в разных частях программы необходимо выполнять одни и те же действия несколько раз. В таком случае повторяемые операторы оформляются в виде функции или процедуры, к которой можно обращаться и вызывать ее выполнение из разных частей программы.

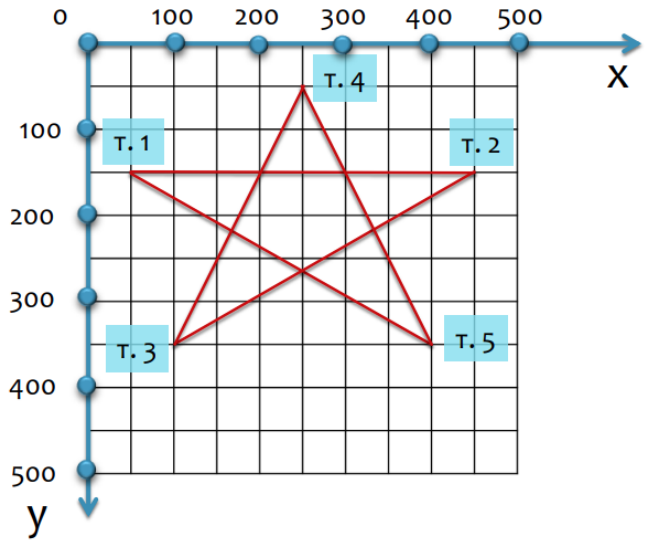
В python существует два вида реализации подпрограмм: функции в роли процедуры и функции в классическом понимании.

Функция в роли процедуры призвана не возвратить значение в основную программу, а вывести его, либо выполнить какие-либо действия с глобальными переменными, при этом не возвращая полученные значения основной программе (не используя ключевое слово return).



Верхняя левая точка экрана имеет координату (0,0), ось OX направлена вправо, ось OY – вниз, т.е. чем ниже на экране расположена точка, тем больше ее координата по оси OY.

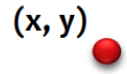
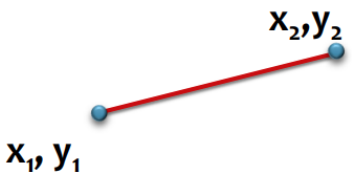
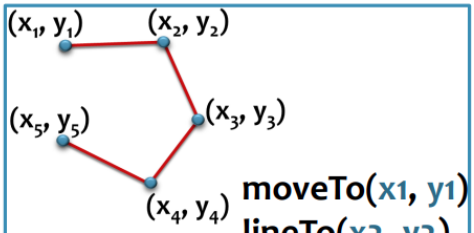
Задание №1 Определить координаты точек, изображённых на рисунке:

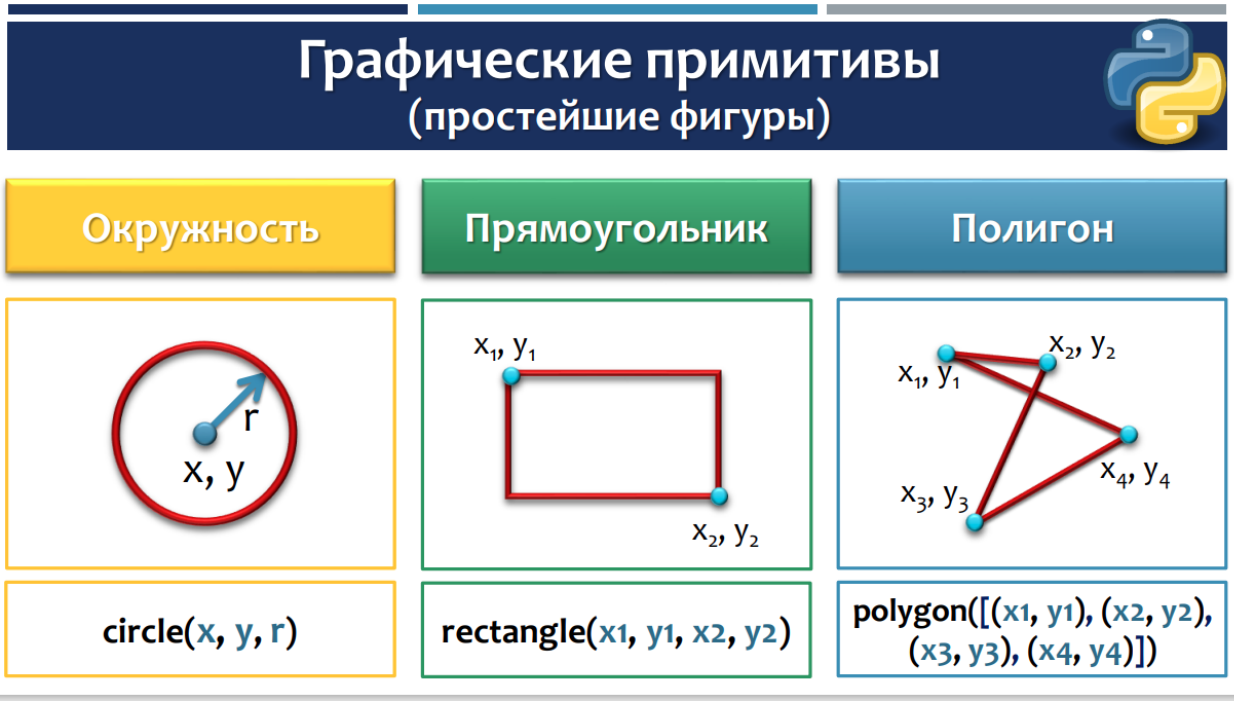


При построении изображения и его оформлении возможно использование нескольких цветов.

Цвет линий:	**penColor("red")**
Толщина линий:	**penSize(2)**
Цвет заливки:	**brushColor("green")**

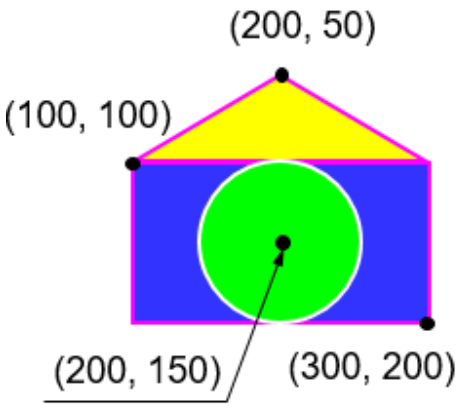
Графические примитивы
(простейшие фигуры)

Точка	Линия	Непрерывный ввод
		
point(x, y)	line(x1, y1, x2, y2)	



Пример программы и результат её работы.

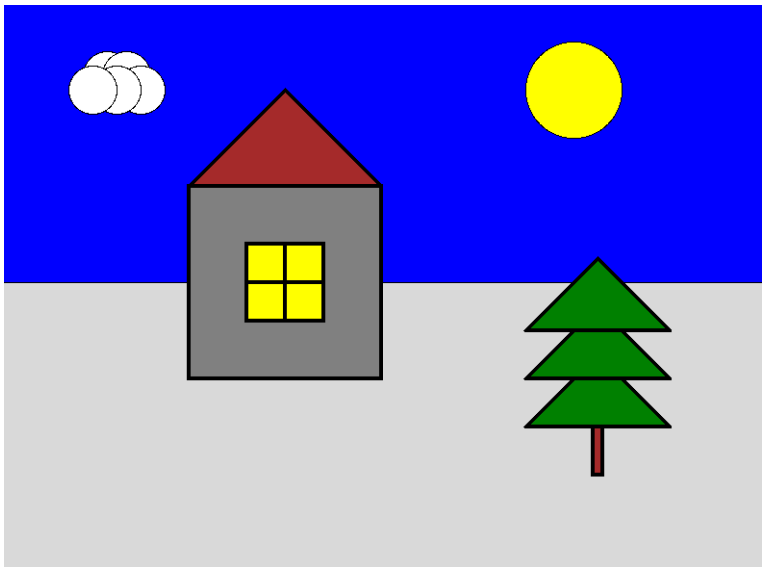
```
from graph import *

penColor("magenta")

brushColor("blue")

rectangle(100,100,300,200)

brushColor("yellow")

polygon([(100,100), (200,50),(300,100), (100,100)])

penColor("white")

brushColor("green")

circle(200, 150, 50)
```



run()	

Примерный вариант задания

Используя полученные знания, нарисуйте любую статическую сцену, которая содержит не менее 3 различных объектов, состоящих из пяти и более примитивов.. Примером сцены может являться следующая картинка:



Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что используют для написания программ с графическими возможностями?
2. Какой оператор позволяет нарисовать точку на экране?

3. Какой оператор позволяет нарисовать линию и прямоугольник?
4. Какие операторы позволяют нарисовать закрашенный прямоугольник?
5. Какой оператор позволяет нарисовать окружность?

Практическое занятие № 40 Использование процедур в графике.

Цель занятия:

1. Изучить понятие процедур в графике в Python;
2. Овладеть операциями и методами над процедурами графики;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №40 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Вспомогательный алгоритм – это алгоритм решения какой-либо подзадачи, который может вызываться из основного алгоритма.

В программировании вспомогательные алгоритмы называют **подпрограммами**. В языке Python существуют два вида подпрограмм: процедуры и функции.

Процедура – это подпрограмма, которая выполняет некоторые действия после вызова её из основной программы или другой процедуры. Каждая процедура имеет уникальное ***имя,*** может иметь произвольное количество входных ***параметров***. При вызове процедуры указываются ***фактические значения параметров***.

Локальные переменные – это переменные, определённые в процедуре, они доступны только внутри процедуры.

Глобальные переменные – это переменные, определённые в основной программе. Они доступны внутри процедуры только для чтения, а для изменения требуется объявить их в процедуре после служебного слова **global**.

Процедура начинается служебным словом **def** (define – «определить»).

Формальные параметры процедуры перечисляются через запятую.

Операторы, входящие в тело процедуры, записываются с отступом. Процедура должна быть определена до первого её вызова.

def <имя>(<параметры>):

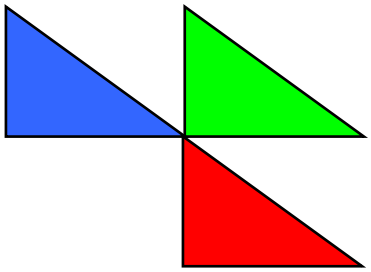
<операторы>

Вызов процедуры осуществляется по её имени с указанием фактических параметров (аргументов).

<имя>(<аргументы>)

Примечание: Между **формальными** и **фактическими** параметрами должно быть соответствие по количеству, порядку следования и типу.

Задача: Построить фигуру:



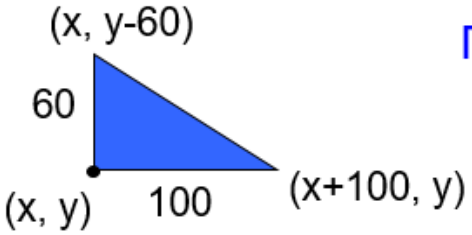
Особенность: Три похожие фигуры.

Общее: размеры, угол поворота

Отличия: координаты, цвет

Отличия обозначаем как переменные,

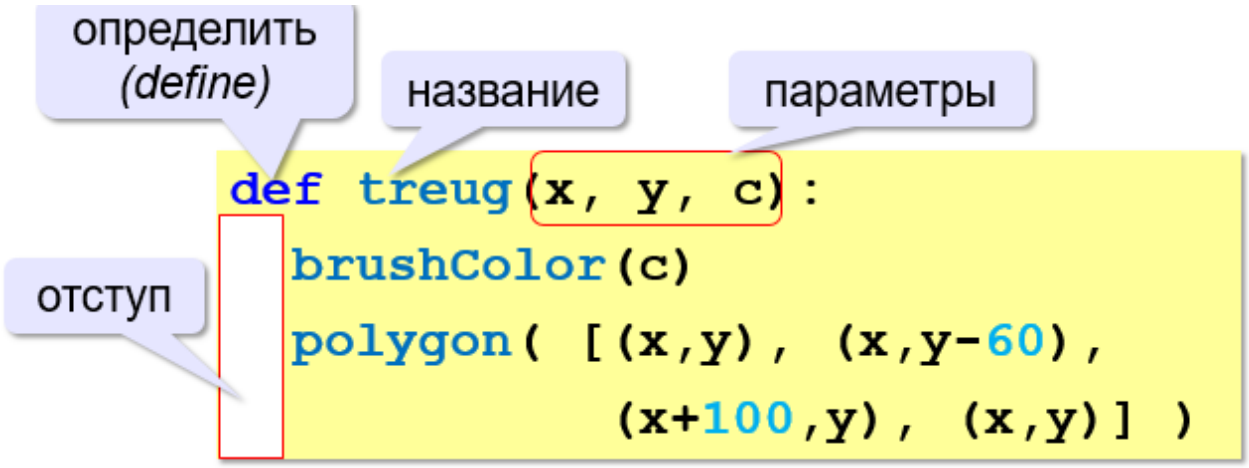
они будут параметрами процедуры.



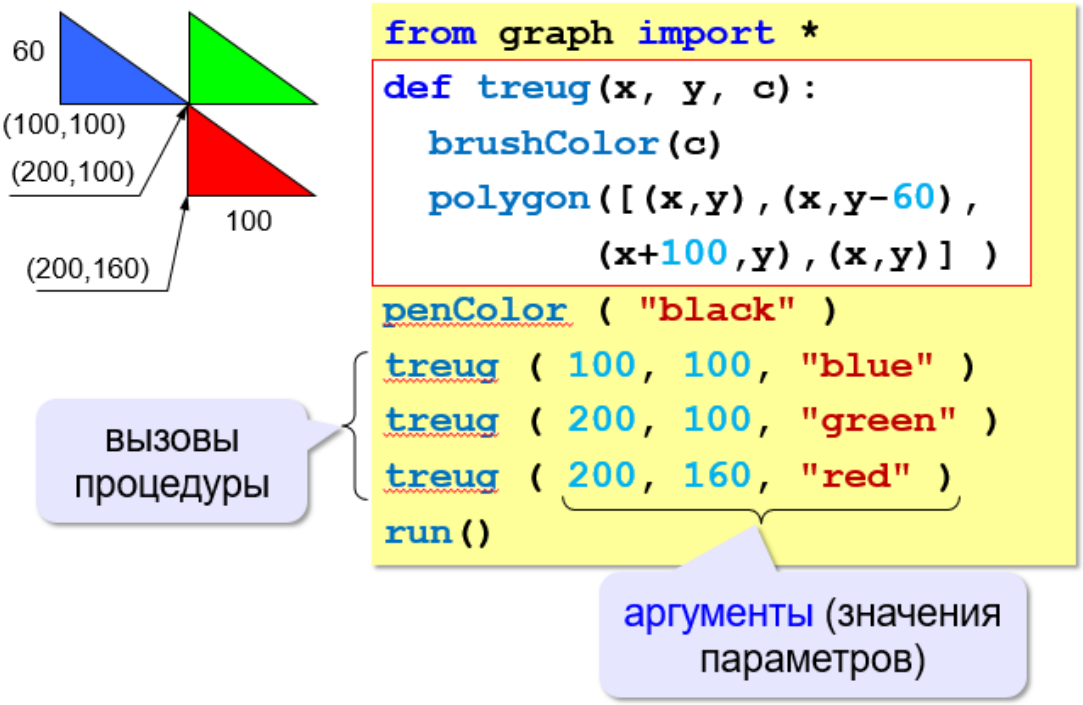
Параметры:

x, y – координаты угла

c – цвет заливки



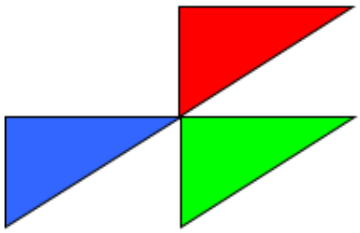
Программа с процедурой

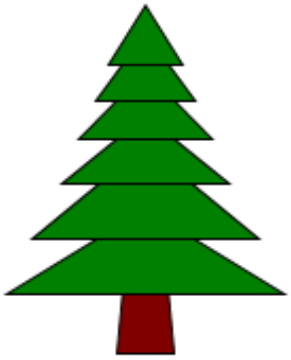


Примерный вариант задания

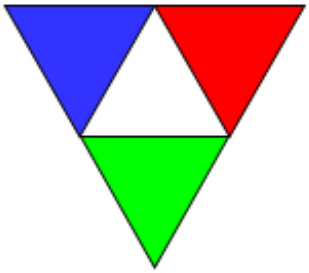
Задание №1 Используя одну процедуру, построить предложенные фигуры:

Оценка «3» Оценка «5»

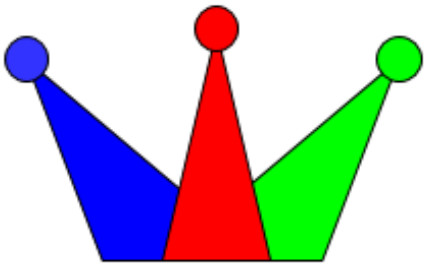




Оценка «4»



Оценка «5»



Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что называют подпрограммами?
2. Какие виды подпрограмм известны?
3. Когда целесообразно использовать подпрограммы в графике?

Практическое занятие № 41 Использование циклов в графике.

Цель занятия:

1. Изучить понятие цикла в графике в Python;
2. Овладеть операциями и методами работы с циклами в графике;
3. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №41 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Теоретическая часть

Алгоритмы описываются с помощью трёх основных структур:

1) следования;

2) ветвления;

3) повторения.

Алгоритм повторения - описание повторяющихся действий. Алгоритм повторения может быть зацикленным и иметь принудительное прерывание и может быть с фиксированным числом повторений с естественным окончанием повторения или цикла.

В языке Python обычно используют два вида циклов:

– «цикл WHILE» или «цикл с предусловием» и попробуем написать первые игры.

WHILE – «пока» в переводе с английского

Например:

Пока <выполняется условие>: делать какие-то действия.

```
n=0
while n<3:
    n=n+1
```

«Пока n меньше 3, прибавлять к n единицу»

Цикл повторяется, пока условие истинно, если же нет, цикл заканчивается

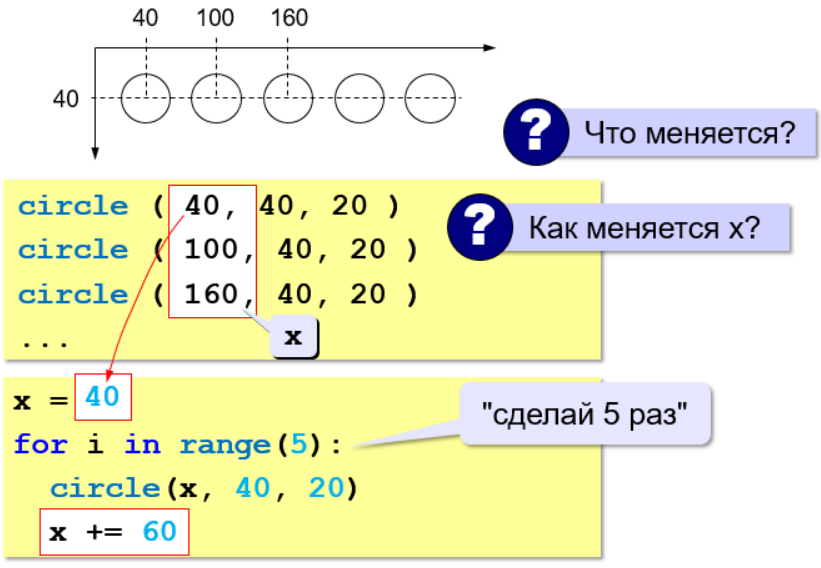
– цикл с параметром FOR.

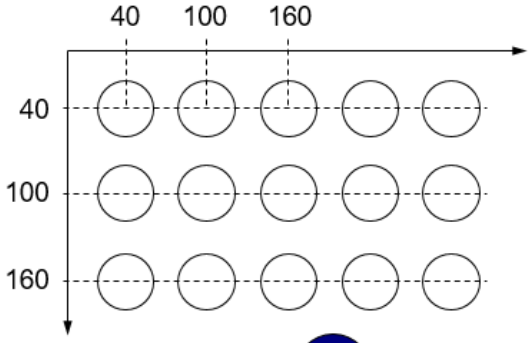
```
for i in range(5):
print("Привет!")
```

В результате работы программы слово «Привет!» будет напечатано 5 раз. Range – диапазон в переводе с английского. При этом переменная i по мере выполнения цикла будет принимать значения 0, 1, 2, 3, 4.

```
for i in range(3, 20):
    print(i)
```

При этом переменная i будет принимать значения **от 3 до 19**





1-й ряд:

Что меняется для 2-го ряда?

```
x = 40
for i in range(5):
  circle(x, 40, 20)
  x += 60
```

y

Можно сделать это процедурой с параметром **y**!

```
from graph import *
def row ( y ):
  x = 40
  for i in range(5):
    circle(x, y, 20)
    x += 60
y = 40
for k in range(3):
  row ( y )
  y += 60
run()
```

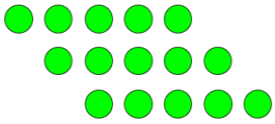
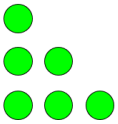
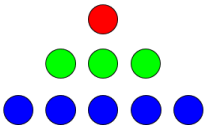
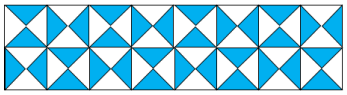
процедура

вызов процедуры

вниз на 60

Примерный вариант задания

Используя циклы и процедуры, нарисуйте узор. Число повторений рисунка N введите с клавиатуры.

	I вариант	II вариант
Оценка «3»	**«3»:** Ввести с клавиатуры число N и нарисовать N рядов по 5 кругов. **Пример** (N = 3): 	**«3»:** Ввести с клавиатуры число N и нарисовать N вертикальных рядов по 5 ромбиков. **Пример** (N = 2): 
Оценка «4»	**«4»:** Ввести с клавиатуры число N и нарисовать из кругов прямоугольный размером N на N. **Пример** (N = 3): 	**«4»:** Используя циклы и процедуры, нарисуйте узор. Число повторений рисунка N введите с клавиатуры. **Пример** (N = 3): 
Оценка «5»	**«5»:** Ввести с клавиатуры число N и нарисовать из кругов равнобедренный треугольник с высотой N. Каждый ряд должен быть покрашен в свой цвет. **Пример** (N = 3): 	**«5»:** Используя циклы и процедуры, нарисуйте узор. 

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что представляет собой алгоритм повторения?

2. Какие виды циклов известны?
3. Когда целесообразно использовать циклы в графике?

Практическое занятие № 42 Штриховка в графике и закрашивание областей.

Цель занятия:

1. Овладеть операциями и методами закрашивания областей в графике Python;
2. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №42 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

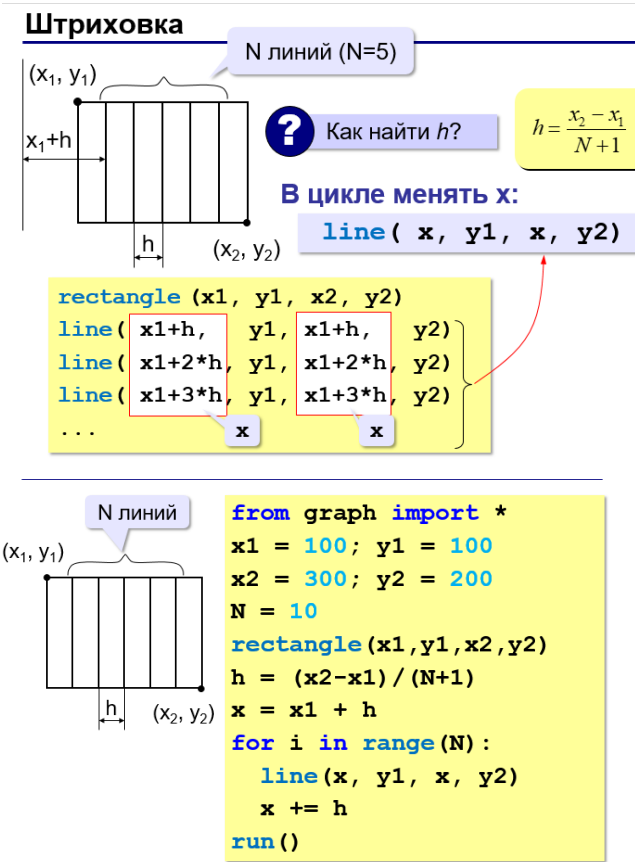
Содержание и порядок выполнения задания:

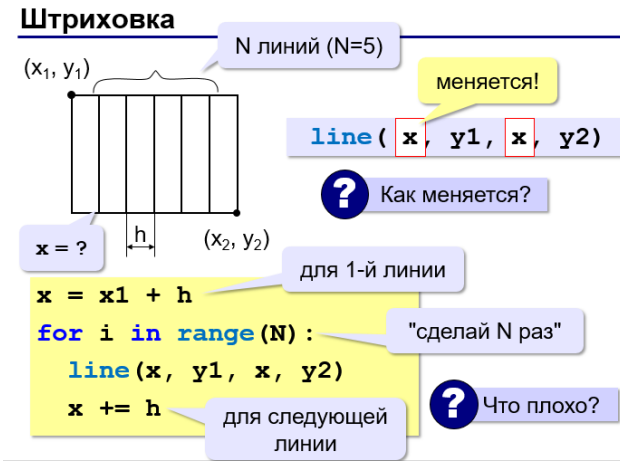
1.Обучающие должны изучить теоретическую часть практического занятия

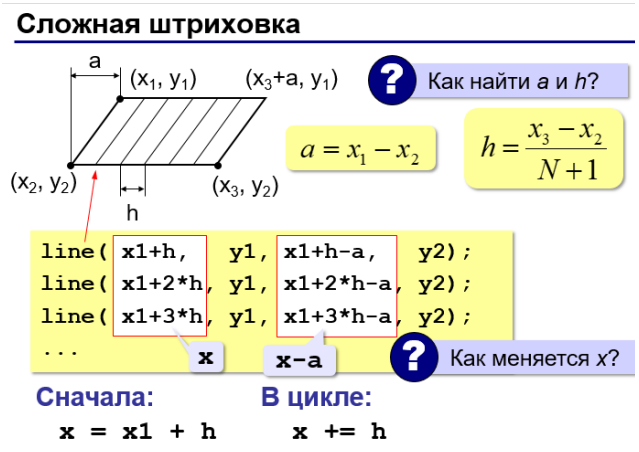
2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

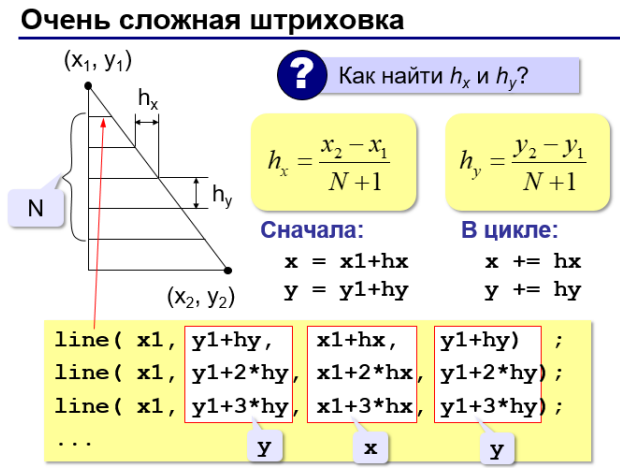
Теоретическая часть повторяется из предыдущего практического занятия

Примерный вариант задания









Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»

5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Как определить шаг между линиями штриховки?
2. Как найти шаг изменения цвета заливки?
3. Как использовать циклы для штриховки области?

Практическое занятие № 43 Построение графиков математических функций в Python.

Цель занятия:

3. Овладеть операциями и методами построения графиков математических функций Python;
4. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №43 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

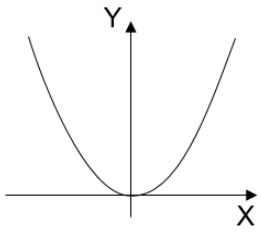
Содержание и порядок выполнения задания:

1.Обучающие должны повторить теоретическую часть практического занятия №41

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Примерный вариант задания

Задача: построить график функции $y = x^2$ на отрезке от -2 до 2.



Анализ:

максимальное значение

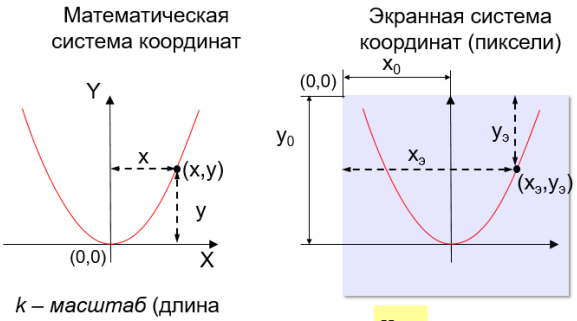
$y_{max} = 4$ при $x = \pm 2$

минимальное значение

$y_{min} = 0$ при $x = 0$

Проблема: функция задана в математической системе координат, строить надо на экране, указывая координаты в пикселях.

Преобразование координат

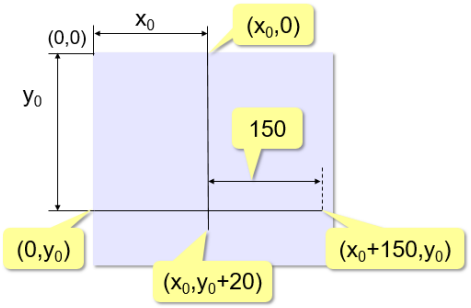


k – масштаб (длина изображения единичного отрезка на экране)

$x_э =$

$y_э =$

Оси координат




```
line(0, y0, x0+150, y0)
line(x0, 0, x0, y0+20)
```

Рисуем оси координат

```
from graph import *
x0 = 150 # начало координат
y0 = 250
k = 50   # масштаб
xmin = -2; xmax = 2 # пределы по x
line(0, y0, x0+150, y0)
line(x0, 0, x0, y0+20)
...
```

Строим по точкам

```
...
x = xmin  # начальное значение x
h = 0.02  # шаг изменения x
penColor("red")
while x <= xmax:
  y = x*x  # функция
  xe = x0 + k*x
  ye = y0 - k*y
  point(xe, ye) # точка на экране
  x += h   # к следующей точке
run()
```

экранные координаты (в пикселях)

Соединяем точки линиями

Идея: сначала создаём в памяти массив точек, затем соединяем точки линиями (`polygon`)

```
points = []  # пустой массив
while x <= xmax:
  y = x*x
  xe = x0 + k*x
  ye = y0 - k*y
  points.append( (xe, ye) )
  x += h
```

добавляем точку в массив

```
penColor("red")
polyline(points) # рисуем линию!
```

Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения
7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Как построить экранную систему координат?
2. Как построить точки графика?
3. Как соединить построенные точки?

Практическое занятие № 44 Анимация в Python.

Цель занятия:

1. Овладеть операциями и методами работы с анимацией в графике Python;
2. Формировать ОК 01, ОК 02.

Исходные данные: теоретический материал, программа Python

Исходные данные:

Папка на РС «Практическое занятие №44 2 семестр» с теоретическим материалом по теме и вопросы для обсуждения.

Содержание и порядок выполнения задания:

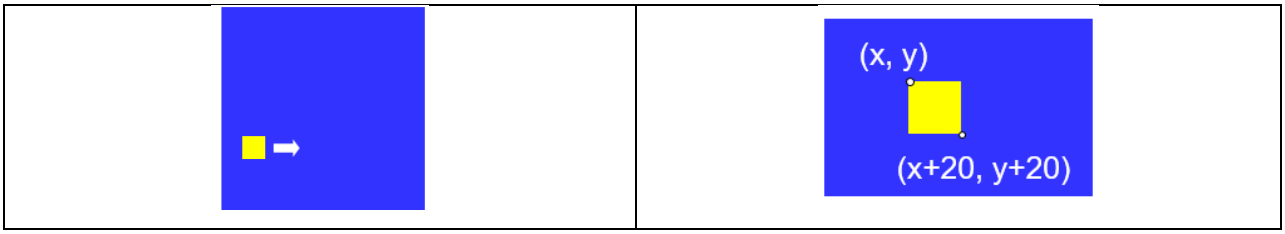
1.Обучающие должны изучить теоретическую часть практического занятия

2. Обучающие должны ответить на вопросы и выполнить задания по вариантам..

Анимация (англ. *animation*) – оживление изображения на экране.

Задача: внутри синего квадрата 400 на 400 пикселей слева направо двигается желтый квадрат 20 на 20 пикселей. Программа останавливается, если нажата клавиша *Esc* или квадрат дошел до границы синей области.

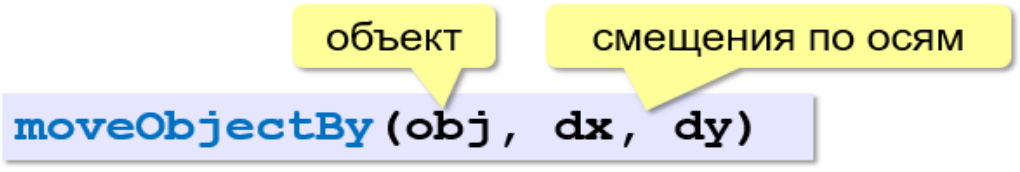
Привязка: состояние объекта задается координатами (x,y)



Принцип анимации:

1. рисуем объект в точке (x,y)
2. задержка на несколько миллисекунд
3. стираем объект
4. изменяем координаты (x,y)
5. переходим к шагу 1

В Python все фигуры, из которых состоит рисунок, – **объекты** (умеют перерисовывать себя сами)!



Начальная картинка

```
from graph import *
brushColor("blue")
rectangle(0, 0, 400, 400)
x = 100
y = 100
penColor("yellow")
brushColor("yellow")
obj = rectangle(x, y, x+20, y+20)
run()
```

синий квадрат

начальные координаты

жёлтый квадрат

Движение

```
def update():
  moveObjectBy(obj, 5, 0)
  if xCoord(obj) >= 380: # если вышел
    close()              # за границу
onTimer(update, 50)
```

x-координата

вызывать update каждые 50 мс

Выход по Escape

Событие (англ. *event*) – изменение состояния какого-то объекта в программе (нажатие на клавишу, щелчок мышью, перемещение или изменение размеров окна и т.п.).

обработчик события

код клавиши Esc = 27

```
def keyPressed(event):
  if event.keycode == VK_ESCAPE:
    close()  # закрыть окно
onKey(keyPressed)
```

вызывать при нажатии любой клавиши

установка обработчика события

Полная программа

```
from graph import *
def update():
  ...
def keyPressed(event):
  ...
brushColor("blue")
rectangle(0, 0, 400, 400)
x = 100
y = 100
penColor("yellow")
brushColor("yellow")
obj = rectangle(x, y, x+20, y+20)
onKey(keyPressed)
onTimer(update, 50)
run()
```

процедуры

обработка событий

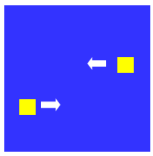
Пример варианта задания

Оценка «3»

Квадрат двигается справа налево:

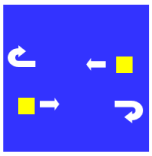
Оценка «4»

Два квадрата двигаются в противоположных направлениях:



Оценка «5»

«5»: Два квадрата двигаются в противоположных направлениях и отталкиваются от стенок синего квадрата:



Содержание отчета:

1. Наименование практического занятия
2. Цель занятия
3. Вариант задания
4. Отчет о выполнении на каждый этап раздела «Содержание и порядок выполнения задания»
5. Список используемых источников
6. Выводы и предложения

7. Дата и подпись курсанта и преподавателя

Вопросы для самопроверки:

1. Что называется анимацией?
2. Как можно имитировать движение объектов?
3. Как можно управлять движением с помощью клавиш?