МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

А.А. МЕРСОВ, А.М. РУСАКОВ, В.В. ФИЛАТОВ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Москва – 2023

УДК
ББК

Мерсов А.А., Русаков А.М., Филатов В. В.
Методические указания по выполнению курсовой работы по языкам программирования / А.А. Мерсов, А.М. Русаков, В. В. Филатов,– М.: МИРЭА – Российский технологический университет, 2023.

Методические указания предназначены для выполнения курсовой работы (проекта) по дисциплине «Языки программирования» и содержит перечень вариантов курсовой работы (проекта), требования к оформлению пояснительной записки, а также краткое изложение теоретического материала в форме пояснений к заданию на работу (проект). Для студентов, обучающихся по направлениям 09.03.02, 10.03.01, 10.05.02, 10.05.03, 10.05.04.

В методических указаниях изложен порядок выполнения курсовой работы по языкам программирования.

Материалы рассмотрены на заседании учебно-методической комиссии КБ-2 Протокол №7 от «14» марта 2023 г.
и одобрены на заседании кафедры КБ-2.

зав. кафедрой КБ-2
к.т.н., доцент _____ / О.В. Трубиенко /

Оглавление

ВВЕДЕНИЕ 5

2 ОБЩИЕ ПОЛОЖЕНИЯ 6

3 ОБЩИЕ ТРЕБОВАНИЯ К ПОРЯДКУ ВЫПОЛНЕНИЯ И СОДЕРЖАНИЮ РАБОТЫ 8

3.1 Структура курсовой работы 8

3.2 Общие рекомендации по разработке программной части 9

3.3 Исходные данные программы 9

3.4 Особенности работы разрабатываемой программы 10

3.5 Выходные данные программы 10

3.6 Объем и содержание разделов пояснительной записки 11

4 ТРЕБОВАНИЯ ПО ОФОРМЛЕНИЮ РАБОТЫ 15

4.1 Общие требования 15

4.2 Правила оформление курсовой работы 15

4.3 Оформление структуры пояснительной записки 16

4.4 Нумерация страниц пояснительной записки 17

4.5 Списки и перечисления 17

4.6 Нумерация иллюстраций 18

4.7 Оформление таблиц 19

4.8 Уравнения и формулы в пояснительной записке 20

4.9 Список использованных источников 21

4.10 Приложения 21

5 ПОРЯДОК ЗАЩИТЫ КУРСОВОЙ РАБОТЫ 22

6 КРИТЕРИИ ОЦЕНКИ КУРСОВОЙ РАБОТЫ 24

6.1 Оценка сформированности компетенций 24

6.2 Критерии определения оценки за курсовую работу 27

7 ВАРИАНТЫ ЗАДАНИЙ 29

8 СВЕДЕНИЯ ИЗ ТЕОРИИ КРИПТОГРАФИИ 43

8.1 Общие сведения 43

8.2 Основные понятия и термины 43

8.3 Основы работы с CryptoAPI 44

8.3.1 Криптопровайдеры 45

8.3.2 Шифрование 49
8.3.3 Расшифрование 50
8.3.4 Криптографические ключи и выполняемые функции 51
8.3.5 Пример программы с использованием CryptoAPI 54
8.4 Основы использования OpenSSL 56
8.4.1 Установка OpenSSL 56
8.4.2 Последовательность действий 59
8.4.3 Команды OpenSSL 60
8.4.4 Используемые команды 60
8.4.5 Код программы 61
8.4.6 Реализация функции *Crypt ( )* 61
8.4.1 Реализация функции *Decrypt( )* 62
8.4.2 Полный листинг кода 63
СПИСОК ЛИТЕРАТУРЫ 66
ПРИЛОЖЕНИЯ 67
Примеры оформления таблиц 68
Примеры оформления формул 69
Примеры оформления источников 70

ВВЕДЕНИЕ

В настоящее время квалификация специалиста по информационной безопасности требует глубокие знания в области информационных технологий, включая практические навыки по разработке программного обеспечения, структур данных и алгоритмов их обработки, понимания принципов организации программных модулей, их взаимодействие с программно-аппаратным окружением, устройствами ввода-вывода информации, её организации хранения как в оперативной памяти, так и во внешней – в виде файлов или баз данных. Как правило, этот материала рассматривается как отдельные, самостоятельные вопросы (темы) учебных занятий. Поэтому в рамках самостоятельной работы, в частности, при выполнения курсовой работы, студенту предлагаются задания, которые требуют от студента умения агрегировать отдельные умения и знания в виде единой программы, реализующей практическую задачу из области профессиональной деятельности, требующую знания из различных областей, а именно – информационной безопасности, кодирования и информационных технологий, что является актуальным на сегодняшний момент времени.

В своей профессиональной деятельности специалист в области информационной безопасности в обязательном порядке будет сталкиваться с различными алгоритмическими и программными криптографическими средствами. В курсовой работе студенту в учебных целях предлагается реализовать подобное программное средство. Рассматриваемые алгоритмы кодирования и шифрования носят учебный характер, позволяющие студенту получить знания и практический опыт их применения.

Предлагаемые методические рекомендации следует использовать при выполнении курсовой работы по дисциплине «Языки программирования», изучаемой по направлению 10.03.01 Информационная безопасность и специальностям 10.05.03 Информационная безопасность автоматизированных систем, 10.05.04 Информационно-аналитические системы безопасности, 10.05.05 Безопасность информационных технологий в правоохранительной сфере.

В данном пособии будут рассмотрены следующие вопросы:

1) общие требования к порядку выполнения и содержанию работ;

2) требования по оформлению работы;

3) варианты курсовой работы;

4) краткие сведения из теории, необходимые для выполнения курсовой работы;

5) перечень рекомендуемой литературы и информационных источников.

1 ОБЩИЕ ПОЛОЖЕНИЯ

Курсовая работа является формой самостоятельной работы обучающихся под руководством преподавателя и представляет собой творческую, самостоятельную работу обучающихся, имеющую целью формирование у них компетенций с учетом требований Федеральных государственных образовательных стандартов высшего образования по направлению (специальности) подготовки.

Настоящие методические рекомендации по выполнению курсовой работы разработаны в соответствии с Федеральным законом от 29 декабря 2012 г. № 273-ФЗ «Об образовании в Российской Федерации», приказом Министерства образования и науки Российской Федерации от 05 апреля 2017 г. № 301 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования - программам бакалавриата, программам специалитета, программам магистратуры». Уставом и локальными нормативными актами Университета.

Основной целью курсовой работы является формирование и закрепление компетенций путём практического использования знаний, умений и навыков, полученных в рамках теоретического обучения, а также выработка самостоятельного творческого подхода к решению конкретных профессиональных задач в области разработки прикладного программного обеспечения.

Курсовая работа выполняется лично студентом под руководством преподавателя.

Закрепление тем курсовой работы за обучающимися и назначение руководителей производится распоряжением заведующего кафедрой и обеспечивается соответствующим бланком задания на курсовую работу, утверждаемым заведующим кафедрой.

Список примерных (типовых) тем курсовой работы приведен в соответствующем разделе конкретной ООП (рабочей программе дисциплины «языки программирования»). Тематика курсовой работы по дисциплине «Языки программирования» актуализируется и утверждается на заседании кафедры КБ-2 «Прикладные информационные технологии». В данных методических рекомендациях приводится одна из таких типовых тем, а также варианты конкретного задания.

Перечень вариантов курсовой работы доводится до сведения обучающихся в течение первых трех недель семестра, в котором предусмотрено учебным планом выполнение курсовой работы. Выбор номера варианта курсовой работы носит случайный характер. Для упорядочивания порядка определения конкретного номера варианта используется следующий принцип: ***номер варианта курсовой работы есть младшие два числовых разряда номера зачетной книжки студента, выполняющего курсовую работу***.

Обучающийся имеет право выбора темы из списка, предложенного кафедрой, которое оформляется личным заявлением на имя заведующего кафедрой. Обучающийся может предложить свою тему при условии обоснования ее целесообразности. Темы курсовой работы обучающихся должны быть определены не позднее трех недель с начала соответствующего семестра.

По обоснованному решению кафедры данный срок может быть в виде исключения изменен.

Копии распоряжения заведующего кафедрой (выписки из протокола заседания кафедры) передаются в учебно-методический отдел института кибербезопасности и цифровых технологий для учета и внесения в информационно-аналитическую систему «Университет».

Решением кафедры допускается изменение темы курсовой работы по личному заявлению обучающегося, согласованному с руководителем и заведующим кафедрой, при этом оформляется новое задание на курсовую работу и издается соответствующее распоряжение заведующего кафедрой.

Лучшие курсовые работы, представляющие учебно-методическую ценность, могут быть использованы в качестве учебных пособий (с указанием авторства), либо могут быть представлены на соответствующие конкурсы студенческих работ.

Практические работы, связанные с курсовой работой, выполняются с использованием персональных компьютеров. Указания по технике безопасности совпадают с требованиями, предъявляемыми к пользователю ЭВМ. Другие опасные факторы отсутствуют.

2 ОБЩИЕ ТРЕБОВАНИЯ К ПОРЯДКУ ВЫПОЛНЕНИЯ И СОДЕРЖАНИЮ РАБОТЫ

2.1 Структура курсовой работы

Курсовая работа по программированию состоит из двух частей:

- программы;
- пояснительной записки к курсовой работе, описывающий процесс разработки программы и являющейся отчетом о её выполнении.

Программа представляет собой отлаженное программное обеспечение, реализующее задачу, указанную в варианте задания на курсовую работу. Отлаженное программное обеспечение представляет собой полный комплект как исходных текстов программы, созданный лично студентом при выполнении курсовой работы, так и используемых стандартных и сторонних библиотек, используемых студентом в своей работе. Указанный комплект исходников должен обеспечивать сборку и компиляцию программного проекта, с целью запуска и демонстрации работы программы. ***Комплект исходников предоставляется и сдается на электронном носителе (CD-ROM, DVD_ROM или USB-накопитель)***.

Пояснительная записка к курсовой работы должна содержать описание всех этапов разработки соответствующей программы, от проработки теоретического аспекта задания, до контрольного примера запуска программы, подтверждающего состоятельность разработанных алгоритмов и программных средств. ***Пояснительная записка к курсовой работе предоставляется в электронном (на носителе) и печатном варианте.***

Пояснительная записка к курсовой работе должна иметь следующую структуру:

титульный лист (приложение № 1);

задание на курсовую работу (приложение № 2);

АННОТАЦИЯ;

ОГЛАВЛЕНИЕ;

ВВЕДЕНИЕ;

основная часть включает следующие разделы:

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ (теоретические сведения, связанные с темой, зданием, а также используемыми технологиями и программными решениями, используемыми при разработке программы, в части параметров оценивания курсовой работы);

2 ПРАКТИЧЕСКАЯ ЧАСТЬ (состоит из подразделов):

2.1 Структура программы (общий алгоритм исполнения – структурное и/или модульное программирование и/или общая структура программы, включая вопросы объектно-ориентированного программирования);

2.2 Алгоритм решения задачи (реализация варианта задания)

2.3 Алгоритм кодирования (шифрования) / Алгоритм взаимодействия с внешним модулем шифрования;

2.4 Программная реализация и общий комментарий к программному фрагменту (в целом) по каждому параметру оценивания, заявляемых студентом для оценивания комиссией по приему курсовой;

2.5 Разработка интерфейса

2.6 Методика тестирования

2.7 Контрольный пример

ЗАКЛЮЧЕНИЕ (выводы по курсовой работе);

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ;

ПРИЛОЖЕНИЕ к курсовой работе, содержащий полный листинг исходных текстов программ, включенных в программный проект.

2.2 Общие рекомендации по разработке программной части

Для выполнения курсовой работы требуется умение писать программы на алгоритмическом языке для выполнения предложенного задания. Общая постановка и варианты задания на курсовую работу приводятся ***в разделе 6*** настоящих рекомендаций.

Таким образом, последовательность выполнения курсовой работы следующая:

1. Ознакомится необходимым алгоритмом согласно варианту задания;
2. В соответствии с заданным вариантом написать программу;
3. Описать соответствующий алгоритм;
4. Написать и отладить на компьютере программу;
5. Привести контрольную распечатку;
6. Оформить отчет;

Работа считается выполненной только после оформления отчета, защиты и подписи преподавателя.

Последовательность написания программного обеспечения.

Шаг 1. Анализ исходных данных, указанных в задании.

Шаг 2. Определение данных, структур, классов, методов и функций, необходимых для выполнения работы согласно варианту.

Шаг 3. Разработка соответствующего алгоритма решения конкретной задачи.

Шаг 4. Реализация элементов, описанных в Шаге 2.

Шаг 5. Подготовка контрольных данных для тестирования программного обеспечения.

Шаг 6. Отладка разработанного программного обеспечения на основе контрольных данных, подготовленных на Шаге 5.

2.3 Исходные данные программы

1. Информация о группе студентов из N человек, где запись о студенте содержит следующие данные:

1) Фамилия, имя, отчество студента;
2) Число, месяц, год рождения;
3) Год поступления в институт;
4) Факультет (институт)
5) , кафедра
6) Группа.
7) Номер зачетной книжки.
8) Пол
9) Названия предметов и оценки по каждому предмету в каждой сессии, максимально 9 сессий и 10 предметов в каждом семестре, которые могут быть разные.

Все данные должны быть форматными: даты, числа и т.д.

2. Конкретное задание, вариант которого определяется 2-мя последними цифрами зачетной книжки, содержится в разделе 6 настоящих методических рекомендаций.

2.4 Особенности работы разрабатываемой программы

1) Программа должна позволять осуществлять следующие функции: организовать ввод, изменение и удаление данных, т.е. иметь возможность изменять содержимое информационной базы без изменения остальной информации;
2) На работу программы не должны оказывать влияние
    a. неправильные данные
    b. случайно нажатые клавиши и т.п.,

    т.е. программа должна работать в любых ситуациях;
3) Программа должна быть реализована с учетом методов объектно-ориентированного программирования с использованием инкапсуляции, наследования, полиморфизма, перегрузки функции и операторов, использования конструкторов и деструкторов;
4) Иметь файловую структуру.

2.5 Выходные данные программы

Результат работы программы представляет собой вывод следующих данных на экран:

1. данные о студентах по всей группе в порядке ввода информации;
2. в части выполнения конкретного варианта задания необходимо выводить все данные по каждому студенту, начиная от "Ф.И.О." и заканчивая "Текущим семестром" (по необходимости выводить промежуточные данные / промежуточную таблицу).

2.6 Объем и содержание разделов пояснительной записки

Пояснительная записка (отчет о выполнении курсовой работы) оформляется в соответствии с требованиями, предъявляемыми к оформлению курсовых работ в РТУ МИРЭА, содержащиеся в инструкции по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18 и более подробно будет изложено в разделе 3 настоящих методических рекомендациях. Структура пояснительной записки представлена в подразделе 2.1 и будет подробно рассмотрена в данном подразделе.

Пояснительная записка к курсовой работе представляет собой электронный документ в формате *.doc, структура которого представлена в разделе 2.1.

Общий объем курсовой работы, не включая приложения, не должен как правило превышать 50 страниц. Сюда относится

– ***Титульный лист*** пояснительной записки по выполнению курсовой работы (1 страница), оформляется по установленному образцу, приведенному в приложении 1;

– *Типовая форма* ***задания на курсовую работу***, приведена в приложении № 2. При большом объеме пунктов 2 и 3 задания, их продолжение переносится на оборотную сторону листа задания. (1-2 страницы);

– ***Аннотация*** к курсовой работе (рекомендуемый объем – 1-2 страницы) представляет собой краткое изложение содержания курсовой работы с основными выводами, принятыми технологическими решениями и рекомендациями. Также приводятся данные об объеме и содержании курсовой работы (структуре, включающей как исходный код программы, так и пояснительную записку), количестве рисунков, таблиц, приложений, использованных источников в пояснительной записки, объем исходного кода программы, количестве строк кода, лично написанных студентом;

– В ***оглавлении*** приводятся наименования структурных частей курсовой работы, разделов и подразделов его основной части с указанием номера страницы, с которой начинается соответствующая часть, раздел (подраздел).

– Во ***введении*** (рекомендуемый объем – 1-2 страницы) дается общая характеристика курсовой работы, включающей:

1. ***обоснование актуальности*** включения в курсовую работу вопросов информационной безопасности, например, программную реализацию методов защиты персональных данных;
2. определяется цель работы и задачи, подлежащие решению для её достижения;
3. описываются объект и предмет исследования, используемые методы и информационная база исследования;
4. кратко характеризуется структура курсовой работы по разделам;

– *Основная часть* (рекомендуемый объем – от 10 до 40 страниц) состоит из двух разделов: ***теоретическая*** и ***практическая части,*** и в целом содержит материал, необходимый для достижения цели курсовой работы и

решения поставленных задач в процессе выполнения. Содержание основной части должно соответствовать теме (заданию варианта), указанной в задании, и полностью ее раскрывать.

В теоретической части (раздел 1) необходимо сформулировать общие сведения, необходимые для реализации параметров оценивания (см. соответствующую строку таблицы «Критерии оценки). В частности, приводятся все необходимые определения понятий, связанные с заявляемыми параметрами оценивания, включая понятия структурного, процедурного, модульного, объектно-ориентированного программирования и алгоритмов кодирования и шифрования, если такое будет задействовано в программной реализации курсовой работы.

В практической части (раздел 2) приводятся конкретные технологические решения, алгоритмы и/или фрагменты программ (каждый алгоритм или фрагмент кода оформляется в пояснительной записке как рисунок), которые отражают:

2.1 Структура программы, где излагается общий алгоритм исполнения, если студент использует структурное и/или модульное (процедурное) программирование – в этом случае указываются основные модули (подпрограммы, функции), их идентификатор, входные и выходные параметры, функциональное назначение. Если программа построена на основе объектно-ориентированной парадигме программирования, то указываются и описываются основные классы, дерево наследования, видимости свойств и методов класса. Показывается и комментируется реализация множественного наследования, переопределение методов, практическое использование полиморфизма и других характерных элементов объектно-ориентированной парадигмы.

2.2 Алгоритм решения задачи (реализация варианта задания). Данный подраздел должен содержать описание реализации задания варианта курсовой работы как в виде вербального описания алгоритма, так и виде блок-схемы. Допускается отклонение от требований ГОСТ по оформлению блок-схем. В этом случае подобные графические изображения алгоритма следует именовать как «схема алгоритма» и подписывать рисунки соответствующим образом.

2.3 Если студент заявляет в качестве реализованной подзадачи – метод реализации защиты персональных данных студентов (в том числе с использованием внешних программных компонент или библиотек), информация о которых представляет исходными данными для курсовой работы, то необходимо в качестве одного из подразделов включить подраздел (на выбор) ***Алгоритм кодирования (шифрования)*** или ***Алгоритм взаимодействия с внешним модулем шифрования,*** где рассматривается соответствующий вычислительный процесс;

2.4 Программная реализация задания – содержит общие комментарии и пояснения (в целом), а также программные фрагменты (оформленные как рисунки) по каждому заявленному студентом параметру оценивания комиссией по приему курсовой. Здесь рекомендуется использовать третий

уровень нумерации подразделов с включением их в оглавление пояснительной записки курсовой работы. В каждом таком подразделе студент наглядно показывает степен проработки вопроса, подкрепляя слова конкретными фрагментами из своей программы, оформленными как рисунки;

2.5 Разработка интерфейса – содержит описание экранных форм (если реализован графический многооконный интерфейс взаимодействия с пользователем или консольный режим ввода–вывода информации с реализацией соответствующего меню выбора действий пользователя. Здесь поясняются основные экранные области ввода и вывода информации и, при необходимости, – формат файлов ввода-вывода;

2.6 Методика тестирования. В данном подразделе излагается общая последовательность действий разработчика программы, чтобы подтвердить корректность принятых решений. Например, описывается последовательность действий разработчика программы при разработке теста на основе контрольного примера. В этом случае описываются исходные данные, на основе которых доказывается корректность реализованного алгоритма решения, и указывается ожидаемый ответ программы. При этом особое внимание уделяется различным исходам решения, так, чтобы подготовить такие входные данные, с помощью которых можно показать корректность вычислений по всем ветвям и возможным исходам работы программы.

2.6 ***Контрольный пример*** – это, как правило, последовательность скриншотов, доказывающая корректность работы программы. Для этого, подразделе *2.5 Методика тестирования* были предложены (разработаны) входные данные, а также приведен ожидаемый результат. В соответствии с разработанной методикой тестирования в качестве исходных данных программы вводятся предложенные входные значения и фиксируется скриншотом, получаемый результат, который включается в пояснительную записку как рисунок. Подобные доказательства работоспособности алгоритма и программы должны быть приведены для всех возможных исходов работы разработанной студентом программы.

Каждый из разделов (теоретическому и практическому) должен заканчиваться выводами (до 1 страницы), где обобщаются решения описанные и принятые к реализации в курсовой работе.

В целом, обязательным для текста пояснительной записки курсовой работы является логическая связь между разделами и последовательное развитие основной темы на протяжении всей работы, самостоятельное изложение материала, критический подход к изучаемым данным, проведение необходимого анализа, аргументированность выводов, обоснованность предложений и рекомендаций. Также обязательным является наличие в основной части курсовой работы ссылок на использованные источники.

В ***заключении*** (рекомендуемый объём - 1-2 стр.) логически последовательно излагаются теоретические выводы и/или практические

предложения, которые сформулировал студент в результате выполнения курсовой работы.

Список использованных источников отражает степень охвата материала при решении поставленной задачи. Подбор литературы (источников) по теме курсовой работы осуществляется обучающимся самостоятельно. В указанном списке должны быть представлены все виды источников информации: научно-технические библиотеки, книжная литература, нормативная база, электронно-библиотечные системы и Интернет. Количество используемых источников при выполнении курсовой работы определяется обучающимся самостоятельно (рекомендуемое количество от 10 до 30). Перечень должен поддерживать (быть использованным в виде ссылок или цитирования в тексте пояснительной записки) все аспекты, изложенные в теоретической части (разделе), а также основные моменты практической части (например, при обосновании технологии разработки программы, выбора инструментальных средств разработки, сторонних библиотек (при необходимости), методы тестирования и т.п. Обучающийся обязательно должен использовать в том числе и источники, изданные за последние пять лет и материалы, доступные в электронной библиотеке РТУ МИРЭА.

В ***приложениях*** помещается вспомогательный материал (при его наличии), который при включении в основную часть работы осложняет её восприятие (таблицы вспомогательных цифровых данных, инструкции, методики, формы отчетности и других документов и т.п.). В качестве *обязательного приложения* к курсовой работе по программированию должен присутствовать ***полный исходный текст*** разработанной студентом программы. Заметим, что требованием по оформлению документации программного обеспечения является наличие комментариев в тексте программе. Однако, с целью объективного оценивания осведомленности студента об используемых в программе конструкциях языка, умении ориентироваться *в своем* программном коде и понимании используемых синтаксических конструкции как в целом, так и в отдельности – ***какие-либо комментарии в полном исходном программном коде разработанной студентом программы НЕ ДОПУСКАЮТСЯ и расцениваются как легитимация подсказок студенту при защите курсовой работы в силу неуверенной ориентации по тексту программы и понимания и истолкования конструкций языка программирования.***

Студент несет полную ответственность за самостоятельность выполнения и достоверность результатов курсовой работы.

3 ТРЕБОВАНИЯ ПО ОФОРМЛЕНИЮ РАБОТЫ

3.1 Общие требования

Пояснительная записка к курсовой работе предоставляется студентом руководителю в сброшюрованном виде. Брошюровке подлежит текстовая часть – пояснительная записка, выполненная с помощью компьютерного набора и оформленная в соответствии с Рекомендациями по оформлению письменных работ обучающихся (СМКО МИРЭА 7.5.1/03.П.69).

При необходимости курсовая работа оформляется в соответствии с требованиями ГОСТов (ГОСТ Р 30-2003 и др.). Приведем основные из них.

- ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления» (оформление работы);
- ГОСТ 2.105-95 «Общие требования к текстовым документам»; (представление текстового, табличного, формульного и иллюстративного материала);
- ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления» (оформление списка использованных источников);
- ГОСТ Р 7.0.5-2008 «Библиографическая ссылка. Общие требования и правила составления» (оформление сносок и ссылок);
- ГОСТ 7.82-2001 «Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления»;
- ГОСТ 7.12-93 «Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила» (использование общепринятых сокращений русских слов и сочетаний).

3.2 Правила оформление курсовой работы

Текст пояснительной записки выполняется в текстовом редакторе для печати на обычной белой офисной бумаге формата А4 на одной стороне листа и следующих параметрах документа:

- Размер кегля: 14пт;
- Шрифт основного текста: Times New Roman;
- Межстрочный интервал: 1,5 строки;
- Абзацный отступ (красная строка): 1,25 см;
- Левое поле: 30 мм;
- Правое поле: 15 мм;
- Верхнее и нижнее поля: 20 мм;

Текст пишется на русском языке, но наименования и собственные имена рекомендуется писать на языке оригинала.

Для уменьшения трудоемкости при оформлении пояснительной записки и обеспечение минимизации затрат при изменении вариантов оформления документов рекомендуется формировать пояснительную

записку в виде структурированного документа с заданным набором стилей для представления различный данных в документе.

3.3 Оформление структуры пояснительной записки

В процессе работы над пояснительной запиской студент структурирует всю основную информацию. Структура пояснительной записки была приведена в разделе 2.1, где выделяется три уровня структуризации:

а) первый уровень – ***раздел.*** Выделяют следующие разделы пояснительной записки: аннотация, оглавление, введение, теоретическая часть, практическая часть, заключение, список использованных источников, приложение;

б) второй уровень – ***подраздел***. Структуризацию *на подразделы теоретической части* определяет студент совместно с преподавателем в зависимости от рассматриваемого материала. Структуризация практического раздела состоит из следующих подразделов: «Структура программы», «Алгоритм решения задачи», «Алгоритм кодирования (шифрования)» / «Алгоритм взаимодействия с внешним модулем шифрования», «Программная реализация», «Разработка интерфейса», «Методика тестирования», «Контрольный пример»;

в) третий уровень – подраздел третьего уровня или ***пункт***. Используется при необходимости разделить подраздел на отдельные самостоятельные части.

Наименования структурных элементов первого уровня «АННОТАЦИЯ», «ОГЛАВЛЕНИЕ», «ВВЕДЕНИЕ», «ТЕОРЕТИЧЕСКАЯ ЧАСТЬ», «ПРАКТИЧЕСКАЯ ЧАСТЬ», «ЗАКЛЮЧЕНИЕ», «СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ», «ПРИЛОЖЕНИЕ» служат заголовками структурных элементов пояснительной записки, которые формируются *прописными буквами без точки в конце в центре строки без абзацного отступа*. Каждая структурная единица начинается с новой страницы. Разделы «ТЕОРЕТИЧЕСКАЯ ЧАСТЬ» и «ПРАКТИЧЕСКАЯ ЧАСТЬ» соответственно нумеруются арабскими цифрами, после которых точка не ставится, например:

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Каждый раздел начинается с новой страницы.

Наименование раздела оформляется шрифтом Times New Roman, начертание – жирное (bold), размер шрифта – 16 пунктов. После наименования раздела как абзаца необходимо установить отступ 6 пунктов.

Основная часть пояснительной записки по выполнению курсовой работы делится на подразделы, а подразделы делятся на пункты. Пункты, при необходимости, могут делиться на подпункты. При делении текста пояснительной записки на пункты и подпункты необходимо, чтобы каждый пункт содержал законченную информацию.

Заголовки подразделов и пунктов начинаются с абзацного отступа после порядкового номера без точки. Подразделы, пункты и подпункты следует нумеровать арабскими цифрами, например

2.5 Программная реализация задания

Наименование подразделов, пунктов и подпунктов оформляется шрифтом Times New Roman, начертание – жирное (bold), размер шрифта – 14 пунктов. До и после наименования подраздела как абзаца необходимо установить отступы в 6 пунктов.

Переносы слов в заголовках недопустимы.

Заметим, что на основе структуры заголовков как стилей современные текстовые редакторы позволяют формировать оглавление в автоматическом режиме с указанием номеров страниц. При оформлении пояснительной записки эту возможность необходимо использовать во избежание несоответствия как по названию, так и по нумерации страниц между оглавлением и заголовками в тексте пояснительной записке. В оглавление следует включат заголовки 1, 2 и 3 уровней, то есть подпункты в оглавление не включаются.

3.4 Нумерация страниц пояснительной записки

Страницы отчета следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту пояснительной записки. Номер страницы проставляют в центре нижней части листа без точки. Размер кегля – 14пт; шрифт – Times New Roman.

Титульный лист и задание на курсовую работу включают в общую нумерацию страниц отчета. Номер страницы на титульном листе и задании на курсовую работу не проставляют.

Иллюстрации и таблицы, расположенные на отдельных листах, включают в общую нумерацию страниц пояснительной записки.

Приложения также включают в общую нумерацию страниц.

3.5 Списки и перечисления

Списки формируются с отступом 1,25см.

Перед каждым элементом перечисления допускается использование только следующих знаков выделения: "–" (тире или символ «минус», но не «дефис», *пример 1*); строчных символов русского алфавита (без символов ё, з, й, щ, ч, ъ, ы, ь) с символом")" (скобка) и арабских цифр со знаком ")" (скобка).

Использование строчных символов русского алфавита допускается только при необходимости ссылки в тексте пояснительной записки на один из элементов списка (перечисления). Использование арабских цифр допускается только при наличии конкретного числа элементов в списке.

После знака выделения добавляется пробел.

Простые перечисления отделяются запятой (*пример 1)*, сложные отделяются точкой с запятой (*примеры 2 и 3*).

Пример 1:

- задание на курсовую работу выполнено полностью,
- задание на курсовую работу выполнено с замечаниями,

– задание на курсовую работу частично.

Пример 2:

1) задание на курсовую работу выполнено полностью;
2) задание на курсовую работу выполнено с замечаниями, представленными в приложении;
3) задание на курсовую работу выполнено частично.

Пример 3:

а) задание на курсовую работу выполнено полностью,
б) задание на курсовую работу выполнено с замечаниями,
в) задание на курсовую работу выполнено частично.

3.6 Нумерация иллюстраций

Пояснительная записка к курсовой работе может включать иллюстративный материал, выполненный как нумерованная иллюстрация со ссылками на нее в тексте пояснительной записки, так и в виде простой вставки в текст, если иллюстративный материал имеет малый объем и на него не требуются в пояснительной записке дополнительные ссылки.

Допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой. Например, Рисунок 1.1.

Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст). Слово "Рисунок" и наименование помещают после пояснительных данных и располагают под рисунком в центре ***без отступа в начале*** и ***точки в конце***, со следующим набором параметров: шрифт – Times New Roman, размером шрифта – 13 пт (допускается размер основного текста 14 пт); абзац без отступа с выравниванием по центру и межстрочным интервалом 1 или 1,5. Допускается подпись как с наклоном, так и без. Для выделения рисунка задается интервал после наименования рисунка – 6 пт. В рамках пояснительной записке необходим придерживаться единого стилистического решения оформления.

Ниже на рисунке 3.1 приведен пример рисунка.

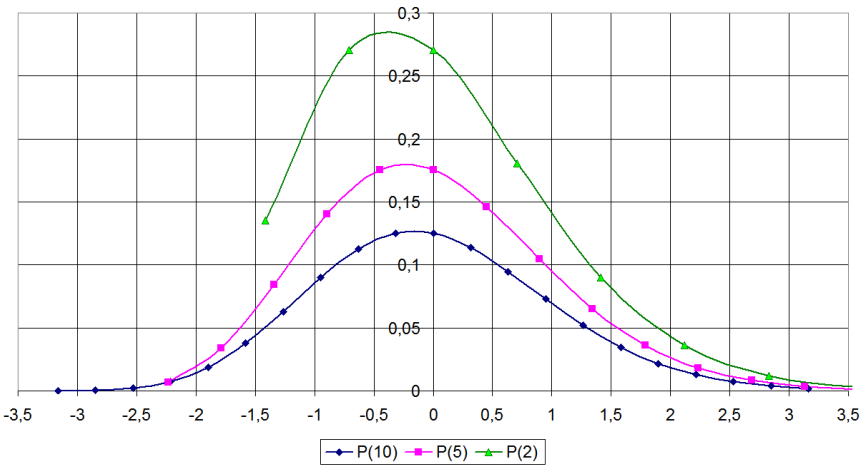

Рисунок 3.1 – Распределения Пуассона при различных значениях исходных параметров в нормализованной форме

Следует отметить, что ***блок-схемы, фрагменты программ, экранные формы, скриншоты, протоколы работы программ, содержимое файлов и т.п. следует также оформлять как рисунок.***

Первая ссылка на рисунок должна ***предшествовать*** самому рисунку, включенному в рукопись, но не далее, чем на 1 страницу.

3.7 Оформление таблиц

Пояснительная записка может содержать разнообразные таблицы, представление которых определено ГОСТ.

Таблицы, за исключением таблиц приложений, следует нумеровать арабскими цифрами сквозной нумерацией.

Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой. Наименование таблицы начинается с прописной буквы и выполняется без точки в конце.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения (буквы).

Если в отчете одна таблица, то она должна быть обозначена "Таблица 1" или "Таблица В.1", если она приведена в приложении В.

Таблицы применяют для лучшей наглядности и удобства сравнения показателей, характеристик. Наименование таблицы, при его наличии, должно отражать ее содержание, быть точным, кратким. Наименование таблицы следует помещать над таблицей слева, без абзацного отступа в одну строку с ее номером через тире.

Таблицу следует располагать в отчете непосредственно после текста, в котором она упоминается впервые, или на следующей странице.

На все таблицы должны быть ссылки в пояснительной записке. При ссылке следует писать слово "таблица" с указанием ее номера.

Таблицу с большим числом строк допускается переносить на другую страницу. При переносе части таблицы на другую страницу слово "Таблица", ее номер и наименование указывают один раз слева над первой частью таблицы, а над другими частями также слева пишут слова "Продолжение таблицы" и указывают номер таблицы. При переносе таблиц на другую с страницу первая часть таблицы должна иметь нижнюю границу в виде сплошной линии, а ее продолжение на следующей странице должно иметь верхнюю границу в виде сплошной линии.

В ячейках таблицы содержимое выравнивается по центру и допускается использование шрифта размером 10 пт., 12 пт. или 14 пт. в зависимости от количества представляемой в таблице информации с единичным межстрочным интервалом.

В приложении приведены примеры представления таблиц, расположенных на одной и нескольких страницах. В последнем случае рекомендуется использование нумерации столбцов в таблице и повторять

указанную нумерацию столбцов при продолжении таблицы на последующих листах;

3.8 Уравнения и формулы в пояснительной записке

Уравнения и формулы в пояснительной записке следует выделять из текста в отдельную строку. Если уравнение не умещается в одну строку, то оно должно быть перенесено после знака равенства или после знаков плюс, минус, умножения, деления или других математических знаков, причем знак в начале следующей строки повторяют.

Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они даны в формуле.

Формулы в записке записываются по центру строки и их следует нумеровать порядковой нумерацией в пределах всей пояснительной записки арабскими цифрами в круглых скобках в крайнем правом положении на строке.

Уравнения и формулы выделяются из текста в отдельную строку. Если уравнение не умещается в одну строку, то допускается перенос уравнения на следующую строку после знака равенства (=) или после знаков плюс(+), минус (–), умножения (×), деления (:) или других математических знаков. На новой строке знак повторяется. При переносе формул на знаке умножения применяют знак (×):

В формулах в качестве символов следует применять обозначения, установленные соответствующими государственными стандартами. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него. Порядковый номер формул выравнивается по правому краю страницы.

Для записи формул удобно использовать табличное представление в виде строки с двумя ячейками. В левую ячейку по центру записывается формула, а в правую ячейку записывается порядковый номер формулы. Границы таблицы при этом невидимые, а для форматирования левой и правой ячейки применяем требуемый вариант форматирования. В приложении показано, как, используя таблицу как средство разметки, можно добиться эффекта прижатого вправо номера формулы, а сама формулы – отцентрирована. *В оформлении таблицы естественно необходимо убрать начертание границ.*

Допускается при оформлении нумерации формул осуществлять выравнивания по правому краю, тем самым нумерация будет выровнена по правой стороне, а между формулой и номером вставлять необходимое количество символов табуляции так, чтобы левый край формулы достиг требуемой позиции по усмотрению студента, но единой для всего документа, например, с позиции двух красных строк.

Если в тексте пояснительной записки используются отдельные математические выражения, то ***не допускается***:

- применять математический знак минус (–) перед отрицательными значениями величин (следует писать слово "минус");
- применять без числовых значений или обозначений переменных и параметров математические знаки, например, (>) (больше), (<) (меньше), (=) (равно), (≥) (больше или равно), (≤) (меньше или равно), (≠) (не равно), а также знаки (№) (номер), (%) (процент).

3.9 Список использованных источников

Сведения об источниках следует располагать в порядке появления ссылок на источники в тексте отчета и нумеровать арабскими цифрами без точки и печатать с абзацного отступа. Например

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Отчет о научно-исследовательской работе. Структура и правила оформления. -М.: Стандартинформ, 2017.-32 с.

При оформлении списка использованных источников приняты правила представления данных для различных видов источников, примеры которых представлены в приложении.

3.10 Приложения

Приложение оформляют как продолжение курсовой работы на последующих её листах.

В тексте на все приложения должны быть даны ссылки. Приложения располагают в порядке ссылок на них в тексте.

Каждое приложение следует размещать с новой страницы с указанием в центре верхней части страницы слова "ПРИЛОЖЕНИЕ".

Приложение должно иметь заголовок, который записывают с прописной буквы, полужирным шрифтом, отдельной строкой по центру без точки в конце.

Приложения нумеруются прописными буквами кириллицы, начиная с А, без использования букв, начертания которых схожи с цифрами или имеют неоднозначное восприятие (Ё, З, Й, Щ, Ч, Ъ, Ы, Ь).

Пример:

ПРИЛОЖЕНИЕ В
Набор программных модулей для моделирования рабочих процессов

На электронном носителе имеется папка Model с проектом Model.alp, в котором содержатся следующие программные модули

4 ПОРЯДОК ЗАЩИТЫ КУРСОВОЙ РАБОТЫ

Аттестация обучающихся по результатам выполнения курсовой работы проводится до начала экзаменационной сессии по расписанию. Форма промежуточной аттестации – дифференцированный зачет (зачет с оценкой), ее содержание – защита работы.

Законченная курсовая работа, пояснительная записка к которой подписана обучающимся, предоставляются руководителю на проверку и подготовку отзыва. Срок сдачи определяется заданием на курсовую работу.

Содержание проверки заключается в определении степени достижения поставленной цели, соответствии предоставленного материала варианту задания курсовой работы и достоверности полученных результатов в соответствии с заданием, а также правильности оформления пояснительной записки к курсовой работе в соответствии с Рекомендациями по оформлению письменных работ обучающихся (СМКО МИРЭА 7.5.1/03.П.69), требованиями ГОСТов. Проверка курсовой работы руководителем завершается написанием (подготовкой) отзыва или *соответствующей визой руководителя* на титульной странице пояснительной записки.

При наличии в курсовой работе недостатков руководитель имеет право допустить ее к защите (указав на них в отзыве или на обороте титульного листа) или предложить обучающемуся устранить их. Обучающийся обязан доработать или переработать курсовую работу в срок, установленный руководителем с учетом сущности замечаний и объема необходимой доработки.

При наличии в курсовой работе существенных недостатков и отсутствии, по мнению руководителя, возможности ее доработки руководитель не допускает курсовую работу к защите и проставляет в экзаменационной ведомости обучающемуся неудовлетворительную оценку.

Курсовая работа оценивается, как правило, преподавателем кафедры, по тематике которого выполняется курсовая работа, не руководившим данной курсовой работой. по результатам её защиты. График защит курсовых работ формируется заведующим кафедрой и доводится до сведения преподавателей кафедры и обучающихся распоряжением по кафедре.

Защита курсовой работы, как правило, состоит в коротком докладе обучающегося (обычно. 5-7 минут) и в ответах на вопросы по существу курсовой работы. Содержание доклада должно отражать материалы, приведенные в ведении, основной части (как теоретической, так и практической), а также в заключении в виде электронной презентации и демонстрации работоспособности разработанной студентом программы.

Вопросы могут относиться к материалам, оформлению и содержанию пояснительной записки к курсовой работе, к теоретическим и практическим вопросам программной реализации задания на курсовую работу, по исходному тексту программы, а также к вводу исходных данных программы, отличных от предлагаемых студентом в контрольном примере.

Вопросы, заданные обучающемуся во время защиты, краткая характеристика его ответов и замечания по существу работы и/или по ответам обучающегося могут быть записаны непосредственно на самой пояснительной записке или на её обороте.

При защите курсовой работы обучающийся должен продемонстрировать уровень сформированности компетенций, предусмотренных в соответствии с рабочей программой дисциплины, ответить на вопросы по теме курсовой работы, а также на замечания руководителя (при наличии). При оценке курсовой работы учитывается качество устного ответа обучающегося, глубина и содержательность проработки используемых алгоритмов, умение обосновать принимаемые в ходе программной реализации технологические, алгоритмические и программные решения, полученные выводы и рекомендации.

Оценка за курсовую работу выставляется в соответствии с показателями и критериями оценивания компетенции и используемыми шкалами оценивания, приведенными далее в тексте методических рекомендаций.

Обучающимся, получившим неудовлетворительную оценку за курсовую работу предоставляется право выбора новой темы или варианта задания на курсовую работу или. по решению руководителя, переработки прежней темы и определяется новый срок для ее выполнения.

Обучающийся, не представивший в установленный срок законченную курсовую работу или не защитивший ее, считается имеющим академическую задолженность.

Электронные версии успешно защищенных курсовых работ в виде файлов в формате .pdf размещаются в электронно-библиотечной системе Университета.

5 КРИТЕРИИ ОЦЕНКИ КУРСОВОЙ РАБОТЫ

5.1 Оценка сформированности компетенций

Курсовая работа предполагает выполнение задания (варианта задания), позволяющего оценить, как степень осведомленности студента в области технологии проектирования, разработки и тестирования программного обеспечения, так и умении и навыки, непосредственно используемые при выполнении этих этапов разработки и оформлении сопутствующей документации.

Для обеспечения профессиональной практической деятельности в области разработки программного обеспечения студент должен продемонстрировать следующие *знания*:

- языки, системы и инструментальные средства программирования в профессиональной деятельности;
- общие принципы построения и использования современных языков программирования высокого уровня;
- основные сведения о базовых структурах данных;
- общие сведения о методах проектирования, документирования, разработки, тестирования и отладки программного обеспечения;
- принципы построения языков и систем программирования;
- возможности библиотек программ и классов для решения различных задач;
- технологии программирования;
- современные средства программного обеспечения для разработки программного продукта;
- методы и принципы разработки программного и иного вида обеспечения с учетом решаемых задач профессиональной сферы деятельности.

Практические *умения* студента предполагает следующее:

- использовать языки, системы и инструментальные средства программирования в профессиональной деятельности;
- строить алгоритм решения задачи, проводить его анализ и реализовывать в современных программных комплексах;
- работать с интегрированной средой разработки программного обеспечения;
- использовать современные средства разработки программного обеспечения на языках высокого уровня, библиотеки программ и классов для решения практических задач;
- проектировать и реализовывать современный пользовательский интерфейс;
- применять современные средства программного обеспечения и вычислительной техники при разработке программного продукта, а также поиска и обработки информации;

– применять основные методы и принципы при разработке программного продукта и иного вида обеспечения специализированных программных комплексов.

При выполнении курсовой работы студент должен продемонстрировать *владение* следующими навыками:

– языками программирования, системами и инструментальными средствами программирования в профессиональной деятельности;
– навыками разработки, документирования, тестирования и отладки программ на языке программирования высокого уровня;
– основными методами разработки алгоритмов и программ;
– методами создания структур данных, используемые для представления типовых информационных объектов;
– основными современными технологиями для осуществления поиска и обработки информации;
– методами и принципами разработки программного и иного вида обеспечения специальных ИАС.

В таблицах 5.1 и 5.2 приведены оценки сформированности отдельных элементов компетенций и комплексная оценка сформированности в части знаний, умений и владений, соответственно.

Таблица 5.1 – Оценка сформированности отдельных элементов компетенций

Обозначения		**Формулировка требований к степени сформированности компетенции**		
Цифр.	**Оценка**	***Знать***	***Уметь***	***Владеть***
1	Неудовлет-ворительно	Отсутствие знаний	Отсутствие умений	Отсутствие навыков
2	Неудовлет-ворительно	Фрагментарные знания	Частично освоенное умение	Фрагментарное применение
3	Удовлетво-рительно	Общие, но не структурированные знания	В целом успешное, но не систематически осуществляемое умение	В целом успешное, но не систематическое применение
4	Хорошо	Сформированные, но содержащие отдельные пробелы знания	В целом успешное, но содержащие отдельные пробелы умение	В целом успешное, но содержащее отдельные пробелы применение навыков
5	Отлично	Сформированные систематические знания	Сформированное умение	Успешное и систематическое применение навыков

Таблица 5.2 – Комплексная оценка сформированности компетенций

Обозначения		Формулировка требований к степени сформированности компетенции
Цифровая	**Оценка**	
1	Неудовлет–ворительно	Не имеет необходимых представлений о проверяемом материале
2	Удовлет–ворительно или неудовлет–ворительно (по усмотрению преподавателя)	Знать на уровне ориентирования, представлений. Субъект учения знает основные признаки или термины изучаемого элемента содержания, их отнесенность к определенной науке, отрасли или объектам, узнает их в текстах, изображениях или схемах и знает, к каким источникам нужно обращаться для более детального его усвоения.
3	Удовлет–ворительно	Знать и уметь на репродуктивном уровне. Субъект учения знает изученный элемент содержания репродуктивно: произвольно воспроизводит свои знания устно, письменно или в демонстрируемых действиях.
4	Хорошо	Знать, уметь, владеть на аналитическом уровне. Зная на репродуктивном уровне, указывать на особенности и взаимосвязи изученных объектов, на их достоинства, ограничения, историю и перспективы развития и особенности для разных объектов усвоения.
5	Отлично	Знать, уметь, владеть на системном уровне. Субъект учения знает изученный элемент содержания системно, произвольно и доказательно воспроизводит свои знания устно, письменно или в демонстрируемых действиях, учитывая и указывая связи и зависимости между этим элементом и другими элементами содержания учебной дисциплины, его значимость в содержании учебной дисциплины.

5.2 Критерии определения оценки за курсовую работу

В процессе оценивания защиты студентом курсовой работы комиссия отмечает степень проработанности отдельных подзадач, сформулированных как критерии оценки курсовой работы. Для подтверждения личного участия в реализации соответствующих подзадач, студент должен ориентироваться в исходном коде программы, уметь самостоятельно находить фрагменты кода, реализующие заявленные действия; уметь отвечать на вопросы по технологии их реализации, пояснять конкретные операторы программы. Критерии оценки курсовой работы представлены в таблице 5.3.

Защита курсовой может проходить – в форме презентации разработанной программы (особенно если техническая возможность демонстрации разработанной программы отсутствует или защита проходит с использованием дистанционных технологий) – в этом случае на слайды выносятся фрагменты кода, соответствующие оцениваемым параметрам, основные вопросы из теории, поясняющие соответствующие технологии,

демонстрационный пример работы программы в виде скриншотов. Объем презентации – до 20 слайдов;
– в форме демонстрации работы программы, когда вопросы комиссии задаются по исходному коду текста программы;
– в смешанном формате – краткий доклад по использованным при разработке программы технологиям, методам, алгоритмам и структурам данных и демонстрации работы программы.

Таблица 5.3 – Критерии оценки курсовой работы

Оценка	**Критерии оценивания**
Оценка – **«удовлетворительно»**	Реализована загрузка данных с клавиатуры.
	Программа запускается и верно выполняет задание согласно варианту
	Студент имеет представление о работе программы.
Оценка **«хорошо»** – то же, что и на оценку «удовлетворительно» с применением следующих элементов:	Реализованы функции записи и чтения информации в/из файл(а).
	Использование динамической памяти (операция new), реализован класс список.
	Функции добавления, удаления и изменения записей.
	Умение объяснить принципы работы разработанной программы.
Оценка **«отлично»** – то же что и на оценку «хорошо», дополнительно: использование классов и объектов для решения поставленной задачи согласно варианту с применением следующих элементов:	Использование ООП
	Реализовано индивидуальное задание
	Шифрование и дешифрование файла (например, на основе системы CRYPTO++ или WinCrypt или OpenSSL)
	Умение объяснить работу любого элемента разработанной программы

6 ВАРИАНТЫ ЗАДАНИЙ

Задание по выполнению курсовой работы: в соответствии с исходными данными, указанными в 2.3, разработать программу (требование к которой указаны в 2.4), реализующую ввод исходных данных с клавиатуры, с занесением информации в файл, обеспечение возможности в дальнейшем корректировки информации внутри файла без изменения остальных данных в файле, а также осуществление шифрования и дешифрования этого файла, используя средства, например, библиотек Microsoft Visual Studio, OpenSSL для Windows или других. В качестве результата работы необходимо вывести на экран данные, указанные в 2.5 с учетом реализации ***варианта задания***, *номер которого* для конкретного студента определяется как ***последние две цифры номера зачетной книжки***.

Таблица 5.4 – Вариант задания по последней цифре в зачетные книжки

Последняя цифра в зачетной книжке	**Формат файла**
0	Текстовый
1	Бинарный
2	Текстовый
3	Бинарный
4	Текстовый
5	Бинарный
6	Текстовый
7	Бинарный
8	Текстовый
9	Бинарный

Список вариантов по двум последним цифрам в зачетной книжке

Вариант 01. Отсортировать группу по убыванию успеваемости любой одной или нескольких сессий (в т.ч. м.б. и всех), вводимых по желанию пользователя.

Вариант 02. Распечатать всех студентов, у которых за все время обучения нет ни одной оценки

а) 3;
б) 3 и 4;
в) 5;
г) 3 и 5;
д) 4 и 5.

Варианты (а – д) выбираются по желанию пользователя. Их можно выбрать как 1, так и несколько или все варианты.

Вариант 03. Распечатать всех студентов, у которых за все время обучения не более 25% оценок

а) 3;
б) 3 и 4;
в) 5;
г) 3 и 5;
д) 4 и 5.

Варианты (а – д) выбираются по желанию пользователя. Их можно выбрать как 1, так и несколько или все варианты.

Вариант 04. Распечатать всех студентов в порядке убывания 5 в одном, нескольких или всех семестрах, которые выбираются по желанию пользователя.

Вариант 05. Отсортировать всех студентов в порядке уменьшения процентного содержания троек за один, несколько или все семестры, которые выбираются по желанию пользователя.

Вариант 6. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом в одной, нескольких или всех сессиях, выбираемых по желанию пользователя.

Вариант 7. Распечатать всех студентов, у которых за все время обучения нет ни одной тройки.

Вариант 8. Распечатать всех студентов, у которых не более 25 процентов троек за все время обучения.

Вариант 9. Распечатать в порядке убывания всех студентов, по количеству пятерок во 2-ом семестре.

Вариант 10. Отсортировать всех студентов в порядке уменьшения процентного содержания "троек" за 1 и 2 семестры.

Вариант 11. Отсортировать группу по уменьшению успеваемости любой сессии.

Вариант 12. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом.

Вариант 13. Отсортировать группу по убыванию успеваемости 2-ой сессии, или же вводимой по желанию пользователя

Вариант 14. Разбить группу на 2 части: мужскую и женскую. Отсортировать каждую часть по алфавиту.

Вариант 15. Разбить группу на 2 части:

- хорошисты и отличники;
- троечники.

Каждую часть отсортировать по номерам зачетных книжек.

Вариант 16. Разбить группу на 2 части:

1) по алфавиту от А до П;
2) по алфавиту от Р до Я.

Каждую часть отсортировать в порядке увеличения среднего бала за все время обучения.

Вариант 17. Разбить группу на 2 части:

1) 50 процентов хороших и отличных оценок за все время обучения;
2) Все остальные студенты

Распечатать в каждой части 2-х наиболее успевающих и наиболее неуспевающих студентов.

Вариант 18. Разбить группу на 3 части:

1) отличников;
2) хорошистов;
3) троечников;

по сессии, вводимой по желанию пользователя за все время обучения.

Вариант 19. Разбить группу на 3 части:

1) отличников;
2) хорошистов;
3) троечников

за все время обучения. Отсортировать каждую часть по алфавиту.

Вариант 20. Разбить группу на 3 части:

1) отличников;
2) отличников и хорошистов;
3) хорошистов и троечников;

за семестр вводимый по желанию пользователя. Распечатать по алфавиту студентов каждой части группы.

Вариант 21. Разбить группу на 2 части:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по номеру зачетных книжек.

Вариант 22. Разбить группу на 2 части:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по алфавиту.

Вариант 23. Разбить группу на 2 части:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по успеваемости за все время обучения.

Вариант 24. Разбить группу на 2 части:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Найти в каждой части наиболее успевающих и наиболее неуспевающих студентов.

Вариант 25. Отсортировать всю группу по увеличению года поступления в ВУЗ.

Вариант 26. Разбить группу на две части:

1) сдавших все спец. предметы только на 4 и 5;
2) сдавших спец. предметы на 3,4,5.

Отсортировать каждую часть по алфавиту.

Вариант 27. Отсортировать группу по успеваемости в каждой сессии или вводимой по требованию пользователя.

Вариант 28. Отсортировать группу по убыванию успеваемости любой одной или нескольких сессий (в том числе, возможно, и всех), вводимых по желанию пользователя. С выбором пола человека

Вариант 29. Распечатать всех студентов, у которых за все время обучения нет ни одной оценки

а) 3;
б) 3 и 4;
в) 5.

Предусмотреть распечатывать лиц определенного пола. Варианты а) – в) выбираются по желанию пользователя.

Вариант 30. Распечатать всех студентов, у которых за все время обучения не более 25% оценок

а) 3;
б) 3 и 4;
в) 5;
г) 3 и 5;
д) 4 и 5.

Варианты а) – д) выбираются по желанию пользователя. Предусмотреть распечатывать лиц определенного пола.

Вариант 31. Распечатать всех студентов в порядке убывания 5 в одном, нескольких или всех семестрах, которые выбираются по желанию пользователя. Предусмотреть распечатывать лиц определенного пола.

Вариант 32. Отсортировать всех студентов в порядке уменьшения процентного содержания троек за один, несколько или все семестры, которые выбираются по желанию пользователя. Предусмотреть осуществлять поиск лиц определенного пола.

Вариант 33. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом в одной, нескольких или всех сессиях, выбираемых по желанию пользователя. Предусмотреть осуществлять поиск среди лиц определенного пола

Вариант 34. Распечатать всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц определенного пола

Вариант 35. Распечатать всех студентов, у которых не более 25 процентов троек за все время обучения с поиском среди лиц определенного пола

Вариант 36. Распечатать в порядке убывания всех студентов, по количеству пятерок во 2-ом семестре с поиском лиц определенного пола

Вариант 37. Отсортировать всех студентов в порядке уменьшения процентного содержания "троек" за 1 и 2 семестры, с поиском среди лиц определенного пола

Вариант 38. Отсортировать группу по уменьшению успеваемости любой сессии, с поиском среди лиц определенного пола.

Вариант 39. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом, с поиском среди лиц определенного пола.

Вариант 40. Отсортировать группу по убыванию успеваемости 2-ой сессии, или же вводимой по желанию пользователя, с поиском среди лиц определенного пола

Вариант 41. Разбить группу на 2 части: мужскую и женскую. Отсортировать каждую часть по алфавиту.

Вариант 42. Разбить группу на 2 части, с поиском среди лиц определенного пола:

- хорошисты и отличники;
- троечники.

Каждую часть отсортировать по номерам зачетных книжек.

Вариант 43. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) по алфавиту от А до П;
2) по алфавиту от Р до Я.

Каждую часть отсортировать в порядке увеличения среднего бала за все время обучения.

Вариант 44. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) 50 процентов хороших и отличных оценок за все время обучения;
2) Все остальные студенты.

Распечатать в каждой части 2-х наиболее успевающих и наиболее неуспевающих студентов.

Вариант 45. Разбить группу на 3 части, с поиском среди лиц определенного пола:

1) отличников;
2) хорошистов;
3) троечников

по сессии, вводимой по желанию пользователя.

Вариант 46. Разбить группу на 3 части, с поиском среди лиц определенного пола:

1) отличников;
2) хорошистов;
3) троечников

за все время обучения. Отсортировать каждую часть по алфавиту.

Вариант 47. Разбить группу на 3 части, с поиском среди лиц определенного пола:

1) отличников;
2) отличников и хорошистов;
3) хорошистов и троечников.

за семестр вводимый по желанию пользователя. Распечатать по алфавиту студентов каждой части группы.

Вариант 48. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по номеру зачетных книжек.

Вариант 49. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по алфавиту.

Вариант 50. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по успеваемости за все время обучения.

Вариант 51. Разбить группу на 2 части, с поиском среди лиц определенного пола:

1) студентов, поступивших в ВУЗ в одном и том же году;

2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Найти в каждой части наиболее успевающих и наиболее неуспевающих студентов.

Вариант 52. Отсортировать всю группу по увеличению года поступления в ВУЗ, с поиском среди лиц определенного пола.

Вариант 53. Разбить группу на две части, с поиском среди лиц определенного пола:

1) сдавших все спец. предметы только на 4 и 5;

2) сдавших спец. предметы на 3,4,5.

Отсортировать каждую часть по алфавиту.

Вариант 54. Отсортировать группу по успеваемости в каждой сессии или вводимой по требованию пользователя, с поиском среди лиц определенного пола.

Вариант 55. Отсортировать группу по убыванию успеваемости любой одной или нескольких сессий (в том числе возможно и всех), вводимых по желанию пользователя, с указанием интервала года рождения.

Вариант 56. Распечатать всех студентов, у которых за все время обучения нет ни одной оценки

а) 3;

б) 3 и 4;

в) 5;

г) 3 и 5;

д) 4 и 5.

Их можно выбрать как 1, так и несколько или все варианты. Варианты а) – д) выбираются по желанию пользователя, с указанием интервала года рождения.

Вариант 57. Распечатать всех студентов, у которых за все время обучения не более 25% оценок

а) 3;
б) 3 и 4;
в) 5;
г) 3 и 5;
д) 4 и 5.

Варианты а) – д) выбираются по желанию пользователя. Их можно выбрать как 1, так и несколько или все варианты, с указанием интервала года рождения.

Вариант 58. Распечатать всех студентов в порядке убывания 5 в одном, нескольких или всех семестрах, которые выбираются по желанию пользователя, с указанием интервала года рождения.

Вариант 59. Отсортировать всех студентов в порядке уменьшения процентного содержания троек за один, несколько или все семестры, которые выбираются по желанию пользователя, с указанием интервала года рождения.

Вариант 60. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом в одной, нескольких или всех сессиях, выбираемых по желанию пользователя, с указанием интервала года рождения.

Вариант 61. Распечатать всех студентов, у которых за все время обучения нет ни одной тройки, с указанием интервала года рождения.

Вариант 62. Распечатать всех студентов, у которых не более 25 процентов троек за все время обучения, с указанием интервала года рождения.

Вариант 63. Распечатать в порядке убывания всех студентов, по количеству пятерок во 2-ом семестре, с указанием интервала года рождения.

Вариант 64. Отсортировать всех студентов в порядке уменьшения процентного содержания "троек" за 1 и 2 семестры, с указанием интервала года рождения.

Вариант 65. Отсортировать группу по уменьшению успеваемости любой сессии, с указанием интервала года рождения.

Вариант 66. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом, с указанием интервала года рождения.

Вариант 67. Отсортировать группу по убыванию успеваемости 2-ой сессии, или же вводимой по желанию пользователя, с указанием интервала года рождения.

Вариант 68. Разбить группу на 2 части: мужскую и женскую. Отсортировать каждую часть по алфавиту, с указанием интервала года рождения.

Вариант 69. Разбить группу на 2 части, с указанием интервала года рождения:

- хорошисты и отличники;
- троечники.

Каждую часть отсортировать по номерам зачетных книжек.

Вариант 70. Разбить группу на 2 части, с указанием интервала года рождения:

1) по алфавиту от А до П;
2) по алфавиту от Р до Я.

Каждую часть отсортировать в порядке увеличения среднего бала за все время обучения.

Вариант 71. Разбить группу на 2 части, с указанием интервала года рождения:

1) 50 процентов хороших и отличных оценок за все время обучения;
2) Все остальные студенты.

Распечатать в каждой части 2-х наиболее успевающих и наиболее неуспевающих студентов, с указанием интервала года рождения.

Вариант 72. Разбить группу на 3 части, с указанием интервала года рождения:

1) отличников;
2) хорошистов;
3) троечников;

по сессии, вводимой по желанию пользователя за все время обучения.

Вариант 73. Разбить группу на 3 части, с указанием интервала года рождения:

1) отличников;
2) хорошистов;
3) троечников$

за все время обучения. Отсортировать каждую часть по алфавиту.

Вариант 74. Разбить группу на 3 части, с указанием интервала года рождения:

1) отличников;
2) отличников и хорошистов;
3) хорошистов и троечников;

за семестр вводимый по желанию пользователя. Распечатать по алфавиту студентов каждой части группы.

Вариант 75. Разбить группу на 2 части, с указанием интервала года рождения:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по номеру зачетных книжек.

Вариант 76. Разбить группу на 2 части, с указанием интервала года рождения:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по алфавиту.

Вариант 77. Разбить группу на 2 части, с указанием интервала года рождения:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Отсортировать каждую часть по успеваемости за все время обучения.

Вариант 78. Разбить группу на 2 части, с указанием интервала года рождения:

1) студентов, поступивших в ВУЗ в одном и том же году;
2) студентов, поступивших в ВУЗ в др. годы, отличные от части 1.

Найти в каждой части наиболее успевающих и наиболее неуспевающих студентов.

Вариант 79. Отсортировать всю группу по увеличению года поступления в ВУЗ, с указанием интервала года рождения.

Вариант 80. Разбить группу на две части, с указанием интервала года рождения:

1) сдавших все спец предметы только на 4 и 5
2) сдавших спец предметы на 3,4,5.

Отсортировать каждую часть по алфавиту.

Вариант 81. Отсортировать группу по успеваемости в каждой сессии или вводимой по требованию пользователя, с указанием интервала года рождения.

Вариант 82. Отсортировать группу по убыванию успеваемости любой одной или нескольких сессий (в том числе возможно и всех), вводимых по желанию пользователя. С указанием интервала года рождения.

Вариант 83. Распечатать всех студентов, у которых за все время обучения нет ни одной оценки

а) 3;
б) 3 и 4;
в) 5.

Предусмотреть распечатывать лиц с указанием интервала года рождения. Варианты а) – в) выбираются по желанию пользователя.

Вариант 84. Распечатать всех студентов, у которых за все время обучения не более 25% оценок

а) 3;
б) 3 и 4;
в) 5;
г) 3 и 5;
д) 4 и 5.

Варианты а) – д) выбираются по желанию пользователя. Предусмотреть распечатывать лиц с указанием интервала года рождения.

Вариант 85. Распечатать всех студентов в порядке убывания 5 в одном, нескольких или всех семестрах, которые выбираются по желанию пользователя. Предусмотреть распечатывать лиц с указанием интервала года рождения.

Вариант 86. Отсортировать всех студентов в порядке уменьшения процентного содержания троек за один, несколько или все семестры, которые выбираются по желанию пользователя. Предусмотреть осуществлять поиск лиц с указанием интервала года рождения.

Вариант 87. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом в одной, нескольких или всех сессиях, выбираемых по желанию пользователя. Предусмотреть осуществлять поиск среди лиц с указанием интервала года рождения.

Вариант 88. Распечатать всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц с указанием интервала года рождения.

Вариант 89. Распечатать всех студентов, у которых не более 25 процентов троек за все время обучения с поиском среди лиц с указанием интервала года рождения.

Вариант 90. Распечатать в порядке убывания всех студентов, по количеству пятерок во 2-ом семестре с поиском лиц с указанием интервала года рождения.

Вариант 91. Отсортировать всех студентов в порядке уменьшения процентного содержания "троек" за 1 и 2 семестры, с поиском среди лиц с указанием интервала года рождения.

Вариант 92. Отсортировать группу по уменьшению успеваемости любой сессии, с поиском среди лиц с указанием интервала года рождения.

Вариант 93. Найти и распечатать все данные о студентах, которые успевают с наибольшим и наименьшим успехом, с поиском среди лиц с указанием интервала года рождения..

Вариант 94. Отсортировать группу по убыванию успеваемости 2-ой сессии, или же вводимой по желанию пользователя, с поиском среди лиц с указанием интервала года рождения.

Вариант 95. Разбить группу на 2 части: мужскую и женскую. Отсортировать каждую часть по алфавиту.

Вариант 96. Разбить группу на 2 части, с поиском среди лиц с указанием интервала года рождения:

- хорошисты и отличники;
- троечники.

Каждую часть отсортировать по номерам зачетных книжек.

Вариант 97. Разбить группу на 2 части, с поиском среди лиц с указанием интервала года рождения:

1) по алфавиту от А до П;
2) по алфавиту от Р до Я.

Каждую часть отсортировать в порядке увеличения среднего бала за все время обучения.

Вариант 98. Разбить группу на 2 части, с поиском среди лиц с указанием интервала года рождения:

1) 50 процентов хороших и отличных оценок за все время обучения;
2) Все остальные студенты.

Распечатать в каждой части 2-х наиболее успевающих и наиболее неуспевающих студентов.

Вариант 99. Разбить группу на 3 части, с поиском среди лиц с указанием интервала года рождения:

1) отличников;
2) хорошистов;
3) троечников;

по сессии, вводимой по желанию пользователя.

Вариант 00. Разбить группу на 3 части, с поиском среди лиц с указанием интервала года рождения:

1) отличников;
2) хорошистов;
3) троечников;

за все время обучения. Отсортировать каждую часть по алфавиту.

7 СВЕДЕНИЯ ИЗ ТЕОРИИ КРИПТОГРАФИИ

7.1 Общие сведения

В предстоящей Вам работе необходимо будет использовать некоторые элементы криптографии, такие как шифрование, дешифрование, сессионный ключ.

В настоящее время используется большое количество программ и библиотек для работы с криптографическими элементами, например, свободно распространяемый продукт Crypto++. Наша цель стоит не в изучении алгоритмов и способов шифрования, а также их программирования, а способность программным образом использовать уже известные и относительно простые способы некоторой защиты информации.

Для примера будем рассматривать возможность работы с системой защиты Windows под названием CRYPTO API.

CryptoAPI— интерфейс программирования приложений, который обеспечивает разработчиков Windows-приложений стандартным набором функций для работы с криптопровайдером. Входит в состав операционных систем Microsoft. Большинство функций CryptoAPI поддерживается, начиная с Windows 2000. CryptoAPI поддерживает работу с асимметричными и симметричными ключами, то есть позволяет шифровать и расшифровывать данные, а также работать с электронными сертификатами. Набор поддерживаемых криптографических алгоритмов зависит от конкретного криптопровайдера.

7.2 Основные понятия и термины

Криптография – наука о защите данных. Алгоритмы криптографии с помощью математических методов комбинируют входной открытый текст и ключ шифрования, в результате чего получаются зашифрованные данные. Применение криптографии обеспечивает относительно надёжную передачу данных и предотвращение их получения несанкционированной стороной. Применяя хороший алгоритм шифрования, можно максимально усложнить взлом защиты и получения открытого текста подбором ключа.

Шифрование – защита информации от несанкционированного просмотра или использования, особенно при передаче по линиям связи.

Дешифрование – процесс, обратный процессу шифрованию.

Аутентификация (authentication) – проверка подлинности, этот процесс надежного определения подлинности поддерживающих связь компьютеров. Аутентификация основана на методах криптографии, и это гарантирует, что нападающий или прослушивающий сеть не сможет получить информацию, необходимую для рассекречивания пользователя или другого объекта. Аутентификация позволяет поддерживающему связь объекту доказать свое тождество другому объекту без пересылки незащищенных данных по сети. Без "сильной" (strong) аутентификации и

поддержания целостности, данных любые данные и компьютер, который их послал, являются подозрительными.

Шифр – обеспечивает возможность передачи сообщения по незащищенным каналам (не обязательно сетевым) с защитой этого сообщения от прочтения посторонними лицами.

Цифровая подпись – это двоичные данные небольшого объема, обычно не более 256 байт. Цифровая подпись есть не что иное, как результат работы хеш-алгоритма над исходными данными, зашифрованный закрытым ключом отправителя. Проще говоря, берем исходные данные, получаем из них хеш и шифруем хеш своим закрытым ключом (с помощью асимметричного алгоритма).

Сертификат – средство, позволяющее гарантированно установить связь между переданным открытым ключом и передавшей его стороной, владеющей соответствующим личным ключом. Сертификат представляет собой набор данных, зашифрованных с помощью цифровой, или электронной, подписи. Информация сертификата подтверждает истинность открытого ключа и владельца соответствующего личного ключа.

В сфере защиты компьютерной информации криптография применяется в основном для: шифрования и дешифрования данных; а также создания и проверки цифровых подписей. Шифрование данных позволяет ограничить доступ к конфиденциальной информации, сделать ее нечитаемой и непонятной для посторонних. Применение цифровых подписей оставляет данные открытыми, но дает возможность верифицировать отправителя и проверять целостность полученных данных.

7.3 Основы работы с CryptoAPI

Для защиты информации специалистами Microsoft был разработан интерфейс CryptoAPI, который является основой средств защиты Microsoft Internet Security Framework. Он позволяет создавать приложения, использующие криптографические методы и обеспечивает базовые функции защиты для безопасных каналов и подписи кода.

Реализация CryptoAPI позволяет интегрировать со своими приложениями усиленные средства шифрования. Интерфейс CryptoAPI обеспечивает API высокого уровня для аутентификации, подписи, шифрования/дешифрации, а также полную инфраструктуру защиты с общим ключом.

Благодаря данной инфраструктуре, можно воспользоваться функциями управления сертификатами, такими как запрос на создание сертификата, его сохранение или верификация.

Программный интерфейс CryptoAPI предоставляет возможности для добавления в приложение функций аутентификации, шифрования, дешифрования и электронной подписи. Задачами CryptoAPI являются:

- Аутентификация сетевых пользователей;
- Шифрование и дешифрование сетевых сообщений;
- Шифрование и дешифрование данных;

– Создание цифровой подписи и ее подтверждение.

Открытость интерфейса: открытая архитектура CryptoAPI позволяет выбирать Cryptographic Service Provider (CSP) по своему усмотрению. CryptoAPI доступен в ОС Windows, Macintosh и UNIX. Кроме того, в CryptoAPI поддерживаются стандартные форматы сертификатов X.509 версии 3, ASN.1 и DER.

7.3.1 Криптопровайдеры

Любой сеанс работы с CryptoAPI начинается с инициализации (получения контекста). Инициализация выполняется при помощи функции **CryptAcquireContext**. В качестве параметров эта функция принимает имя контейнера ключей, имя криптопровайдера, тип провайдера и флаги, определяющие тип и действия с контейнером ключей и режим работы криптопровайдера:

Криптопровайдер – это независимый модуль, содержащий библиотеку криптографических функций со стандартизованным интерфейсом. Криптопровайдер отвечает за реализацию функций интерфейса, а также играет роль хранилища для ключей всех типов. Подобная архитектура позволяет переходить от одного провайдера к другому с минимальными изменениями исходного кода, так как интерфейс (т. е. сами функции) не меняется.

Список криптопровайдеров Microsoft представлен в таблице 7.1

Таблица 7.1 – Список криптопровайдеров

Провайдер	Описание
Microsoft Base Cryptographic Provider	Включает большой набор криптографических функций; может экспортироваться за пределы США
Microsoft Strong Cryptographic Provider	Расширяет Microsoft *Base Cryptographic* Provider ; доступен, начиная с версии *Windows 2000*
Microsoft Enhanced Cryptographic Provider	Представляет собой Microsoft *Base Cryptographic* Provider с более *длинными ключами* и дополнительными криптоалгоритмами.
Microsoft AES Cryptographic Provider	Microsoft *Enhanced Cryptographic* Provider с поддержкой *AES*
Microsoft DSS Cryptographic Provider	Предоставляет функции *хеширования*, генерации и *проверки ЭЦП* с использованием алгоритмов *Secure Hash Algorithm* (*SHA*) и *Digital Signature Standard* (*DSS*)
Microsoft Base DSS and Diffie-Hellman Cryptographic Provider	Помимо функциональности *DSS Cryptographic* Provider, поддерживает схему обмена ключами Диффи-Хеллмана
Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider	Поддерживает схему обмена ключами Диффи-Хеллмана, функции *хеширования*, генерации и *проверки ЭЦП*, соответствующие стандарту *FIPS* 186-2, *симметричное шифрование* на базе *RC4*
Microsoft DSS and Diffie-Hellman/Schannel Cryptographic Provide r	Поддерживает хеширование и *ЭЦП* на базе *DSS* ; схему обмена ключами Диффи-Хеллмана, генерацию и экспорт этих ключей; получение ключей для протоколов SSL3 и TLS1
Microsoft RSA/Schannel Cryptographic Provider	Поддерживает хеширование и *ЭЦП*, а также получение ключей для протоколов SSL2, PCT1, SSL3 и TLS1
Microsoft RSA Signature Cryptographic Provider	Предоставляет функциональность для реализации *ЭЦП*

Со списком криптопровайдеров, представленных в Вашей системе, можно ознакомиться через редактор реестра (рисунок 7.1).

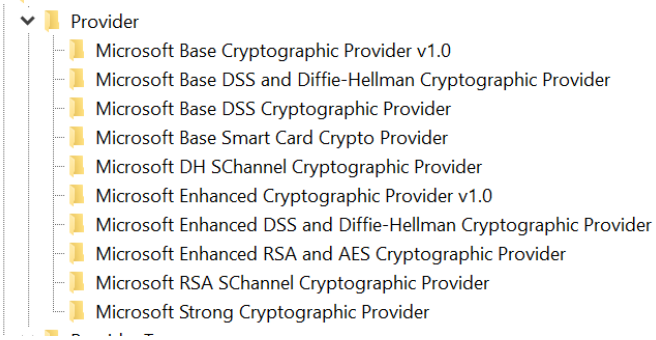

Рисунок 7.1 – Пример списка установленных криптопровайдоров

Там же можно увидеть тип провайдера и его возможности, рисунок 7.2. и расширенная информация о соответствующем типе (рисунки 7.3 и 7.4).

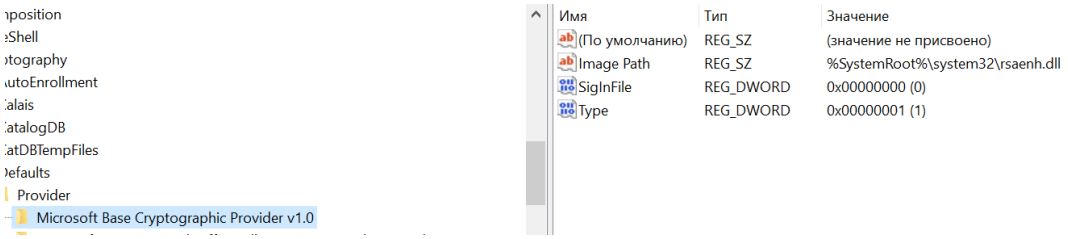

Рисунок 7.2 – Строка реестра с информацией о криптопровайдоре

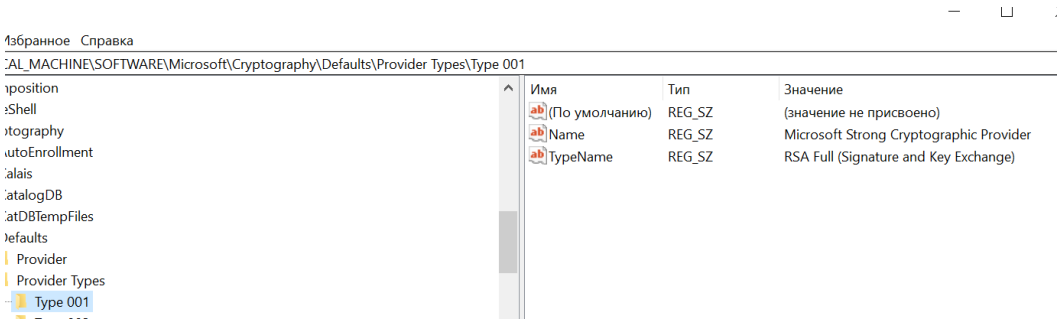

Рисунок 7.3 – Пример информации о криптопровайдоре, тип RSA

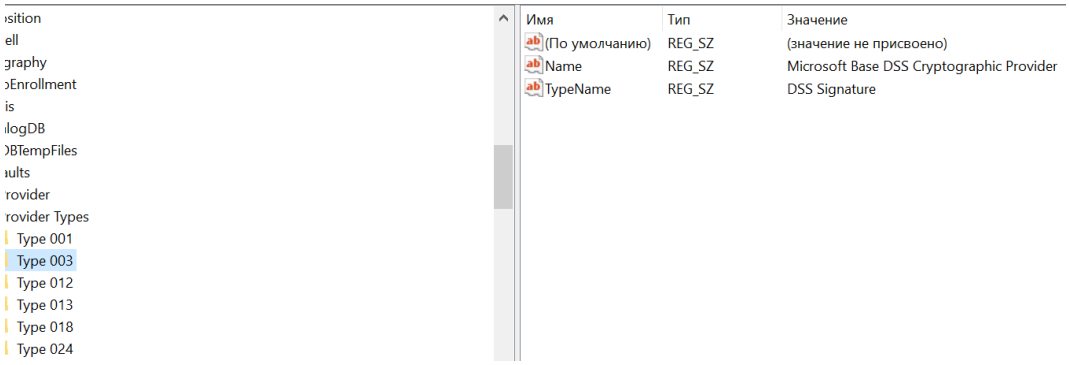

Рисунок 7.4 – Пример информации о криптопровайдоре, тип DSS

Прежде чем использовать какие-либо функции Crypto API, необходимо запустить криптопровайдер. Делается это с помощью функции CryptAcquireContext:

```
BOOL CryptAcquireContext (
HCRYPTPROV* hCryptProvider,          //дескриптор провайдера
LPCTSTR pszContainer,                 // имя контейнера ключей
LPCTSTR pszProvider,                  // имя провайдера
DWORD dwProvType,                     // тип провайдера
DWORD dwFlags                         // флаги
)
```

где:

hCryptProvider - указатель на дескриптор CSP, является выходным параметром. По завершении процесса работы с криптопровайдером(CSP), необходимо вызвать функцию CryptReleaseContext для освобождения(очистки) дескриптора и контейнера ключей.

pszContainer - имя ключевого контейнера. Это строка с нулевым завершением, которая идентифицирует контейнер ключей для CSP. Это имя не зависит от метода, используемого для хранения ключей (может быть

массивом символов, строкой). Некоторые CSP хранят свои контейнеры ключей внутри (в аппаратном обеспечении), некоторые используют системный реестр, а другие-файловую систему. В большинстве случаев, когда dwFlags имеет значение CRYPT_VERIFYCONTEXT, pszContainer должен иметь значение NULL

pszProvider - имя провайдера. Это строка с нулевым завершением, которая идентифицирует имя используемого провайдера.

dwProvType – тип провайдера:

- PROV_RSA_FULL;
- PROV_RSA_AES;
- PROV_RSA_SIG;
- PROV_RSA_SCHANNEL;
- PROV_DSS;
- PROV_DSS_DH;
- PROV_DH_SCHANNEL;
- PROV_FORTEZZA;
- PROV_MS_EXCHANGE;
- PROV_SSL.

Подробное описание типа криптопровайдера содержится в документации по адресу https://docs.microsoft.com/ru-ru/windows/win32/seccrypto/cryptographic-provider-types

dwFlags – флаги работы:

- CRYPT_VERIFYCONTEXT;
- CRYPT_NEWKEYSET;
- CRYPT_MACHINE_KEYSET;
- CRYPT_DELETEKEYSET;
- CRYPT_SILENT;
- CRYPT_DEFAULT_CONTAINER_OPTIONAL.

Более подробное описание типа криптопровайдера можно посмотреть в документации https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptacquirecontextw

Кроме инициализации криптопровайдера данную функцию можно использовать для создания и удаления контейнеров ключей. Для этого параметру dwFlags присваивается значение, соответственно, CRYPT_NEWKEYSET и CRYPT_DELETEKEYSET. Функция CryptAcquireContext работает в два этапа: сначала она ищет криптопровайдер по имени и типу, указанному в аргументах, а затем контейнер ключей с заданным именем. По окончании работы с криптопровайдером необходимо вызвать функцию CryptReleaseContext.

Криптопровайдеры отличаются друг от друга:

- составом функций (например, некоторые криптопровайдеры не выполняют шифрование данных, ограничиваясь созданием и проверкой цифровых подписей);

– требованиями к оборудованию (специализированные криптопровайдеры могут требовать устройства для работы со смарт-картами для выполнения аутентификации пользователя);
– алгоритмами, осуществляющими базовые действия (создание ключей, хеширование и пр.).

По составу функций и обеспечивающих их алгоритмов криптопровайдеры подразделяются на типы. Например, любой CSP типа PROV_RSA_FULL поддерживает как шифрование, так и цифровые подписи, использует для обмена ключами и создания подписей алгоритм RSA, для шифрования – алгоритмы RC2 и RC4, а для хеширования – MD5 и SHA. В зависимости от версии операционной системы состав установленных криптопровайдеров может существенно изменяться. Однако на любом компьютере с Windows можно найти Microsoft Base Cryptographic Provider, относящийся к типу PROV_RSA_FULL. Именно с этим провайдером по умолчанию будут взаимодействовать все программы. Приложение обращается к криптопровайдеру не напрямую, а через CryptoAPI. Первым делом, хотя бы из любопытства, выясним, какие же криптопровайдеры установлены в системе.

Для определения имени криптопровайдера, его типа, реализации, версии, поддерживаемых алгоритмов в CryptoAPI используются следующие функции:

CryptEnumProviders (i, резерв, флаги, тип, имя, длина_имени) – возвращает имя и тип i-го по порядку криптопровайдера в системе (нумерация начинается с нуля);

CryptGetProvParam (провайдер, параметр, данные, размер_данных, флаги) – возвращает значение указанного параметра провайдера, например, версии (второй параметр при вызове функции – PP_VERSION), типа реализации (программный, аппаратный, смешанный – PP_IMPTYPE), поддерживаемых алгоритмов (PP_ENUMALGS). Список поддерживаемых алгоритмов при помощи этой функции может быть получен следующим образом: при одном вызове функции возвращается информация об одном алгоритме; при первом вызове функции следует передать значение флага CRYPT_FIRST, а при последующих флаг должен быть равен 0;

CryptReleaseContext (провайдер, флаги) – освобождает дескриптор криптопровайдера.

Каждая из этих функций, как и большинство других функций CryptoAPI, возвращает логическое значение, равное *true*, в случае успешного завершения, и *false* – если возникли ошибки. Код ошибки может быть получен при помощи функции GetLastError.

7.3.2 Шифрование

Базовая функция шифрования данных имеет следующее объявление:

```
BOOL CryptEncrypt(HCRYPTKEY hKey,
                  HCRYPTHAS hHash,
                  BOOL Final,
```

DWORD dwFlags,
BYTE* pbData,
DWORD* pdwDataLen,
DWORD dwBufLen);

Первым параметром данной функции передается хендл сессионного ключа, применяемого для шифрования. Второй параметр достаточно редко используется и предназначен для получения хеша данных одновременно с их шифрованием. Такая возможность достаточно полезна при формировании одновременно как шифрованных данных, так и цифровой подписи этих же данных.

Эта функция может обрабатывать данные блоками. То есть нет необходимости сразу загружать в память целиком весь массив данных, а лишь потом передавать ссылку на него криптографической функции. Достаточно передавать массив данных поблочно, специальным образом отметив лишь последний блок данных (это обычно нужно, чтобы криптопровайдер провел некоторые действия после использования сессионного ключа). Для указания того, что это последний блок данных, в функции **CryptEncrypt** используется третий параметр *Final*. Четвертый параметр служит указателем на массив входных/выходных данных. Здесь нужно сразу отметить некоторую общую схему работы с данными в Crypto API. Если возвращаемые данные могут быть любого размера (а это возможно, ведь, скажем, в алгоритме может происходить простая замена, когда одна буква кодируется четырьмя цифрами), то работа с функцией состоит из двух этапов. На первом этапе в функцию передается общий размер входных данных и NULL в качестве ссылки на сам массив выходных данных. Функция возвращает длину выходного массива данных, пользователь инициализирует память необходимого размера и лишь затем заново передает функции ссылку на этот массив. Такая же схема используется и в работе с функцией **CryptEncrypt**. Параметр *pdwDataLen* служит для возврата размера данных, возвращаемых функцией. Параметр *dwBufLen* служит для указания длины входного буфера данных. Параметр *dwFlags* обычно не используется и устанавливается в 0.

7.3.3 Расшифрование

Базовая функция расшифрования имеет следующее описание:

BOOL CryptDecrypt(HCRYPTKEY hKey,
HCRYPTHASH hHash,
BOOL Final,
DWORD dwFlags,
BYTE* pbData,
DWORD* pdwDataLen);

Первым параметром данной функции передается инициализированный контекст сессионного ключа, применяемого для расшифровывания данных. Второй параметр, как и в предыдущем примере, связан, по большей части, с функцией получения и проверки цифровой подписи. Обычно он не

используется и устанавливается в 0. Параметр *dwFlags* чаще всего не используется и также устанавливается в 0. Параметры *pbData* и *pdwDataLen* используются точно так же, как и у CryptEncrypt и представляют собой ссылку на входной/выходной массив данных и длину этого массива данных.

7.3.4 Криптографические ключи и выполняемые функции

В CryptoAPI существуют ключи двух типов:

- сессионные ключи (session keys);
- пары открытый/закрытый ключ (public/private key pairs).

Сессионные ключи - это симметричные ключи, так как один и тот же ключ применяется и для шифрования, и для расшифровки. Сессионные ключи меняются. Алгоритмы, использующие сессионные ключи (так называемые симметричные алгоритмы), – RC2, RC4, DES. Microsoft RSA Base Provider работает с 40-разрядными сессионными ключами.

Пары открытый/закрытый ключ используются в так называемых асимметричных алгоритмах шифрования. Если шифрование выполнялось одним ключом из пары, то дешифрование производится другим. Открытые (public) ключи могут передаваться другим лицам для проверки цифровых подписей и шифрования пересылаемых данных. Длина открытого ключа в Microsoft RSA Base Provider составляет 512 разрядов. Закрытые (private) ключи не могут быть экспортированы; они используются для создания цифровых подписей и дешифровки данных. Закрытый ключ должен быть известен только его владельцу. При шифровании с открытым ключом жизненно важна абсолютно достоверная ассоциация открытого ключа и передавшей его стороны, поскольку в обратном случае возможна подмена открытого ключа и осуществление несанкционированного доступа к передаваемым зашифрованным данным. Необходим механизм, гарантирующий достоверность корреспондента, например, применение сертификата, созданного авторизованным генератором сертификатов. Обычно сертификаты содержат дополнительную информацию, позволяющую идентифицировать владельца личного ключа, соответствующего данному открытому ключу. Сертификат должен быть подписан авторизованным генератором сертификатов. Выполняемые функции:

Функции кодирования сертификатов:

Эти функции управляют сертификатами и сопутствующими данными через сеть OSI (соединение открытых систем, семиуровневая модель) как описано в CCIT X.200. Методы OSI, описывающие абстрактные объекты, используют абстрактную синтаксическую нотацию один (ASN.1), как описано в CCITT X.209.

Функции базы сертификатов:

Используются для хранения сертификатов и управления ими. Пользователь со временем может собрать весьма много сертификатов. Обычно это сертификаты, описывающие самого пользователя и сертификаты сущностей с которыми он взаимодействует. Обычно для каждой сущности

есть несколько сертификатов - связок, используемых для проверки отслеживания существующих сертификатов у авторитета сертификатов (обычно это сайт с сертификатами, отвечающий своим авторитетом за их верность).

Базовые криптографические функции:

Используются для наиболее полного использования криптографических возможностей в приложении. Это функции, взаимодействующие с провайдером.

Функции сообщений низкого уровня:

Используются для быстрого создания сообщений отвечающих требованиям PKCS#7(RFC2315, RFC2316). Эти функции предназначены для шифрования данных при передаче и дешифровке их при приеме, а также для дешифровки и проверке подписей получаемых сообщений.

Упрощенные функции сообщений:

Они находятся на верхнем уровне функций сообщений и в принципе представляют низкоуровневые функции сообщений и сертификатов в одном. Они уменьшают число вызовов функций для приложения.

Хранение ключей

Криптопровайдер отвечает за хранение и разрушение ключей. Программист не имеет доступа непосредственно к двоичным данным ключа, за исключением операций экспорта открытых ключей. Вся работа с ключами производится через дескрипторы (handle). В CryptoAPI ключи для шифрования / дешифрования и создания / проверки подписей разделены. Называются они соответственно «пара для обмена ключами» и «пара для подписи». База данных ключей состоит из контейнеров, в каждом из которых хранятся ключи, принадлежащие определенному пользователю. Контейнер ключей имеет уникальное имя и содержит пару для обмена и пару для подписи. Все ключи хранятся в защищенном виде. По умолчанию для каждого пользователя создается контейнер с именем этого пользователя.

Генерация ключей и обмен ключами

Для генерации ключей в CryptoAPI предусмотрены две функции – CryptGenKey и CryptDeriveKey. Первая из них генерирует ключи случайным образом, а вторая – на основе пользовательских данных. При этом гарантируется, что для одних и тех же входных данных CryptDeriveKey всегда выдает один и тот же результат. Это способ генерации ключей может быть полезен для создания симметричного ключа шифрования на базе пароля.

BOOL WINAPI CryptGenKey (HCRYPTPROV hProv, ALG_ID Algid, DWORD dwFlags, HCRYPTKEY* phKey);

Первый и четвертый параметры говорят сами за себя. Вторым параметром передается идентификатор алгоритма шифрования, для которого генерируется ключ (например, CALG_3DES). При генерации ключевых пар RSA для шифрования и подписи используются специальные значения AT_KEYEXCHANGE и AT_SIGNATURE. Третий параметр задает различные опции ключа, которые зависят от алгоритма и провайдера. Например,

старшие 16 битов этого параметра могут задавать размер ключа для алгоритма RSA. Подробное описание всех флагов можно найти в MSDN.

Набор именованных констант – идентификаторов криптографических алгоритмов, представлен в таблице 7.2.

Таблица 7.2 – Список идентификаторов криптографических алгоритмов

Идентификатор алгоритма		**Класс алгоритма**	**Тип алгоритма**	**Комментарий**
Имя	**ALG_ID**			
CALG_MD2	32769	Хеширование	любой	MD2, 128 бит
CALG_MD4	32770	Хеширование	любой	MD4, 128 бит
CALG_MD5	32771	Хеширование	любой	MD5, бит
CALG_SHA, CALG_SHA1	32772	Хеширование	любой	SHA-1, бит
CALG_DES	26113	Симметричное шифрование	Блочный - 64 бита	DES с ключом 56 бит
CALG_3DES_112	26121	Симметричное шифрование	Блочный - 64 бита	Тройной DES с ключом 112 бит
CALG_3DES	26115	Симметричное шифрование	Блочный - 64 бита	Тройной DES с ключом 168 бит
CALG_RC2	26114	Симметричное шифрование	Блочный - 64 бита	RC2 с переменной длиной ключа
CALG_RC4	26625	Симметричное шифрование	Поточный	RC4 с переменной длиной ключа

В зависимости от провайдера могут отличаться классом алгоритма, типом, размером, например, для CALG_RC4 представлены в таблице 7.3.

Таблица 7.3 – Детализация информации для алгоритма CALG_RC4

Криптопровайдер	**Класс алгоритма**	**Тип**	**Размер (по умолчанию/ минимальный/ максимальный)**
Microsoft Base Cryptographic Provider v1.0	шифрование	блочный	40/40/56
Microsoft Base DSS and Diffie-Hellman Cryptographic Provider	шифрование	поточный	40/40/56
Microsoft Base Smart Card Crypto Provider	шифрование	поточный	128/40/128
Microsoft DH Schannel Cryptographic Provider	шифрование	поточный	40/40/128

Обмен ключами в CryptoAPI реализуется с помощью функций CryptExportKey и CryptImportKey

После окончания работы с ключом, его нужно уничтожить вызовом CryptDestroyKey. При этом закрытый ключ сохраняется в контейнере (если,

конечно, не использовался режим CRYPT_VERIFYCONTEXT), а сессионные ключи уничтожаются совсем.

7.3.5 Пример программы с использованием CryptoAPI

Пример: задана строка символов, осуществить ее шифрование, ее запись в файл, считывание из файла и ее последующую расшифровку. использовать для шифрования сессионный ключ.

Основные этапы использования CryptoAPI:

1. Подключение к провайдеру;
2. Генерация ключа и его сохранение в памяти (аналогично – в файле);
3. Шифрование;
4. Открытие файла;
5. Запись шифрованной строки в файл;
6. Закрытие файла;
7. Открытие файла;
8. Считывание записи из файла;
9. Расшифровка с помощью сохраненного ключа;
10. Закрытие файла.

Программа реализована на языке программирования C++ в среде Visual Studio 17:

```
#include "pch.h"
#include <iostream>
#include <windows.h>
#include <wincrypt.h>
#include <stdio.h>
using namespace std;

DWORD dwIndex = 0;
DWORD dwType;
DWORD cbName;
LPTSTR pszName,x;
void main()
{
	HCRYPTPROV hProv;
	HCRYPTKEY hSessionKey;

	// Получение контекста криптопровайдера
	if (!CryptAcquireContext(&hProv, NULL, NULL,
		PROV_RSA_FULL, CRYPT_VERIFYCONTEXT))
	{
		std::cout << "CryptAcquireContext" << std::endl;
		return;
	}
	std::cout << "Cryptographic provider initialized" << std::endl;

	// Генерация сессионного ключа
```

```
	if (!CryptGenKey(hProv, CALG_RC4, CRYPT_EXPORTABLE, &hSessionKey))
	{
		std::cout << "CryptGenKey" << std::endl;
		return;
	}
	std::cout << "Session key generated" << std::endl;

	// Данные для шифрования
	char string[] = "TestTest";
	char string1[100], string2[100];
	//memcpy(string2, '\0', 100);
	strcpy_s(string1, string);
	DWORD count = strlen(string);

	// Шифрование данных
	if (!CryptEncrypt(hSessionKey, 0, true, 0, (BYTE*)string, &count, strlen(string)))
	{
		std::cout << "CryptEncrypt" << std::endl;
		return;
	}
	std::cout << "Encryption completed" << std::endl;

	// Тестовый вывод на экран
	std::cout << "Encrypted string: " << string << "::::" << strlen(string) << std::endl;

	//-------------------------------------------------------------------------
	// работа с файлом
	//-------------------------------------------------------------------------
	FILE *out = nullptr, *in = nullptr;
	char c;
	fopen_s(&out, "AAA.ddd", "w");
	fwrite(string, strlen(string), 1, out);
	fclose(out);
	cin >> c;
	fopen_s(&in, "AAA.ddd", "r");
	fread(string1, strlen(string), 1, in);
	fclose(in);

	DWORD count1 = strlen(string1);
	std::cout << "Encrypted string: " << string1 << "::::" << strlen(string1) << std::endl;

	if (!CryptDecrypt(hSessionKey, 0, true, 0, (BYTE*)string1, &count1))
	{
		std::cout << "CryptDecrypt" << std::endl;
		return;
	}
	std::cout << "Deccrypted string: " << string1 << "::::" << strlen(string1) << std::endl;

}
```

Контрольный пример запуска данной программы представлен на рисунке 7.5.

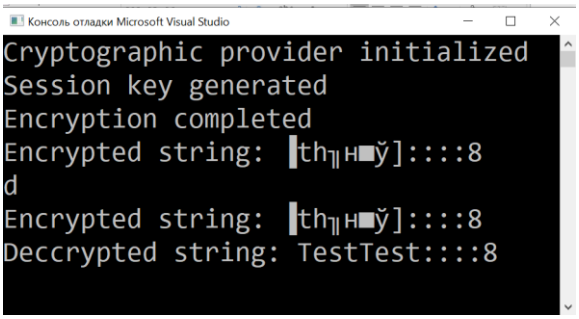

Рисунок 7.5 – Экран консоли запуска программы с использованием CryptoAPI

7.4 Основы использования OpenSSL

7.4.1 Установка OpenSSL

В отличие от Linux, где OpenSSL зачастую предустановлен (в дистрибутивах типа Ubuntu, Debian или Mint), в Windows OpenSSL отсутствует. Скачать OpenSSL для Windows можно по ссылке (https://slproweb.com/products/Win32OpenSSL.html).

File	Type	Description
Win64 OpenSSL v1.1.1k Light EXE | MSI	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1k (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.1k EXE | MSI	63MB Installer	Installs Win64 OpenSSL v1.1.1k (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1k Light EXE | MSI	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1k (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1k EXE | MSI	54MB Installer	Installs Win32 OpenSSL v1.1.1k (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Рисунок 7.6 – Выбор файла

Далее запускается инсталлятор (файл .exe), в установщике выбираем «I accept the agreement» (рисунок 7.7).

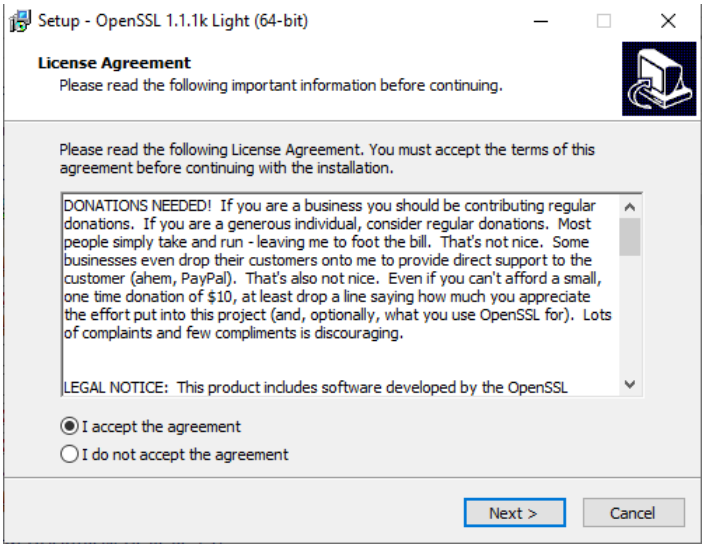

Рисунок 7.7 – Установка: шаг 1

После чего выбираете путь установки, для удобства выберем тот путь, где лежит cpp файл с кодом курсовой работы (рисунок 7.8).

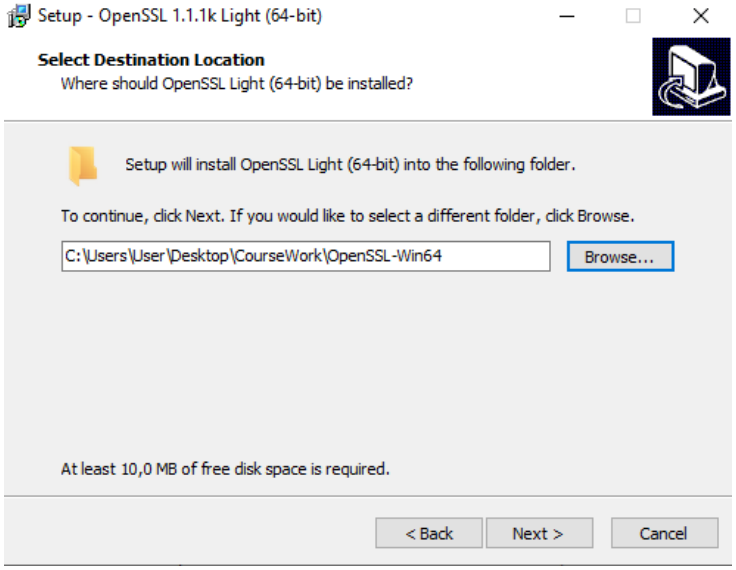

Рисунок 7.8 – Установка: шаг 2

После выбираем The OpenSSL binaries (рисунок 7.9).

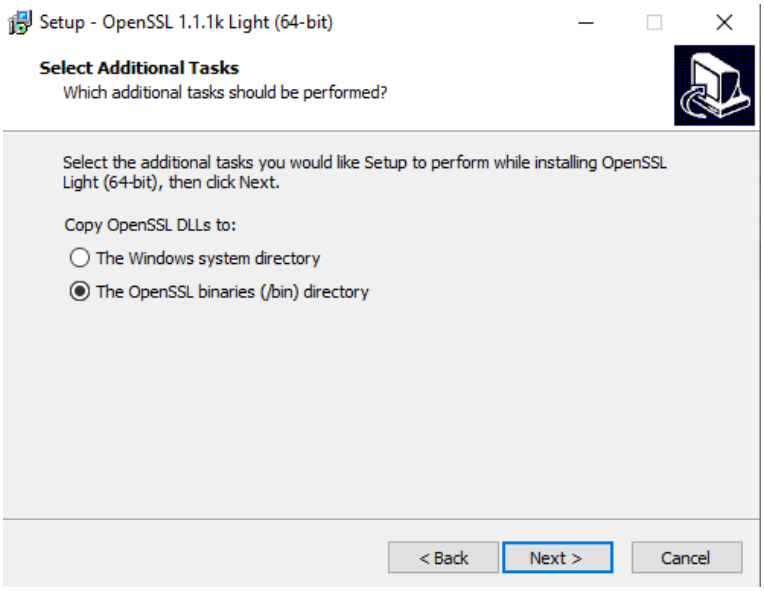

Рисунок 7.9 – Установка: шаг 3

Далее нажимаем Install и установка завершена (рисунок 7.10).

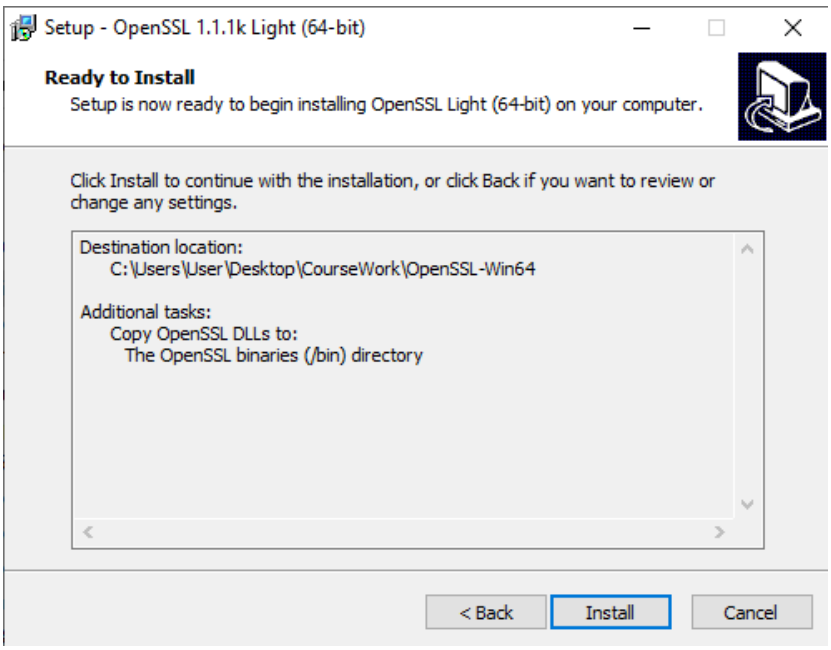

Рисунок 7.10 – Установка: шаг 4

После установки не забудьте проверить, что все галочки убраны, и после этого жмите Finish (рисунок 7.11).

Рисунок 7.11 – Установка: шаг 5

После этого, в папке с кодом появляется папка «OpenSSL-Win64» (рисунок 7.12). Переименовываем данную папку в «openssl» для удобства в дальнейшем.

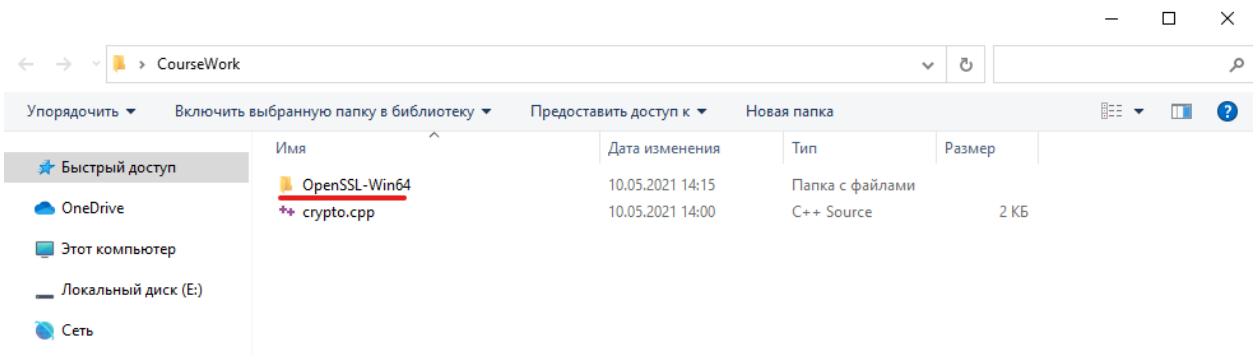

Рисунок 7.12 – Расположение OpenSSL-Win64

На этом этап установки завершен.

7.4.2 Последовательность действий

В криптографии есть два типа шифров: симметричные и ассиметричные. Их главное отличие в том, что ассиметричный алгоритм шифрования имеет два ключа (обычно это открытый ключ, которым шифруют данные и закрытый ключ, которым расшифровывают данные) и симметричный алгоритм, который для шифрования и дешифрования использует один и тот же пароль.

Довольно популярные симметричные алгоритмы это: 3DES и AES-256, в примере будем использовать AES-256-CBC (Cipher Block Chaining).

В качестве ассиметричного алгоритма возьмем RSA и для него сгенеририм 4096 битный ключ.

Необходимо отметить, что обычно ассиметричными алгоритмами не шифруют большой объем данных, так как они плохо для этого подходят и достаточно медленно шифруют большой объем данных, нежели симметричные.

Более подробно можно узнать про шифрование здесь:

1. https://habr.com/ru/post/449552/
2. https://otus.ru/nest/post/726/
3. https://academy.binance.com/ru/articles/symmetric-vs-asymmetric-encryption

Логика программы заключается в том, что у нас есть некая база данных, в которой хранятся записи о студентах, и неважно как они там записаны, и как хранятся. Важно то, что эта база есть и ее надо зашифровывать, чтобы обеспечить безопасность конфиденциальных данных и расшифровывать тогда, когда это необходимо.

Рассмотрим следующую последовательность пользователя: пользователь запускает программу для работы с базой данных студентов, и сразу же после запуска программы она расшифровывает базу данных. После того как пользователь поработал с базой (например, добавил или удалил информацию в ней, а в меню программы нажал «выйти»). В этот момент в файл сохраняется база данных, после чего база данных шифруется.

Так для чего же нам два разных типа алгоритмов шифрования? RSA довольно надежный алгоритм, но из-за того, что он не предназначен для

шифрования большого объёма данных (хоть так можно сделать, но НЕ рекомендуется), мы используем следующие шаги ***для шифрования***:

1. Генерируем рандомный пароль для шифра AES;
2. Шифруем файл с помощью AES;
3. Сохраняем наш пароль, которым мы шифровали базу в файл;
4. Шифруем файл с паролем с помощью публичного ключа RSA.

Для дешифрования:

1. Расшифровываем с помощью приватного ключа RSA файл с паролем;
2. Считываем пароль из файла;
3. Расшифровываем файл с помощью AES используя полученный пароль.

7.4.3 Команды OpenSSL

Для начала, необходимо будет заранее сгенерировать публичный и приватный RSA ключ. Для этого открываем в курсовой работе папку openssl, которую переименовывали, в ней открываем папку bin и запускаем openssl.exe.

В открывшийся терминал вводим две команды (рисунок 7.13).

```
genrsa -out rsa.private 4096
rsa -in rsa.private -pubout -out rsa.public
```

Рисунок 7.13 – команды генерации ключа RSA

Первая команда на рисунке 7.13 – генерирует приватный ключ с именем "rsa.private", ключ будет 4096 бит.

Второй командой на рисунке 7.13 генерируется публичный ключ с именем "rsa.public", и его на его основе создается публичный ключ.

После этого необходимо перенести публичный и приватный ключ из этого каталога в каталог, где у пользователя находится код приложения.

Важно отметить то, что если потеряете публичный ключ, то можно еще раз его сгенерировать, зная приватный, но если потеряете приватный, то больше его никак не сможете получить, и данные так и останутся зашифрованными.

7.4.4 Используемые команды

Для шифрования с помощью **AES** используем следующую команду:

```
aes-256-cbc -salt -a -e -in text.txt -out
text.enc
```

Для расшифровки алгоритмом **AES**:

```
aes-256-cbc -salt -a -d -in text.enc -out
text.txt
```

где параметры «-in» получает входной фай файл,

«-out» формирует выходной;
«-e» означает encrypt, то есть зашифровать файл,
«-d» означает decrypt, то есть расшифровать файл.

Для шифрования алгоритмом **RSA** используем следующую команду:

```
rsautl -encrypt -pubin -inkey rsa.public -in text.txt -out text.enc
```

Для расшифровки алгоритмом **RSA**:

```
rsautl -decrypt -inkey rsa.private -in text.enc -out
text.txt
```

Отметим, что для шифрования используется публичный ключ, а для расшифровки приватный ключ, то есть приватный ключ – главный ключ в RSA алгоритме и его потеря – означает потеря зашифрованной информации. Также для удобства у зашифрованных файлов меняем расширение на .enc (encrypted), для того, чтобы визуально было видно, что файл зашифрован.

7.4.5 Код программы

Теперь перейдем непосредственно к коду программы. Необходимо реализовать две функции (Crypt и Decrypt),

Пусть имя файла базы данных называется file.txt, также имеется публичный и приватный ключ. Должна быть папка openssl, файлы с кодом, файл с базой и ключи (рисунок 4.1)

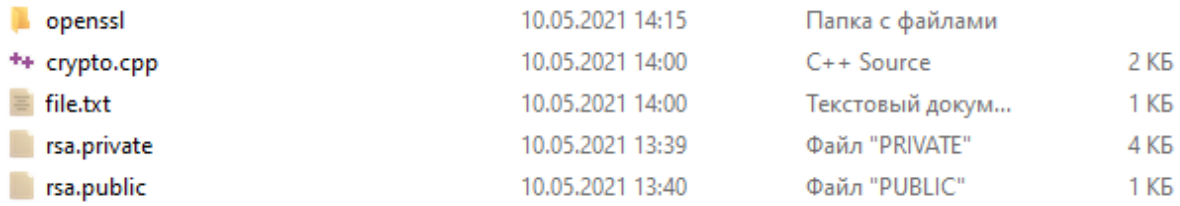

Рисунок 7.14 – Файлы и каталоги

7.4.6 Реализация функции *Crypt ( )*

Реализация функции ***Crypt*** состоит из следующих шагов:

1. Генерация случайного(рандомного) пароля

```
srand(time(NULL));
char* pass = new char[64];
for (int i = 0; i < 64; ++i) {
    switch (rand() % 3) {
    case 0:
        pass[i] = rand() % 10 + '0';
        break;
    case 1:
        pass[i] = rand() % 26 + 'A';
        break;
```

```
11      case 2:
12          pass[i] = rand() % 26 + 'a';
13      }
14  }
```

2. Шифрование алгоритмом AES файла базы данных

```
1  string command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -salt -in file.txt -out file.txt.enc -pass
       pass:";
2  command += pass;
3  system(command.c_str());
```

3. Удаление файла не зашифрованной базы данных

```
1   if (remove("file.txt") != 0) {
2       cout << "[ERROR] - deleting file" << endl;
3   }
```

4. Запись пароля в файл

```
1   ofstream file;
2   file.open("key.txt", ios::binary);
3   file.write(pass, 65);
4   file.close();
```

5. Шифрование алгоритмом RSA файла с паролем с помощью публичного ключа

```
1   command = "openssl\\bin\\openssl.exe rsautl -encrypt -inkey rsa.public -pubin -in key.txt -out key.txt.enc
2   system(command.c_str());
```

6. Удаление не зашифрованного файла с паролем

```
1   if (remove("key.txt") != 0) {
2       cout << "[ERROR] - deleting file" << endl;
3   }
```

7.4.1 Реализация функции *Decrypt( )*

Для реализации функции ***Decrypt*** поэтапно реализуем следующее:

1. Расшифровка зашифрованного файла с паролем с помощью приватного ключа и RSA

```
1   string command = "openssl\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.txt.enc -out key.txt
2   system(command.c_str());
```

2. Удаление зашифрованного файла с ключом

```
1   if (remove("key.txt.enc") != 0) {
2       cout << "[ERROR] - deleting file" << endl;
3   }
```

3. Считывание ключа из файла

```
1   char* pass = new char[64];
2   ifstream file;
3   file.open("key.txt", ios::binary);
```

```
4       file.read(pass, 65);
5       file.close();
```

4. Удаление файла с ключом

```
1       if (remove("key.txt") != 0) {
2           cout << "[ERROR] - deleting file" << endl;
3       }
```

5. Расшифровка зашифрованной базы данных с помощью ключа и алгоритма AES

```
1       command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -d -in file.txt.enc -out file.txt -pass pass:";
2       command += pass;
3       system(command.c_str());
```

6. Удаление зашифрованной базы данных

```
1       if (remove("file.txt.enc") != 0) {
2           cout << "[ERROR] - deleting file" << endl;
3       }
```

7.4.2 Полный листинг кода

```
1       #include <iostream>
2       #include <windows.h>
3       #include <string>
4       #include <time.h>
5       #include <fstream>
6
7       using namespace std;
8
9       void Crypt();
10      void Decrypt();
11
12      int main()
13      {
14      }
15
16      void Crypt()
17      {
18          srand(time(NULL));
19          char* pass = new char[64];
20          for (int i = 0; i < 64; ++i) {
21              switch (rand() % 3) {
22              case 0:
23                  pass[i] = rand() % 10 + '0';
24                  break;
25              case 1:
26                  pass[i] = rand() % 26 + 'A';
27                  break;
28              case 2:
```

```
29                  pass[i] = rand() % 26 + 'a';
30              }
31          }
32
33          string command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -salt -in
34      file.txt -out file.txt.enc -pass pass:";
35          command += pass;
36          system(command.c_str());
```

```
    if (remove("file.txt") != 0) {
        cout << "[ERROR] - deleting file" << endl;
    }

    ofstream file;
    file.open("key.txt", ios::binary);
    file.write(pass, 65);
    file.close();

    command = "openssl\\bin\\openssl.exe rsautl -encrypt -inkey rsa.public -
pubin -in key.txt -out key.txt.enc";
    system(command.c_str());
    if (remove("key.txt") != 0) {
        cout << "[ERROR] - deleting file" << endl;
    }
}

void Decrypt()
{
    string command = "openssl\\bin\\openssl.exe rsautl -decrypt -inkey
rsa.private -in key.txt.enc -out key.txt";
    system(command.c_str());
    if (remove("key.txt.enc") != 0) {
        cout << "[ERROR] - deleting file" << endl;
    }

    char* pass = new char[64];
    ifstream file;
    file.open("key.txt", ios::binary);
    file.read(pass, 65);
    file.close();
    if (remove("key.txt") != 0) {
        cout << "[ERROR] - deleting file" << endl;
    }

    command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -d -in file.txt.enc -
out file.txt -pass:";
    command += pass;
    system(command.c_str());
    if (remove("file.txt.enc") != 0) {
        cout << "[ERROR] - deleting file" << endl;
    }
}
```

СПИСОК ЛИТЕРАТУРЫ

1. Брайан У. Керниган, Роб Пайк. Практика программирования. – М.:Вильямс. – 2017, 288 с.
2. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения. – СПБ.: Питер. – 2018, 352 с.
3. Майкл Ховард, Дэвид Лебланк, Джон Виега. Как написать безопасный код на C++, Java, Perl, PHP, ASP.NET. – М.:ДМК Пресс. – 2017, 288 с.
4. Сергей Никифоров. Прикладное программирование. – М.:Лань, 2018, 124 с.
5. Э Фримен, Э Фримен, К Сьерра, Б Бейтс. Паттерны проектирования //СПб.: Питер. – 2018, 656 с.
6. И.Г. Гниденко, Ф.Ф. Павлов, Д.Ю. Федоров Технологии и методы программирования. – М.: Юрайт. – 2017, 235 с.
7. Адитья Бхаргава, Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – М.:Издательство «Питер». – 2017, 288 с.
8. Дино Эспозито, Андреа Сальтарелло. Microsoft .NET. Архитектура корпоративных приложений. – М.:Вильямс. – 2017, 432 с.

ПРИЛОЖЕНИЯ

Примеры оформления таблиц

Таблица 1 – Результаты численных расчетов в простой таблице

Время	Вероятность безотказной работы	Вероятность отказов	Частота отказов	Интенсивность потока отказов	Наработка до отказа (расчет)	Дисперсия времени работы до отказа (расчет)
T	P(t)	Q(t)	f(t)	f(t)/P(t)	Расч_T_0	Расч D(T)
0	1	0	0,000125	0,000125	0	0
500	0,939413063	0,06058694	0,000117	0,000125	484,85327	3651312,024
1000	0,882496903	0,1175031	0,00011	0,000125	455,47749	3002635,407
49000	0,002187491	0,99781251	2,73E–07	0,000125	1,1290158	234269,1587
49500	0,002054958	0,99794504	2,57E–07	0,000125	1,0606122	225510,1101
50000	0,001930454	0,99806955	2,41E–07	0,000125	0,996353	217014,7515

Таблица 2 – Результаты численных расчетов в многостраничной таблице

Время	Вероятность безотказной работы	Вероятность отказов	Частота отказов	Интенсивность потока отказов	Наработка до отказа (расчет)	Дисперсия времени работы до отказа (расчет)
T	P(t)	Q(t)	f(t)	f(t)/P(t)	Расч_T_0	Расч D(T)
1	2	3	4	5	6	7
0	1	0	0,000125	0,000125	0	0

1	2	3	4	5	6	7
49000	0,002187491	0,99781251	2,73E–07	0,000125	1,1290158	234269,1587
49500	0,002054958	0,99794504	2,57E–07	0,000125	1,0606122	225510,1101
50000	0,001930454	0,99806955	2,41E–07	0,000125	0,996353	217014,7515

Продолжение таблица 2

Таблица 3 – Результаты численных расчетов в многостраничной таблице

Время	Вероятность безотказной работы	Вероятность отказов	Частота отказов	Интенсивность потока отказов	Наработка до отказа (расчет)	Дисперсия времени работы до отказа (расчет)
T	P(t)	Q(t)	f(t)	f(t)/P(t)	Расч_T_0	Расч D(T)
0	1	0	0,000125	0,000125	0	0

Продолжение таблица 3

Время	Вероятность безотказной работы	Вероятность отказов	Частота отказов	Интенсивность потока отказов	Наработка до отказа (расчет)	Дисперсия времени работы до отказа (расчет)
T	P(t)	Q(t)	f(t)	f(t)/P(t)	Расч_T_0	Расч D(T)

49000	0,002187491	0,99781251	2,73E–07	0,000125	1,1290158	234269,1587
49500	0,002054958	0,99794504	2,57E–07	0,000125	1,0606122	225510,1101

Примеры оформления формул

$$З_{эо} = \sum_{i=1}^{m} C_i^{мч} t_i^{м},$$	(5.5)

где $З_{эо}$ – затраты на содержание и эксплуатацию оборудования (руб.);

$C_i^{мч}$ – расчётная себестоимость одного машино-часа работы оборудования на i-й технологической операции (руб./м-ч);

$t_i^{м}$ – количество машино-часов, затрачиваемых на выполнение i-й технологической операции (м-ч).

Примеры оформления источников

Нормативные документы:

1 Федеральный закон «О банках и банковской деятельности» от 02.12.1990 № 395-1.

2 ГОСТ 7.32-2017. Отчет о научно-исследовательской работе. Структура и правила оформления. -М.: Стандартинформ, 2017.-32 с.

3 ГОСТ Р 51275-2006 Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения. -М.: Стандартинформ, 2017.-8 с.

4. "Гражданский кодекс Российской Федерации (часть первая)" от 30.11.1994 № 51-ФЗ (ред. от 16.12.2019).

Книги, монографии:

1 Панин В.В. Основы теории информации: 3-е изд., испр. -М.: Бином: Лаборатория знаний, 2016.-438 с.

2 Голиков А.М. Модуляция, кодирование и моделирование в телекоммуникационных системах. Теория и практика: Учебное пособие.- Изд. "Лань", 2019.-452 с.

Учебные пособия:

1 Аппаратные и программные средства защиты информации: Учебное пособие / Душкин А.В., Кольцов А., Кравченко А. – Воронеж: Научная книга, 2016.–232 с.

Статьи в периодических изданиях и сборниках статей:

1 Зуев А.С., Болбаков Р.Г. О телекоммуникационных сервисах на основе технологии виртуальной реальности // Российский технологический журнал. 2017. Т.5 № 6. С.3-10.

2 Мамедов Ш.Г., Лебедев А.С. Система автоматизированного распараллеливания линейных программ для машин с общей и распределенной памятью //Российский технологический журнал. 2019. Т.7 № 5. С.7-19. https://doi.org/10.32362/2500-316X-2019-7-5-7-19.

Электронные ресурсы:

1 Уровни конфиденциальности информации [Электронный ресурс].-URP: http://5rik.ru/best/best-144011.php (дата обращения 12.04.2020).

2 Руководство администратора ОС «Astra Linux Special Edition» [Электронный ресурс].-URP: http://www.astralinex.ru/rukovodstvo-administratora-chast-1-astra-se.pdf (дата обращения 12.04.2020).

3 Аппаратные и программные средства защиты информации: Учебное пособие / Душкин А.В., Кольцов А., Кравченко А. - Воронеж: Научная книга, 2016.-232 с. – [Электронный ресурс].-URL: https://znanium.com/ catalog/product/923168.