Федеральное агентство по рыболовству
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Калининградский государственный технический университет»
(ФГБОУ ВО «КГТУ»)

УТВЕРЖДАЮ
Начальник УРОПСП

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
ФГБОУ ВО «КГТУ»
СВЕДЕНИЯ О СЕРТИФИКАТЕ ЭП
Сертификат: 01ff058c007bae2c8149e7712c28ce8469
Владелец: Мельникова Виктория Александровна
Действителен: с 18.04.2022 по 18.04.2023

Фонд оценочных средств
(приложение к рабочей программе модуля)

«ПРОГРАММИРОВАНИЕ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ»

основной профессиональной образовательной программы специалитета
по специальности

10.05.03 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

Специализация

«БЕЗОПАСНОСТЬ ОТКРЫТЫХ ИНФОРМАЦИОННЫХ СИСТЕМ»

ИНСТИТУТ цифровых технологий

РАЗРАБОТЧИК кафедра информационной безопасности

1. РЕЗУЛЬТАТЫ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Таблица 1 – Планируемые результаты обучения по дисциплине, соотнесенные с установленными индикаторами достижения компетенций

Код и наименование компетенции	Индикаторы достижения компетенции	Дисциплина	Результаты обучения (владения, умения и знания), соотнесенные с компетенциями/индикаторами достижения компетенции
ОПК-7: Способен создавать программы на языках общего назначения, применять методы и инструментальные средства программирования для решения профессиональных задач, осуществлять обоснованный выбор инструментария программирования и способов организации программ.	ОПК-7.5: Использует алгоритмы на языках программирования высокого уровня для разработки программных и технических средств защиты информации.	Программирование средств защиты информации	*Знать:* профессиональную и криптографическую терминологию в области безопасности информации. *Уметь*: реализовывать алгоритмы на языках программирования высокого уровня; написать и отладить программы, реализующие соответствующие алгоритмы защиты информации в автоматизированных системах. *Владеть:* методами разработки программного обеспечения средств защиты информации.

2. ПЕРЕЧЕНЬ ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПОЭТАПНОГО ФОРМИРОВАНИЯ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ (ТЕКУЩИЙ КОНТРОЛЬ) И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

2.1 Для оценки результатов освоения дисциплины используются:

- оценочные средства текущего контроля успеваемости;

- оценочные средства для промежуточной аттестации по дисциплине.

2.2 К оценочным средствам текущего контроля успеваемости относятся:

- тестовые задания;

- задания и контрольные вопросы по лабораторным работам;

2.3 К оценочным средствам для промежуточной аттестации по дисциплине, проводимой в форме дифференцированного зачета и экзамена, относятся:

- задание по курсовой работе;

- промежуточная аттестация в форме дифференцированного зачета проходит по результатам прохождения всех видов текущего контроля успеваемости;

- экзаменационные вопросы.

3. ОЦЕНОЧНЫЕ СРЕДСТВА ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ

3.1. Тестовые задания и ключи к ним приведены в приложениях № 1 и 4 соответственно.

Таблица 4 - Шкала оценок уровня освоения дисциплины по тесту.

Оценка			
Неудовлетворительный	Пороговый	Углубленный	Продвинутый
«2» (неудовлетворительно)	«3» (удовлетворительно)	«4» (хорошо)	«5» (отлично)
Менее 50% правильных выполненных заданий.	50-70% правильных выполненных заданий.	71-90% правильных выполненных заданий.	91-100% правильных выполненных заданий.

3.2. Задания и контрольные вопросы к лабораторным работам приведены в приложении № 2.

4. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

4.1 Промежуточная аттестация по дисциплине проводится в форме дифференцированного зачета, экзамена. Типовые вопросы к экзамену приведены в приложении № 3

Промежуточная аттестация проходит по результатам прохождения всех видов текущего контроля успеваемости. К зачету, экзамену допускаются студенты, положительно аттестованные по результатам текущего контроля.

Дисциплина рассчитана на два семестра (9 и 10 семестр), в 9 семестре проводится зачет с оценкой, в 10 семестре – экзамен.

Таблица 2 - Шкала оценок уровня освоения дисциплины по зачету

Оценка			
Неудовлетворительный	Пороговый	Углубленный	Продвинутый
«2» (неудовлетворительно)	«3» (удовлетворительно)	«4» (хорошо)	«5» (отлично)
Правильные ответы даны менее чем на 50% включительно. Материал излагается непоследовательно, сбивчиво, не представляет определенной системы знаний по	Правильные ответы даны на 51-64% вопросов. Допускаются нарушения в последовательности изложения. Демонстрируются поверхностные знания вопроса.	Правильные ответы даны на 65-94% вопросов. Ответы на поставленные вопросы излагаются систематизировано и последовательно. Материал излагается уверенно. Демон-	Правильные ответы даны на 95-100% вопросов. Ответы на поставленные в билете вопросы излагаются логично, последовательно и не требуют дополнительных пояснений. Делаются обоснованные выводы.

дисциплине. Имеются заметные нарушения норм литературной речи.	Имеются затруднения с выводами. Допускаются нарушения норм литературной речи.	стрируется умение анализировать материал, однако не все выводы носят аргументированный и доказательный характер. Соблюдаются нормы литературной речи.	Демонстрируются глубокие знания предмета. Соблюдаются нормы литературной речи.

Таблица 3 - Шкала оценок уровня освоения дисциплины по экзамену.

Оценка			
Неудовлетворительный	Пороговый	Углубленный	Продвинутый
«2» (неудовлетворительно)	«3» (удовлетворительно)	«4» (хорошо)	«5» (отлично)
Не знает значительной части программного материала, допускает существенные ошибки, с большими затруднениями выполняет практические задания, задачи.	Усвоил только основной материал, но не знает отдельных деталей, допускает неточности, недостаточно правильные формулировки, нарушает последовательность в изложении программного материала и испытывает затруднения в выполнении практических заданий.	Твердо знает программный материал, грамотно и по существу излагает его, не допускает существенных неточностей в ответе на вопрос, может правильно применять теоретические положения и владеет необходимыми умениями и навыками при выполнении практических заданий.	Глубоко и прочно усвоил весь программный материал, исчерпывающе, последовательно, грамотно и логически стройно его изложил, не затрудняется с ответом при видоизменении задания, свободно справляется с задачами и практическими заданиями, правильно обосновывает принятые решения, умеет самостоятельно обобщать и излагать материал, не допуская ошибок.

4.3. Примерные темы курсовых работ даны в приложении № 4.

Таблица 5 - Шкала оценок курсовой работы

Оценка			
Неудовлетворительный	Пороговый	Углубленный	Продвинутый
«2» (неудовлетворительно)	«3» (удовлетворительно)	«4» (хорошо)	«5» (отлично)
Работа носит реферативный характер, студент допускает существенные ошибки при защите, с	Работа носит реферативный характер, допускает неточности, недостаточно правильные формулировки, нарушает последовательность	В работе углублены теоретические и практические знания, материал излагается грамотно и по существу, не до-	В процессе выполнения работы приобретены навыки самостоятельного планирования и выполнения научно-исследовательской работы; получен опыт сбора и обработки

большими затруднениями отвечает на вопросы, оформление работы не соответствует правилам.	в изложении при защите, оформление работы имеет незначительные отклонения от правил.	пускается существенных неточностей в ответе на вопрос, оформление работы соответствует правилам.	исходного материала, анализа научно-технической литературы, материал излагается грамотно, оформление работы соответствует правилам.

5. СВЕДЕНИЯ О ФОНДЕ ОЦЕНОЧНЫХ СРЕДСТВ И ЕГО СОГЛАСОВАНИИ

Фонд оценочных средств для аттестации по дисциплине «Программирование средств защиты информации» представляет собой компонент основной профессиональной образовательной программы специалитета по специальности 10.05.03 Информационная безопасность автоматизированных систем (специализация «Безопасность открытых информационных систем»).

Фонд оценочных средств рассмотрен и одобрен на заседании кафедры информационной безопасности 20.04.2022 г. (протокол № 7).

Заведующий кафедрой  Н.Я. Великите

Приложение № 1

ТИПОВЫЕ ВАРИАНТЫ ТЕСТОВЫХ ЗАДАНИЙ (семестр 9)

	ВАРИАНТ 1
1.	В методе дихотомии условие $F(a)F(b)<0$ необходимо для 1. определения нового интервала поиска корня 2. вычисления корня на интервале [a,b] 3. проверки необходимости следующей итерации
2.	В методе Ньютона условие $F(x_0)F''(x_0)>0$ необходимо для: 1. определения начального значения корня 2. для вычисления погрешности решения 3. проверки необходимости следующей итерации
3.	В методе Ньютона первая производная должна быть: 1. отлична от нуля 2. равна нулю 3. это зависит от отрезка, на котором вычисляется корень
4.	Обращаться к динамической памяти необходимо, если: 1. в тексте программы присутствуют указатели 2. размерность программного кода вместе с данными превышает 64 Кб 3. в тексте программы присутствует обращение к файлам
5.	При высвобождении динамической памяти квадратные скобки в операторе *delete[]*: 1. не нужны, при их указании возникнет ошибка 2. обязательны, при их отсутствии программа поведет себя непредсказуемо 3. не нужны, достаточно указать после ключевого слова *delete* размерность удаляемого массива
6.	В аффинном шифре в качестве аддитивного ключа выбирается: 1. число, имеющее мультипликативную инверсию в Zn 2. число, не имеющее мультипликативную инверсию в Zn 3. любое число в Zn
7.	В аффинном шифре мультипликативный ключ: 1. может быть равен единице, потому что при любом n в Zn единица имеет мультипликативную инверсию 2. не должен быть равен единице потому что в этом случае :аффинный шифр вырождается в аддитивный шифр 3. должен быть равен единице, потому что это является обязательным условием в аффинном шифре
8.	При программировании классов доступ из main() к закрытым членам класса возможен только через: 1. открытые методы 2. объекты 3. непосредственное обращение к закрытым полям
9.	Конструктор класса вызывается: 1. при создании объекта этого класса 2. при непосредственном обращении к конструктору в программе 3. по окончании работы программы
10.	Производный класс: 1. является точной копией базового класса 2. наследует черты базового класса и добавляет свои свойства

	3. работает только с собственными свойствами, не совпадающими с свойствами базового класса
11.	Если производный класс наследуется как *private,* это значит, что все открытые: 1. элементы базового класса будут также открытыми элементами производного класса 2. и все закрытые элементы базового класса будут также открытыми элементами для производного класса 3. и закрытые члены базового класса в производном классе становятся недоступными
12.	Если спецификатор доступа производного класса не указан, то базовый класс по умолчанию наследуется как: 1. private 2. protected 3. public
13.	В объявлении дружественной функции указывается ключевое слово: 1. friend 2. brother 3. sister
14.	Функция, являясь дружественной некоторому классу: 1. не может быть одновременно дружественной другому классу 2. может быть одновременно дружественной другому классу 3. может быть одновременно дружественной другому классу, если ее тип void
15.	Любой класс: 1. не может быть базовым для нескольких производных классов 2. не может быть базовым для нескольких производных классов, если наследуется как private 3. может быть базовым для нескольких производных классов
16.	Базовый класс наследуется как protected, тогда производные классы: 1. могут «видеть» его элементы, имеющие спецификатор доступа protected 2. не могут «видеть» его элементы, имеющие спецификатор доступа protected 3. как наследники, могут «видеть» его элементы, имеющие любой спецификатор доступа
17.	Явно вызывать деструктор необходимо: 1. всегда 2. только при условии существования конструктора объекта 3. для освобождения оперативной памяти
18.	Перегруженные функции имеют: 1. различные имена, но одинаковые параметры 2. одинаковые имена, но различные параметры 3. одинаковые имена и одинаковые параметры
19.	Для определения того, какую версию перегруженной функции вызвать, компилятор использует: 1. тип и/или количество аргументов функции 2. имя функции 3. возвращаемый тип функции
20.	Если компилятор не в состоянии выполнить запрос создания *inline*-функции, то: 1. сгенерируется ошибка компиляции 2. сгенерируется ошибка выполнения 3. функция будет компилироваться, как обычная функция, а запрос будет проигнорирован
	ВАРИАНТ 2

1.	В методе дихотомии вычисление $\|b-a\|$ необходимо для: 1. определения нового интервала поиска корня 2. вычисления корня на интервале [a,b] 3. проверки необходимости следующей итерации
2.	В методе Ньютона формула $x_{k+1} = x_k - \frac{F(x_k)}{F\prime(x_k)}$. необходима для: 1. определения начального значения корня 2. вычисления погрешности решения 3. проверки необходимости следующей итерации 4. вычисления корня текущей итерации
3.	Для программирования метода дихотомии как входные данные необходимы: начальное значение,…….. 1. интервала, конечное значение интервала, погрешность 2. интервала, начальное значение корня, погрешность 3. корня, количество итераций, погрешность
4.	Инициализировать динамический массив: 1. можно 2. необходимо 3. нельзя.
5.	Входные данные для программирования аффинного шифра: исходный текст, ….. 1. мультипликативный ключ (аддитивный ключ вычисляется на основе мультипликативного ключа) 2. аддитивный ключ (мультипликативный ключ вычисляется на основе аддитивного ключа) 3. мультипликативный ключ, аддитивный ключ
6.	При программировании аффинного шифра в качестве входного данного подать отрицательный мультипликативный ключ: 1. можно, но его необходимо перепрограммировать в положительный 2. нельзя, потому что в любой системе вычетов отсутствуют отрицательные числа 3. можно, перепрограммировать в положительный не нужно
7.	Классы необходимы для: 1. уменьшения объема программы 2. уменьшения объема используемой оперативной памяти 3. для создания новых типов данных
8.	При программировании классов доступ из main() к открытым членам класса возможен только через: 1. открытые методы 2. объекты. 3. непосредственное обращение к закрытым полям
9.	Конструктор класса может: 1. не только инициализировать поля, но и, как любая функция, может выполнять другие запрограммированные в нем операции 2. только инициализировать поля класса 3. выполнять любые операции, и это полезно для оптимизации кода программы
10.	Если производный класс наследуется как *public*, это значит, что все открытые: 1. элементы базового класса будут также открытыми элементами производного класса 2. и все закрытые элементы базового класса будут также открытыми элементами для производного класса

	3. и все закрытые элементы базового класса будут закрытыми элементами, и к ним не будет прямого доступа из производного класса.
11.	Если производный класс наследуется как *protected,* это значит, открытые и защищенные члены базового класса становятся: 1. закрытыми членами производного класса 2. защищенными членами производного класса 3. открытыми членами производного класса
12.	Если спецификатор доступа к полю класса не указан, то поле по умолчанию объявляется как: 1. private 2. protected 3. public
13.	Дружественная функция: 1. наследуется производным классом 2. наследуется, если спецификатором доступа к производному классу является public 3. не наследуется производным классом
14.	Дружественная функция: 1. в качестве параметров получает объекты дружественных классов 2. не имеет параметров 3. в качестве параметров получает простые переменные.
15.	Базовый класс наследуется как private, тогда производные классы: 1. могут «видеть» его элементы, имеющие спецификатор доступа protected 2. не могут «видеть» его элементы, имеющие спецификатор доступа protected 3. как наследники, могут «видеть» его элементы, имеющие любой спецификатор доступа
16.	Виртуальные базовые классы нужны для: 1. преодоления неоднозначности в множественном наследовании 2. указания порядка наследования в множественном наследовании 3. удаления наследования
17.	Деструктору класса присваивается имя: 1. такое же, как имя класса 2. такое же, как имя класса, кроме того, его имя предваряется символом тильды (~) 3. выбранное программистом
18.	Функцию можно перегружать: 1. только один раз 2. не более двух раз 3. много раз
19.	Перегруженные функции должны отличаться: 1. типами и/или числом параметров функции 2. именем и/или возвращаемым типом функции 3. именем и/или числом параметров функции
20.	Спецификатор *inline* для компилятора является: 1. командой 2. запросом 3. указателем
	ВАРИАНТ 3
1.	В методе дихотомии вычисление $(a + b)/2$ | необходимо для: 1. определения нового интервала поиска корня

	2. вычисления корня на интервале [a,b] 3. проверки необходимости следующей итерации
2.	В методе Ньютона условие $\|x_{k+1} - x_k\| < \varepsilon$ необходимо для: 1. определения начального значения корня 2. вычисления погрешности решения 3. проверки необходимости следующей итерации
3.	Для программирования метода Ньютона как входные данные необходимы: начальное значение ……. 1. интервала, конечное значение интервала, погрешность 2. интервала, начальное значение корня, погрешность 3. корня, количество итераций, погрешность
4.	Правильное выделение динамическую память для типов double, int: 1. *int *a=new int[10];* *double *b=new double[20];* 2. *int a=new int[10];* *double b=new double[20];* 3. *int *a= int[10];* *double *b= double[n];*
5.	В аффинном шифре в качестве мультипликативного ключа выбирается: 1. число, имеющее аддитивную инверсию в Zn 2. число, имеющее мультипликативную инверсию в Zn 3. любое число в Zn
6.	При программировании аффинного шифра в Zn в качестве входного данного подать аддитивный ключ величиной больше n: 1. можно, но его необходимо перепрограммировать так, чтобы он стал числом, входящим в Zn. 2. нельзя, в Zn нет чисел, больших n. 3. нельзя, аддитивный ключ должен быть обязательно равен нулю.
7.	В классе существуют следующие спецификаторы доступа: 1. private, protected, public 2. private, real, public 3. private, public
8.	Конструктор класса должен: 1. иметь имя, отличное от имени класса, не иметь типа 2. иметь то же имя, что и класс, иметь тип void 3. иметь то же имя, что и класс, не иметь типа
9.	Конструктор класса должен: 1. не только инициализировать поля, но и выполнять любые другие запрограммированные в нем операции 2. только инициализировать поля класса 3. выполнять все операции, оптимизирующие код программы
10.	Если производный класс наследуется как *public*, это значит: 1. все закрытые элементы базового класса останутся закрытыми 2. все закрытые элементы базового класса будут также открытыми элементами для производного класса 3. все открытые и все закрытые элементы базового класса будут закрытыми элементами, и к ним не будет прямого доступа из производного класса.
11.	Если производный класс наследуется как *protected,* это значит:

	1. открытые и защищенные члены базового класса становятся закрытыми членами производного класса 2. закрытые члены базового класса остаются закрытыми и недоступными для производного класса 3. закрытые члены базового класса становятся открытыми и доступными для производного класса
12.	При объявлении класса его имя указывать: 1. обязательно, в противном случае сгенерируется ошибка компилятора 2. не нужно, оно создается в теле программы 3. необязательно, но это создаст неудобства для программиста
13.	Дружественная функция: 1. объявляется внутри класса, а определяется вне класса 2. и объявляется, и определяется вне класса 3. объявляется вне класса, а определяется внутри класса
14.	Множественное наследование бывает: 1. прямое и обратное 2. косвенное и обратное 3. прямое и косвенное
15.	Если у базового и производного классов есть конструкторы с параметрами, то конструктор производного класса: 1. должен передавать параметры для конструктора базового класса в том случае, если у базового класса нет полей 2. должен передавать параметры для конструктора базового класса 3. не должен передавать параметры для конструктора базового класса
16.	Базовый класс наследуется как public, тогда производные классы: 1. могут «видеть» его элементы, имеющие спецификатор доступа protected 2. не могут «видеть» его элементы, имеющие спецификатор доступа protected 3. как наследники, могут «видеть» его элементы, имеющие любой спецификатор доступа
17.	Работа деструктора класса инициируется: 1. по указанию программиста 2. при объявлении объекта. 3. при удалении объекта.
18.	Перегрузка функций – один из способов реализации: 1. наследования 2. полиморфизма 3. инкапсуляции
19.	Одно и то же имя функции использовать для перегрузки не связанных общими действиями функций: 1. можно, но это плохой стиль программирования 2. нельзя, иначе компилятор сгенерирует ошибку 3. нельзя, потому что существует правило, которое обязывает программиста связывать перегруженные функции общими действиями
20.	Встраиваемая *inline*-функция – это функция, которая: 1. вызывается как обычная функция 2. не вызывается, а ее тело встраивается в программу в месте ее вызова 3. вызывается как обычная функция, а ее тело встраивается в программу в месте ее вызова

ТИПОВЫЕ ВАРИАНТЫ ТЕСТОВЫХ ЗАДАНИЙ (семестр А)

	ВАРИАНТ 1
1.	Перед спецификатором доступа виртуального базового класса ставится ключевое слово: 1. *derived* 2. *virt* 3. *virtual*
2.	Если класс, содержащий виртуальную функцию, наследуется, то в производном классе виртуальная функция: 1. переопределяется 2. не переопределяется 3. вообще не упоминается, потому что виртуальные функции не наследуются
3.	Прототип чистой виртуальной функции: 1. *virtual тип имя_функции (список_параметров) = 1;* 2. *virtual тип имя_функции (список_параметров) = -1;* 3. *virtual тип имя_функции (список_параметров) = 0;*
4.	Объект абстрактного класса объявить: 1. можно в любом случае 2. нельзя, абстрактный класс технически неполноценен 3. можно, если абстрактный класс является производным классом
5.	Входные данные для программы «Вычисление суммы точек эллиптической кривой»: 1. координаты обеих точек, характеристика поля, коэффициенты уравнения эллиптической кривой 2. координаты обеих точек, коэффициенты уравнения эллиптической кривой 3. характеристика поля, коэффициенты уравнения эллиптической кривой
6.	Формулы сложения точек эллиптической кривой при условии $P \neq \pm Q$: 1. $\lambda = \frac{y_2 - y_1}{x_2 - x_1} (mod\ p)$ $x_3 = \lambda^2 - x_1 - x_2\ (mod\ p)$ $y_3 = \lambda(x_1 - x_3) - y_1 (mod\ p)$ 2. $\lambda = \frac{3{x_1}^2 + a}{2y_1} (mod\ p)$ $x_3 = \lambda^2 - 2x_1\ (mod\ p)$ $y_3 = \lambda(x_1 - x_3) - y_1 (mod\ p)$ 3. $\lambda = \frac{y_2 - y_1}{x_2 - x_1} mod\ p$ $x_3 = \lambda^2 - x_1 - x_2\ mod\ p$ $y_3 = \lambda(x_1 - x_3) - y_1 mod\ p$
7.	Оптимизация программы – это: 1. упрощение кода программы с целью экономии ОП

	2. уменьшение количества решаемых задач в программе 3. построение эквивалентной программы, обладающей лучшими характеристиками времени работы и/или объема занимаемой ОП
8.	Концепция модульного программирования: 1. разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей; модули связаны между собой только по входным и выходным данным 2. разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей; модульный подход не позволяет разрабатывать части программ одного проекта на разных языках программирования 3. задача состоит из одного крупного модуля, который можно модернизировать, но нельзя разбивать на самостоятельные модули
9.	Инкапсуляция – это: 1. возможность добавлять в существующий класс новые методы 2. возможность добавлять в существующий класс новые методы и новые свойства 3. объединение в одном классе данные и методы, манипулирующие этими данными
10.	Задавать аргументы по умолчанию одновременно и в определении, и в прототипе функции: 1. можно 2. нельзя 3. можно при условии, что аргумент по умолчанию является глобальной переменной
11.	Freeware – это: 1. бесплатные, свободно распространяемые программы 2. некоммерческие программы, которые могут использоваться, как правило, бесплатно 3. программное обеспечение, которое является полнофункциональным в течение определенного времени или количества запусков
12.	Рекурсия – это: 1. многократный вызов подпрограммы головной программой 2. подпрограмма вызывает сама себя 3. помещение тела функции в программу, избегая таким образом издержек на вызов функции
13.	Динамическими структурами в программировании являются: 1. списки, очереди, деревья 2. перечисления, очереди, деревья 3. списки, очереди, ветви
14.	Обращение к динамической памяти для типов float, int в C++: 1. float ptrf = new (float) [10]; int ptri=new (int)[10] 2. float *ptrf = new (float) [10]; int *ptri=new (int)[10] 3. float *ptrf = new float [10]; int *ptri=new int[10]
15.	Выражение *ivector.at(1) = 5;* означает: 1. объявление длины вектора 2. изменение значение второго элемента вектора 3. добавление пятерки в начало вектора
16.	Метод *pop_back()*: 1. добавляет ячейку в начало вектора. 2. добавляет ячейку в конец вектора.

	3. удаляет ячейку в конце вектора.
17.	Указатель – это: 1. переменная, значением которой является адрес ячейки памяти 2. константа, которая указывает, по какому адресу оперативной памяти записать программу 3. переменная, которая содержит адрес сегмента памяти, в котором сохранена программа
18.	Правильное объявление производного класса: 1. class MoreDetails:: Details; 2. class MoreDetails: public class Details; 3. class MoreDetails: class(Details); 4. class MoreDetails: public Details;
19.	Правильные соответствия между спецификатором базового класса, спецификатором доступа в объявлении производного класса и правами доступа производного класса к элементам базового; спецификатор доступа: 1. public; в базовом классе – private; права доступа в производном классе – protected 2. private; в базовом классе – public; права доступа в производном классе – public 3. protected или public; в базовом классе – protected; права доступа в производном классе – protected
20.	Для доступа к элементам объекта используются при обращении через имя объекта: 1. два двоеточия, при обращении через указатель – операция «точка» 2. точка, при обращении через указатель – два двоеточия 3. два двоеточия, при обращении через указатель – операция «–>» 4. точка, при обращении через указатель – операция «–>»
21.	Перегрузке поддаются операции: 1. только бинарные 2. унарные и бинарные 3. только унарные
22.	Входные данные для программы «Алгоритм рюкзака»: 1. исходное сообщение, секретный ключ, число m (которое должно быть больше суммы элементов секретного ключа) 2. открытый и закрытый ключ, исходное сообщение 3. мультипликативная инверсия числа m, исходное сообщение, секретный ключ
23.	В программе «Вычисление мультипликативной инверсии» используется: 1. алгоритм Евклида 2. шифр Магма 3. расширенный алгоритм Евклида
24.	В программе «Вычисление наибольшего общего делителя» используется: 1. шифр DES 2. алгоритм Евклида 3. алгоритм ECDH
25.	Указатель на базовый класс: 1. может использоваться как указатель на любой его производный класс 2. не может использоваться как указатель на производный класс 3. может использоваться как указатель на любой его производный класс при условии наследования со спецификатором public

26.	Для указания на объект производного класса можно воспользоваться указателем базового класса, при этом доступ: 1. будет обеспечен ко всем членам производного класса 2. может быть обеспечен только к тем членам производного класса, которые не были унаследованы от базового 3. может быть обеспечен только к тем членам производного класса, которые были унаследованы от базового
27.	В родовых классах определены все: 1. необходимые алгоритмы, а фактические типы обрабатываемых данных задаются в качестве параметров при создании объектов этого класса 2. необходимые алгоритмы и типы обрабатываемых данных 3. типы обрабатываемых данных
28.	Обработка исключительных ситуаций в *C++* организуется с помощью ключевых слов: 1. *try, catch* 2. *try, catch* и *throw* 3. *try, catch, type*
29.	Функции, которые вызываются из блока *try:* 1. технически неполноценны 2. не могут возбуждать исключительную ситуацию 3. также могут возбуждать исключительную ситуацию
30.	Правильное объявление вектора: 1. *vector <double> vd[n];* 2. *vector (double) vd[n];* 3. *vector (double) vd;*
	ВАРИАНТ 2
1.	Виртуальная функция определяется: 1. внутри базового класса. 2. внутри производного класса. 3. вне любого класса.
2.	Класс, содержащий виртуальную функцию, называется: 1. изоморфным 2. полиморфным 3. гомеоморфным
3.	Абстрактный класс: 1. содержит хотя бы одну виртуальную функцию 2. содержит ровно одну виртуальную функцию 3. это то же самое, что и виртуальный базовый класс
4.	Родовая функция (*функция-шаблон*): 1. работает только с целочисленными типами данных. 2. работает только с символьными типами данных 3. определяет базовый набор операций, которые будут применяться к разным типам данных
5.	Входные данные для программы «Вычисление порядка точки эллиптической кривой»: 1. координаты точки, характеристика поля, коэффициенты уравнения эллиптической кривой 2. координаты точки, коэффициенты уравнения эллиптической кривой 3. характеристика поля, коэффициенты уравнения эллиптической кривой

6.	Формулы сложения точек эллиптической кривой при условии *P=Q:* 1. $$\lambda = \frac{y_2 - y_1}{x_2 - x_1}(mod\ p)$$ $$x_3 = \lambda^2 - x_1 - x_2\ (mod\ p)$$ $$y_3 = \lambda(x_1 - x_3) - y_1(mod\ p)$$ 2. $$\lambda = \frac{3{x_1}^2 + a}{2y_1}(mod\ p)$$ $$x_3 = \lambda^2 - 2x_1\ (mod\ p)$$ $$y_3 = \lambda(x_1 - x_3) - y_1(mod\ p)$$ 3. $$\lambda = \frac{3{x_1}^2 + a}{2y_1} mod\ p$$ $$x_3 = \lambda^2 - 2x_1\ mod\ p$$ $$y_3 = \lambda(x_1 - x_3) - y_1 mod\ p$$
7.	Порядок следования основных этапов технологического процесса разработки программ: 1. а) постановка задачи; б) построение математической модели; в) разработка (выбор и адаптация) алгоритма; г) составление программы. тестирование и отладка программы; д) сдача в эксплуатацию 2. а) построение математической модели; б) постановка задачи. разработка (выбор и адаптация) алгоритма; в) составление программы; г) сдача в эксплуатацию; д) тестирование и отладка программы 3. а) постановка задачи. разработка (выбор и адаптация) алгоритма; б)построение математической модели; в)тестирование и отладка программы; г)составление программы; д)сдача в эксплуатацию
8.	Основными характеристиками качества программного продукта (ПП) являются: 1. надежность, строгая зависимость от аппаратного обеспечения ЭВМ 2. качество характеризуется количеством продаж данного ПП 3. мобильность, надежность, эффективность
9.	Полиморфизм – это возможность: 1. многократного использования методов, определенных в классе 2. включать в состав класса несколько функций с одинаковыми именами 3. многократного использования объектов класса
10.	Все параметры функции, задаваемые по умолчанию: 1. должны указываться правее параметров, передаваемых обычным путем 2. должны указываться перед параметрами, передаваемыми обычным путем 3. не должны передаваться функции
11.	Shareware – это: 1. бесплатные, свободно распространяемые программы 2. программное обеспечение, которое является полнофункциональным в течение определенного времени или количества запусков 3. некоммерческие программы, которые могут использоваться, как правило, бесплатно

12.	Отладка – это 1. процесс выполнения программы на некотором наборе данных 2. обнаружение и исправление ошибок в программе 3. компиляция программы
13.	Вектор – это 1. набор однотипных значений, к которым можно обращаться в произвольном порядке. 2. набор разнотипных значений. 3. числовой динамический массив.
14.	Прототип функции – это: 1. объявление функции, где указываются тип возвращаемого значения, имя функции и список параметров в скобках 2. определение функции, где указываются имя и тело функции 3. объявление и определение функции
15.	Выражение *vector_second.reserve(5);* означает: 1. добавление пятерки в начало вектора 2. изменение значение второго элемента вектора 3. объявление длины вектора
16.	Метод *push_back():* 1. добавляет ячейку в начало вектора 2. добавляет ячейку в конец вектора 3. удаляет ячейку в конце вектора
17.	Для скорости выполнения реализация вложенных циклов: 1. предпочтительнее, если внешний цикл имеет меньше итераций по отношению к внутреннему 2. предпочтительнее, если внутренний цикл имеет меньше итераций по отношению к внешнему 3. не имеет значения количество итераций внешних и внутренних циклов
18.	Возможность и способ обращения производного класса к элементам базового определяется: 1. спецификаторами доступа: private, public, protected в теле производного класса 2. только спецификатором доступа protected в заголовке объявления производного класса 3. спецификаторами доступа: private, public, protected в теле базового класса 4. спецификаторами доступа: private, public, protected в заголовке объявления производного класса
19.	Правильные соответствия между спецификатором базового класса, спецификатором доступа в объявлении производного класса и правами доступа производного класса к элементам базового спецификатор доступа: 1. public; в базовом классе – private; права доступа в производном классе – protected 2. private; в базовом классе – public; права доступа в производном классе – public 3. любой; в базовом классе – public; права доступа в производном классе – такие же, как ключ доступа
20.	При динамическом выделении памяти память под объект (переменную): 1. может выделяться не сразу, а в процессе работы программы, освобождение памяти производится автоматически после завершения программы 2. может выделяться не сразу, а в процессе работы программы, освобождение памяти

	производится вручную 3. выделяется каждый раз при обращении к переменной
21.	Деструктор перегружать: 1. нельзя 2. можно, если перегружается конструктор 3. обязательно, если перегружается конструктор
22.	Сверхвозрастающая последовательность является входным параметром для программы: 1. ECDH 2. «Алгоритм RSA» 3. «Алгоритм рюкзака»
23.	В программе «Вычисление мультипликативной инверсии» используется: 1. усовершенствованный шифр Цезаря 2. свойство вложенных циклов 3. алгоритм RSA
24.	Трудоемкость программы (алгоритма) – это: 1. зависимость количества массовых операций от объема (размерностей) обрабатываемых данных 2. количество итераций во всех циклах программы 3. время, затраченное на обработку данных
25.	Механизм виртуальных базовых классов необходим для того, чтобы: 1. упростить запись обращения к элементам базового класса 2. исключить неоднозначность при многократном косвенном наследовании 3. исключить неоднозначность при многократном прямом наследовании
26.	Если указатель базового класса указывает на объект производного класса, а затем инкрементируется, то он будет указывать: 1. на следующий объект производного класса 2. на следующий объект базового класса 3. на тот объект, на который указывал изначально
27.	С помощью родового класса можно создать класс, реализующий очередь, связанный список для 1. любых типов данных 2. для целочисленных типов данных 3. для строковых типов данных
28.	Благодаря обработке исключительных ситуаций можно: 1. исключить возникновение ошибок во время выполнения программы 2. локализовать синтаксические ошибки в программе 3. упростить управление и реакцию на ошибки во время выполнения программы
29.	Инструкции *catch* могут перехватывать: 1. любые типы данных, включая типы создаваемых программистом классов 2. только типы данных, которые встроены в язык программирования 3. только типы создаваемых программистом классов
30.	Правильное объявление вектора: 1. *vector (string) ivector[25];* 2. *vector <string> ivector;* 3. *vector “string” ivector[25];*
	ВАРИАНТ 3

1.	Перед объявлением виртуальной функции ставится ключевое слово: 1. *derived* 2. *virt* 3. *virtual*
2.	Когда в виртуальной функции базового класса отсутствует значимое действие, она является чистой виртуальной функцией, то: 1. должна быть переопределена в производном классе 2. ее переопределение в производном классе необязательно 3. она только объявляется в производном классе, но не переопределяется
3.	Абстрактный класс является технически неполноценным, потому что: 1. содержит чистую виртуальную функцию, которая объявлена, но не определена 2. не имеет полей 3. не имеет методов
4.	Родовая функция создается с помощью ключевого слова: 1. *template* 2. *temporary* 3. *temp*
5.	Входные данные для программирования секретных ключей по алгоритму ECDH: 1. координаты примитивного элемента, уравнение эллиптической кривой 2. секретные ключи абонентов, координаты примитивного элемента, уравнение эллиптической кривой 3. секретные ключи абонентов, уравнение эллиптической кривой
6.	Формулы для операций с точками эллиптической кривой корректны при условии: *1. $p\neq0$ и $p\neq3$.* *2. $p\neq0$.* *3. $p\neq2$ и $p\neq3$.*
7.	В рамках хорошего стиля программирования необходимы: 1. мнемоника и выравнивание текста при кодировании программы 2. достаточно следовать правилам составления идентификаторов DOS 3. необходимо избегать комментариев
8.	Жизненный цикл программного продукта – это интервал времени от момента: 1. написания программы до момента ввода ее в эксплуатацию 2. возникновения объективной необходимости в программе до момента изъятия ее из эксплуатации 3. ввода программы в эксплуатацию до возникновения в ней ошибки в процессе работы
9.	Наследование – это: 1. создание на базе существующих классов новых классы, получающих от базовых классов свойства и методы 2. создание на базе существующих классов новых классы, получающих от базовых классов свойства и методы 3. возможность использовать созданные классы в нескольких программах
10.	Параметры функции по умолчанию должны быть: 1. локальными или глобальными переменными 2. константами или локальными переменными 3. константами или глобальными переменными
11.	Trial – это: 1. бесплатные, свободно распространяемые, программы

	2. программное обеспечение, которое является полнофункциональным в течение определенного времени или количества запусков 3. некоммерческие программы, которые могут использоваться, как правило, бесплатно
12.	Тестирование – это 1. это процесс выполнения программы на некотором наборе данных 2. обнаружение и исправление ошибок в программе 3. компиляция и запуск программы
13.	Разыменовать указатель – значит: 1. поменять имя указателя 2. обратиться к объекту, на который указывает указатель 3. поменять переменную, на которую указывает указатель
14.	inline-функции используются для: 1. того, чтобы функция могла вызвать сама себя 2. написания компактных программ 3. улучшения производительности за счет уменьшения издержек на вызов функции
15.	Для сравнения двух векторов потребуется: 1. применить оператор *if* без помещения его в цикл 2. конструкцию *if* поместить в цикл 3. применить цикл без конструкции *if*
16.	В C++ выделенную динамическую память освобождает оператор: 1. delete 2. erase 3. kill
17.	Из следующих вариантов вычисления Y быстрее выполнится: 1. X = 2*Y + (A-1)/P + 2*T 2. X = 2*(Y + T) + (A-1)/P 3. X = 2*Y+2*T+A/P-1/P
18.	Правильные соответствия между спецификатором базового класса, спецификатором доступа в объявлении производного класса и правами доступа производного класса к элементам базового; спецификатор доступа: 1. public; в базовом классе – private; права доступа в производном классе – protected 2. private; в базовом классе – public; права доступа в производном классе – public 3. любой; в базовом классе – private; права доступа в производном классе – нет прав
19.	Шаблон функции – это: 1. прототип функции, в котором вместо имен параметров указан условный тип 2. определение функции, в котором указаны возможные варианты типов обрабатываемых параметров 3. определение функции, в котором в прототипе указан условный тип, а в определении указаны варианты типов обрабатываемых параметров 4. определение функции, в которой типу обрабатываемых данных присвоено условное обозначение
20.	Полиморфизм реализован через механизмы: 1. перегрузки функций, наследования методов, шаблонов; 2. наследования методов, виртуальных функций, шаблонов 3. перегрузки функций, наследования, виртуальных функций. 4. перегрузки функций, виртуальных функций, шаблонов

21.	Перегрузка конструктора – это обеспечение возможности: 1. выбора способа инициализации полей класса 2. добавлять в конструктор обращение к методам класса 3. исключать из класса поля, которые не нуждаются в инициализации
22.	Сверхвозрастающая последовательность: 1. вычисляется в программе «Алгоритм рюкзака» 2. является входным параметром для программы «Алгоритм рюкзака» 3. вычисляется в программе «Алгоритм RSA»
23.	Нормальная последовательность при программировании «Алгоритм рюкзака»: 1. вычисляется в программе 2. является входным параметром для программы 3. является конечным результатом в работе программы
24.	На оценку трудозатрат при разработке программного влияет следующая совокупность факторов: 1. ресурсные факторы, факторы технологий разработки 2. сложность решаемой проблемы, факторы технологий разработки 3. человеческий фактор, ресурсные факторы, сложность решаемой проблемы, факторы технологий разработки
25.	Указатель базового класса может указывать на объект производного класса: 1. и указатель производного класса может указывать на объект базового класса 2. обратный порядок недействителен 3. и указатель производного класса может указывать на объект базового класса при условии наследования со спецификатором protected
26.	При переопределении виртуальной функции в производном классе: 1. ключевое слово *virtual* не требуется 2. ключевое слово *virtual* обязательно 3. виртуальная функция становится обычной функцией, поэтому ключевое слово *virtual* не требуется
27.	Функции-члены родового класса: 1. становятся родовыми, если для них явно задавать ключевое слово *template* 2. сами автоматически становятся родовыми, для них не обязательно явно задавать ключевое слово *template* 3. остаются обычными функциями, для них не задать ключевое слово *template*
28.	Если исключительная ситуация имеет место внутри блока, она возбуждается: 1. (ключевое слово *try)*, перехватывается (ключевое слово *throw)* и обрабатывается 2. (ключевое слово *try)*, перехватывается (ключевое слово *catch)* и обрабатывается 3. (ключевое слово *throw*), перехватывается (ключевое слово *catch*) и обрабатывается
29.	Объекты *vector*: 1. изменяют свои размеры при вставке или добавлении значений к ним, поэтому можно при объявлении объекта *vector* указать размер *0* 2. работают только при указании их размера при объявлении 3. не могут изменять размеры, указанные при объявлении, поэтому необходимо зарезервировать достаточное количество элементов при объявлении вектора
30.	Метод вектора *end()* возвращает итератор, который ссылается на: 1. последний элемент контейнера 2. область памяти, следующую за последним элементом контейнера 3. предпоследний элемент контейнера

Приложение № 2

ТЕМЫ И ОБРАЗЦЫ ЗАДАНИЙ ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ (семестр 9)

Тема 1. Численные методы. Программирование метода дихотомии. Программирование метода Ньютона.
1.1 Запрограммировать метод дихотомии и протестировать программу на примере уравнения $x^3+x-1=0$ на отрезке $[0; 1]$ с точностью $\varepsilon=0,01$
1.2 Запрограммировать метод Ньютона и протестировать программу на примере уравнения $x^3+x-1=0$ на отрезке $[0; 1]$ с точностью $\varepsilon=0,01$

Тема 2. Аффинный шифр и методы его программирования.
2.1 Запрограммировать аффинное шифрование и протестировать программу на примере своей фамилии.
2.2 Запрограммировать аффинное дешифрование, объединить 2.1 и 2.2 в одну программу.

Тема 3. Структура класса. Спецификаторы доступа.
3.1 Создать класс, содержащий фамилию, пол и год рождения в закрытой части класса. Включите в класс открытую функцию для ввода этих данных и открытую функцию для вывода данных на экран.

Тема 4. Программирование конструкторов с параметрами.
4.1 Создать класс, конструктору которого передаются три значения типа *float*, являющиеся длинами сторон треугольника. В классе создать: 1) функцию, которая вычисляет площадь треугольника по формуле Герона; 2) функцию, выводящую результат на экран.

Тема 5. Программирование конструкторов с параметрами.
5.1 Создать класс, конструктору которого передаются три значения типа *float*, являющиеся длинами сторон треугольника. В классе создать: 1) функцию, которая вычисляет площадь треугольника по формуле Герона; 2) функцию, выводящую результат на экран.

Тема 6. Управление доступом к базовому классу.
6.1 Создать базовый класс *area_cl,* **открытые** члены которого – основание и высота геометрической фигуры. Создать производные классы *rectangle* и *isosceles.* Каждый производный класс должен включать в себя функцию *area(),* которая возвращает площадь соответственно прямоугольника и равнобедренного треугольника и функцию *show(),* выводящую полученные значения на экран. Для инициализации высоты и длины основания используйте конструктор с параметрами.

ТЕМЫ И ОБРАЗЦЫ ЗАДАНИЙ ДЛЯ ЛАБОРАТОНЫХ ЗАНЯТИЙ (семестр А)

Тема 7. Встраиваемые функции.
7.1 Используя класс *stack* из примера 5.4 (лабораторная работа №5), добавьте в программу функцию *showstack(),* которой в качестве аргумента передается объект типа *stack.* Эта функция должна выводить содержимое стека на экран.

Тема 8. Дружественные функции.
8.1 Создать классы *parall* и *rectang*; оба класса содержат закрытые переменные – длину и ширину геометрической фигуры. Оба класса имеют конструктор и функцию, которая вычисляет

площадь параллелограмма в классе *parall* и площадь прямоугольника в классе *rectang*. Функция *pl_greater()* дружественна для обоих классов. Эта функция возвращает положительное число, если площадь параллелограмма больше площади прямоугольника; нуль, если их площади одинаковы; отрицательное число, если площадь параллелограмма меньше площади прямоугольника.
8.2 Выполнить задания 8.1, при условии, чтобы функция *pl_greater()* была членом одного класса и дружественной другому.

Тема 9. Множественное наследование.
9.1 Создать производный класс *bilet*, содержащий закрытый член класса для хранения стоимости билета. Базовые классы: класс *bus* содержит количество посадочных мест и количество колес; класс *gorod* – конечный пункт и время нахождения в пути. Наследование обоих классов прямое! Класс *bilet* содержит метод, выводящий поля всех классов на экран. Конструктор есть только у производного класса

Тема 10. Виртуальные функции.
10.1 Создать базовый класс *dimen*, в котором хранятся катеты прямоугольного треугольника. В классе *dimen* также объявляется виртуальная функция *pl_gip(),* которая, при ее подмене в производном классе *pl* возвращает площадь треугольника, а при ее подмене в производном классе *gip* возвращает его гипотенузу. Во всех трех классах создать конструкторы. К виртуальной функции обратиться через указатель.

Тема 11. Родовые функции.
11.1 Напишите родовую функцию *min()*, возвращающую меньший из двух своих аргументов. Например, версия функции *min(3, 4)* должна возвратить *3*, а версия *min('c', 'a') – a.*

Тема 12. Вычисление и программирование порядка точки на эллиптической кривой.
12.1 Написать программу поиска порядка точки эллиптической кривой и протестировать ее на примере: *P(0,1)* в группе $y^2=x^3+1$ над полем *GF(5).*

Тема 13. Вычисление секретных ключей по алгоритму ECDH.
13.1 Написать программу вычисления секретных ключей и протестировать ее на примере: $y^2 = x^3 + 2x + 2\ mod 17$. Примитивный элемент равен *P=(5,1).* Секретный ключ Алисы *c=3,* секретный ключ Боба *d=10.*

Приложение №3.

ВОПРОСЫ К ЭКЗАМЕНУ (семестр А)

1. Объявление конструктора производного класса и особенности его работы.
2. Понятие множественного наследования. Прямое наследование.
3. Понятие множественного наследования. Косвенное наследование.
4. Виртуальные базовые классы.
5. Указатели на производные классы.
6. Виртуальные функции.
7. Чистые виртуальные функции.
8. Абстрактные классы.
9. Родовые функции.
10. Обработка исключительных ситуаций.
11. Обработка исключительных ситуаций, возбуждаемых оператором *new*
12. Программная реализация шифрования по таблице Виженера.
13. Программная реализация алгоритма Диффи-Хеллмана.
14. Группа точек эллиптической кривой.
15. Законы сложения точек эллиптической кривой.
16. Программная реализация вычисления порядка точки в группе эллиптической кривой.
17. Схема открытого распределения ключей Диффи-Хеллмана над группой точек эллиптической кривой.
18. Программная реализация вычисления секретного ключа на эллиптической кривой.
19. Шифра Шамира. Программная реализация
20. Шифр Эль-Гамаля. Программная реализация.
21. ЭЦП. Проверка подлинности подписи с помощью online-калькулятора.
22. Анализ сложности алгоритмов. Модель RAM (Random Access Machine).
23. Анализ сложности алгоритмов. Подсчет операций. Классы входных данных.
24. Примеры анализа алгоритмов.
25. Математический аппарат анализа алгоритмов
26. Сравнение сложности алгоритмов. Пределы.
27. Логарифмы и сложность алгоритмов
28. Программное логирование сетевых пакетов в файл
29. Программная реализация обнаружения заданной сетевой атаки,
30. Автоматическое реагирование и логирование подозрительной активности.

ПРИМЕРЫ К ЭКЗАМЕНУ (семестр А)

1. Дана программа, содержащая некоторый класс, переделайте все соответствующие обращения к членам класса так, чтобы в них явно присутствовал указатель *this*.

2. Создать классы *tryan* и *rectan*; оба класса содержат закрытые переменные: *rectan* – длину и ширину прямоугольника, *tryan* – катеты прямоугольного треугольника. Оба класса имеют конструктор и функцию, которая вычисляет периметр соответствующей фигуры. Функция *per_greater()* дружественна для обоих классов, она вычисляет кратность периметров друг другу. *Примечание:* оператор % работает с целочисленными переменными.

3. Выполнить задание 2, при условии, чтобы функция *pl_greater()* была членом одного класса и дружественной другому.

.4. Создать производный класс *familia,* содержащий закрытый член класса для хранения фамилии. Наследование косвенное: классу *familia* предшествует класс *shtat,* который содержит должность и зарплату, а классу *shtat* предшествует класс *sotrudnik,* содержащий год рождения и пол. Все три класса содержат конструкторы.

5. Напишите программу, в которой базовый класс *dist* используется для хранения *double* расстояния между двумя точками. В классе *dist* создайте виртуальную функцию *trav_time(),* которая выводит на экран время, необходимое для прохождения этого расстояния с учетом того, что расстояние задано в милях, а скорость равна *60* миль в час. В производном классе *metric* переопределите функцию *trav_time()* так, чтобы она выводила на экран время, необходимое для прохождения этого расстояния, считая теперь, что расстояние задано в километрах, а скорость равна *100* километров в час..

6. Создать родовую функцию *sum,* возвращающую сумму элементов массива.

7.Создать родовую функцию *mas_otr*, выводящую на экран только отрицательные элементы массива.

8. Создать базовый класс *dimen,* в котором хранятся катеты прямоугольного треугольника. В классе *dimen* объявляется чистая виртуальная функция *pl_gip(),* которая, при ее подмене в производном классе *pl* возвращает площадь треугольника, а при ее подмене в производном классе *gip* возвращает его гипотенузу. Во всех трех классах создать конструкторы. К виртуальной функции обратиться через указатель.

9. Получить с экрана координаты двух точек эллиптической кривой и просуммировать их.

10. Написать программу поиска порядка точки эллиптической кривой и протестировать ее на примере, предложенном преподавателем.

11. Показать свою программу вычисления секретных ключей в алгоритме ECDH и протестировать ее на примере, предложенном преподавателем.

Приложение № 4

ПРИМЕРНЫЕ ТЕМЫ КУРСОВЫХ РАБОТ

1. Оценка защищенности компьютерной системы университета на основе ОС WIndows в соответствии с требованиями руководящих документов ФСТЭК России.
2. Оценка защищенности компьютерной системы офиса коммерческой организации на основе ОС Windows в соответствии с требованиями руководящих документов ФСТЭК России.
3. Оценка защищенности компьютерной системы офиса коммерческой организации на основе ОС Linux в соответствии с требованиями руководящих документов ФСТЭК России.
4. Оценка защищенности компьютерной системы университета на основе ОС Windows в соответствии с требованиями «Оранжевой книги».
5. Оценка защищенности компьютерной системы университета на основе ОС Linux в соответствии с требованиями «Оранжевой книги».
6. Оценка защищенности компьютерной системы офиса коммерческой организации на основе ОС Windows в соответствии с требованиями «Оранжевой книги».
7. Оценка защищенности компьютерной системы офиса коммерческой организации на основе ОС Linux в соответствии с требованиями «Оранжевой книги».
8. Оценка защищенности ОС Windows в соответствии со стандартами ISO («Общими критериями»).
9. Оценка защищенности ОС Linux в соответствии со стандартами ISO («Общими критериями»).
10. Сравнительный анализ антивирусных пакетов.
11. Анализ методов изучения поведения нарушителей безопасности компьютерных систем.
12. Сравнение анализаторов безопасности компьютерных систем.
13. Сравнительный анализ средств защиты электронной почты.
14. Анализ методов перехвата паролей пользователей компьютерных систем и методов противодействия им.
15. Анализ методов нарушения безопасности сетевых ОС и методов противодействия им.
16. Анализ методов организации антивирусной защиты компьютерных систем.
17. Сравнительный анализ персональных брандмауэров.
18. Анализ средств безопасности в пакете MS Office.
19. Анализ средств защиты от спама.
20. Анализ методов повышения надежности хранения информации на жестких магнитных дисках.
21. Анализ методов обеспечения безопасности электронного магазина.
22. Анализ методов обеспечения безопасности домашней сети.
23. Сравнительный анализ средств защиты компакт-дисков от несанкционированного копирования.
24. Анализ методов гарантированного удаления конфиденциальной информации на электронных носителях.
25. Использование языка сценариев ОС Windows для разграничения прав пользователей компьютерных систем.
26. Разработка профиля защиты в соответствии с требованиями «Общих (единых) критериев».
27. Сравнительный анализ средств шифрования файлов.
28. Сравнительный анализ средств шифрования дисков.

29. Сравнительный анализ методов и средств создания защищенных частных сетей (VPN).
30. Сравнительный анализ средств контроля содержимого WEB-трафика (контент анализа).
31. Сравнительный анализ методов и средств обеспечения безопасности и конфиденциальности в обозревателях Интернета.