СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc135245884)

[АНОТАЦИЯ 4](#_Toc135245885)

[ВВЕДЕНИЕ 5](#_Toc135245886)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc135245887)

[1.1 Структурное программирование 7](#_Toc135245888)

[1.2 Процедурное программирование 8](#_Toc135245889)

[1.3 Модульное программирование 9](#_Toc135245890)

[1.4 Ссылки 10](#_Toc135245891)

[1.5 Работа с памятью. Указатели 11](#_Toc135245892)

[1.6 Работа с файлами 12](#_Toc135245893)

[1.7 Динамические структуры данных 12](#_Toc135245894)

[1.8 Линейный список 12](#_Toc135245895)

[1.9 Функции 13](#_Toc135245896)

[1.10 Объектно-ориентированное программирование 14](#_Toc135245897)

[1.11 Конструкторы и деструкторы 15](#_Toc135245898)

[1.12 Классы и дружественные функции 16](#_Toc135245899)

[1.12.1 Описание классов и их методов 17](#_Toc135245900)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 18](#_Toc135245901)

[2.1 Структура программы 18](#_Toc135245902)

[2.2 Алгоритм решения индивидуального задания 27](#_Toc135245903)

[2.3 Программная реализация задания 29](#_Toc135245904)

[2.3.1 Пример выполнения задания 29](#_Toc135245905)

[2.3.2 Отображение списка студентов 29](#_Toc135245906)

[2.3.3 Изменение данных о студенте 29](#_Toc135245907)

[2.3.3 Добавление информации о новом студенте 29](#_Toc135245908)

[2.3.4 Удаление данных о студенте 29](#_Toc135245909)

[2.4 Руководство пользователя 29](#_Toc135245910)

[2.8 Системные требования 30](#_Toc135245911)

[ЗАКЛЮЧЕНИЕ 31](#_Toc135245912)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc135245913)

[ПРИЛОЖЕНИЕ А Исходный код программы 33](#_Toc135245914)

АНОТАЦИЯ

Тема курсовой работы: Разработка программы информационного поиска студентов по заданным критериям с возможностью шифрования данных

Выполнил: Чернитевич Максим Ильич, БББО-06-22

Работа состоит из введения, двух глав, заключения и списка используемых источников.

Во введении рассмотрены цели, задачи работы и ее актуальность.

В первой главе, относящейся к теоретической части, рассмотрены и приведены основные термины, понятия и определения из языка программирования С++.

Во второй главе, относящейся к практической части, приведены листинги кода, представлена программа, выполняющая поставленную задачу, а приведены блок схемы и разобраны алгоритмы ее работы.

В заключении приведены основные выводы, полученные в хотя выполнения работы.

Ключевые слова: С++, объектно-ориентированное программирование, шифрование и дешифрование, шифрование данных, проектирование базы информационного поиска студентов, поиск по заданным критериям.

ВВЕДЕНИЕ

Задачей курсовой работы является: создание программы на языке программирования c++, представляющая собой базу данных студентов с возможностями ввода и вывода информации на экран; чтение и запись базы данных в файл и из файла в бинарном виде; изменение, добавление, удаление информации о студенте; сортировка группы по успеваемости в каждой сессии или сессии, вводимой по требованию пользователя.

Цель курсовой работы по дисциплине «Языки программирования» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа предполагает выполнение задания повышенной сложности по проектированию, разработке и тестированию программного обеспечения, а также оформлению сопутствующей документации.

В результате изучения дисциплины обучающийся должен:

Знать:

1) языки, системы и инструментальные средства программирования в профессиональной деятельности;

2) математический аппарат, математически пакеты, программные комплексы;

3) общие принципы построения и использования современных языков программирования высокого уровня.

Уметь:

1) использовать языки, системы и инструментальные средства программирования в профессиональной деятельности;

2) строить алгоритм решения задачи, проводить его анализ и реализовывать в современных программных комплексах;

3) работать с интегрированной средой разработки программного обеспечения.

Владеть:

1) языками программирования, системами и инструментальными средствами программирования в профессиональной деятельности;

2) навыками разработки, документирования, тестирования и отладки программ на языке программирования высокого уровня;

3) основными методами разработки алгоритмов и программ;

4) методами создания структур данных, используемые для представления типовых информационных объектов.

Задачи курсовой работы:

1) проанализировать исходные данные, указанные в задании;

2) определить данные, структуры, классы, методы и функции, необходимые для выполнения работы согласно варианту;

3) разработать соответствующей алгоритм решения конкретной задачи;

4) реализовать элементы, описанные в пункте 2;

5) подготовить контрольные данные для тестирования программного обеспечения;

6) отладить разработанное программное обеспечение на основе контрольных данных, подготовленных в предыдущем пункте.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Структурное программирование

Структурное программирование воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

1) алгоритм и программа должны составляться поэтапно;

2) сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход;

3) логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Принципы структурного программирования:

1) программа построена из трёх базовых управляющих конструкций: последовательность, ветвление, цикл;

2) в программе базовые управляющие конструкции вложены друг в друга произвольным образом;

3) повторяющиеся фрагменты программы были оформлены в виде подпрограмм (процедур и функций);

4) перечисленные конструкции должны имеют один вход и один выход;

5) разработка программы ведётся пошагово, методом «сверху вниз».

1.2 Процедурное программирование

Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Процедурное программирование является отражением архитектуры традиционных ЭВМ.

Особенности процедурного программирования:

1) предопределенные функции. Предопределенная функция — это инструкция, идентифицируемая именем. Обычно предопределенные функции встроены в языки программирования более высокого уровня, но они получены из библиотеки или реестра, а не из программы;

2) локальная переменная. Локальная переменная — это переменная, которая объявлена, в основной структуре метода и ограничена локальной областью действия, которую она задает. Локальная переменная может использоваться только в том методе, в котором она определена;

3) глобальная переменная. Глобальная переменная — это переменная, которая объявляется вне любой другой функции, определенной в коде. Благодаря этому глобальные переменные могут использоваться во всех функциях, в отличие от локальной переменной;

4) модульность: Модульность — это когда две разные системы имеют под рукой две разные задачи, но сгруппированы вместе, чтобы сначала выполнить более крупную задачу. В этом случае каждая группа систем будет выполнять свои собственные задачи один за другим, пока все задачи не будут выполнены;

5) передача параметров: Передача параметров — это механизм, используемый для передачи параметров в функции, подпрограммы или процедуры.

1.3 Модульное программирование

Модуль в программировании — это фрагмент кода, имеющий определенное функциональное значение и характеризующийся логической завершенностью.

Модульное программирование — это способ создания программы посредством объединения модулей в единую структуру.

В основе модульного программирования лежат три основных концепции:

Основные концепции модульного программирования:

1) каждый модуль имеет единственную точку входа и выхода;

2) размер модуля по возможности должен быть минимизирован;

3) вся система построена из модулей;

4) каждый модуль не зависит от того, как реализованы другие модули.

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

1.4 Ссылки

Ссылка — это псевдоним для другой переменной. Они объявляются при помощи символа &. Ссылки должны быть проинициализированы при объявлении, причем только один раз.

Ссылка при определении сразу же инициализируется. Инициализация ссылки производится следующим образом:

int i = 0;

int& iref = i;

Физически iref представляет собой постоянный указатель на int - переменную типа int\* const.

Поскольку ссылка — это псевдоним, то при передаче объекта в функцию по ссылке внутри нее объект можно изменять. Ссылки не могут ссылаться на другие ссылки или на поле битов. Не может быть массивов ссылок или указателей на ссылку. Ссылка может использоваться для возврата результата из функции. Возвратить результат по ссылке — значит возвратить не указатель на объект и не его значение, а сам этот объект.

1.5 Работа с памятью. Указатели

Указатели являются одним из основных понятий языка Си. В такие переменные можно записывать адреса любых участков памяти, на чаще всего – адрес начального элемента динамического массива.

Важно знать, что:

1) указатель – это переменная, в которой записан адрес другой переменной;

2) при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед именем поставить знак \*;

3) знак & перед именем переменной обозначает ее адрес;

4) знак \* перед указателем в рабочей части программы (не в объявлении) обозначает значение ячейки, на которую указывает указатель;

5) нельзя записывать по указателю, который указывает непонятно куда – это вызывает сбой программы, поскольку что-то стирается в памяти;

6) для обозначения недействительного указателя используется константа NULL;

7) при изменении значения указателя на n он в самом деле сдвигается к n-ому следующему числу данного типа, то есть для указателей на целые числа на n\*sizeof(int) байт;

8) указатель печатаются по формату %p.

1.6 Работа с файлами

Файл – это именованная область ячеек памяти, в которой хранятся данные одного типа. Файл имеет следующие характерные особенности: уникальное имя; однотипность данных; произвольная длина, которая ограничивается только емкостью диска.

В C++ для работы с файлами используются потоки ввода-вывода (ifstream и ofstream) и функции для чтения и записи данных, описанные в заголовочном файле fstream.

Для записи или чтения данных в файл в бинарном формате можно использовать функцию, которая принимает указатель на массив байт и количество байт для записи или чтения.

1.7 Динамические структуры данных

Часто в серьезных программах надо использовать данные, размер и структура которых должны меняться в процессе работы. Динамические массивы здесь не выручают, поскольку заранее нельзя сказать, сколько памяти надо выделить – это выясняется только в процессе работы. Например, надо проанализировать текст и определить, какие слова и в каком количество в нем встречаются, причем эти слова нужно расставить по алфавиту. В таких случаях применяют данные особой структуры, которые представляют собой отдельные элементы, связанные с помощью ссылок.

1.8 Линейный список

В простейшем случае каждый узел содержит всего одну ссылку. Для определенности будем считать, что решается задача частотного анализа текста – определения всех слов, встречающихся в тексте и их количества. В этом случае область данных элемента включает строку (длиной не более 40 символов) и целое число.

1.9 Функции

Функция определяет действия, которые выполняет программа. Функции позволяют выделить набор инструкций и придать ему имя. А затем многократно по присвоенному имени вызывать в различных частях программы. По сути функция — это именованный блок кода.

Функцию можно вызвать из любого количества мест в программе. Значения, передаваемые функции, являются аргументами, типы которых должны быть совместимы с типами параметров в определении функции.

Длина функции практически не ограничена, однако для максимальной эффективности кода целесообразно использовать функции, каждая из которых выполняет одиночную, четко определенную задачу. Сложные алгоритмы лучше разбивать на более короткие и простые для понимания функции, если это возможно. Функции могут быть перегружены, что означает, что разные версии функции могут использовать одно и то же имя, если они отличаются числом и типом формальных параметров.

Определение функции состоит из объявления и тела функции, заключенных в фигурные скобки, которые содержат объявления переменных, операторы и выражения. В следующем примере показано полное определение функции.

1.10 Объектно-ориентированное программирование

Объектно-ориентированное программирование – это подход, при котором вся программа рассматривается как набор взаимодействующих друг с другом объектов. При этом нам важно знать их характеристики.

Основные задачи ООП — структурировать код, повысить его читабельность и ускорить понимание логики программы. Косвенно выполняются и другие задачи: например, повышается безопасность кода и сокращается его дублирование.

Такой подход помогает строить сложные системы более просто и естественно благодаря тому, что вся предметная область разбивается на объекты и каждый из них слабо связан с другими объектами. Слабая связанность возникает вследствие соблюдения трех принципов: инкапсуляции, наследования и полиморфизма.

1) инкапсуляция – сокрытие поведения объекта внутри него. Объекту «водитель» не нужно знать, что происходит в объекте «машина», чтобы она ехала. Это ключевой принцип ООП;

2) наследование. Есть объекты «человек» и «водитель». У них есть явно что-то общее. Наследование позволяет выделить это общее в один объект (в данном случае более общим будет человек), а водителя — определить, как человека, но с дополнительными свойствами и/или поведением. Например, у водителя есть водительские права, а у человека их может не быть;

3) полиморфизм – это переопределение поведения. Можно снова рассмотреть «человека» и «водителя», но теперь добавить «пешехода». Человек умеет как-то передвигаться, но как именно, зависит от того, водитель он или пешеход. То есть у пешехода и водителя схожее поведение, но реализованное по-разному: один перемещается ногами, другой – на машине.

ООП позволяет упростить сложные объекты, составляя их из более маленьких и простых, поэтому над программой могут работать сотни разработчиков, каждый из которых занят своим блоком. Большинство современных языков программирования — объектно-ориентированные, и, однажды поняв суть, вы сможете освоить сразу несколько языков.

1.11 Конструкторы и деструкторы

Конструктор – это функция-член, имя которой совпадает с именем класса, инициализирующая переменные-члены, распределяющая память для их хранения.

Деструктор – это функция-член, имя которой представляет собой имя класса, предназначенная для уничтожения переменных.

При создании объектов одной из наиболее широко используемых операций которую вы будете выполнять в ваших программах, является инициализация элементов данных объекта. Чтобы упростить процесс инициализации элементов данных класса, C++ использует специальную функцию, называемую конструктором, которая запускается для каждого создаваемого вами объекта. Также C++ обеспечивает функцию, называемую деструктором, которая запускается при уничтожении объекта.

Конструктор представляет собой метод класса, который облегчает вашим программам инициализацию полей при создании объекта класса. Конструктор имеет такое же имя, как и сам класс. Конструктор не имеет возвращаемого значения. Конструкторы относят к интерфейсу класса, чтобы с их помощью можно было создавать объекты данного класса из внешней части программы.

Таким образом, деструктор не может быть перегружен и должен существовать в классе в единственном экземпляре. Деструктор вызывается автоматически при уничтожении объекта.

1.12 Классы и дружественные функции

Класс представляет составной тип, который может использовать другие типы. Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования — ООП.

Класс предназначен для описания некоторого типа объектов. То есть класс является планом объекта. А объект представляет конкретное воплощение класса, его реализацию. Объекты — конкретное представление абстракции, которые имеют свои свойства и методы. Свойства — это любые данные, которыми можно характеризовать объект класса. Методы — это функции, выполняющие различные действия над данными (свойствами) класса. Поле класса в объектно-ориентированном программировании — переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях.

Класс может определять переменные и константы для хранения состояния объекта и функции для определения поведения объекта.

На объекты классов, как и на объекты других типов, можно определять указатели. Затем через указатель можно обращаться к членам класса - переменным и методам. Однако если при обращении через обычную переменную используется символ точка, то для обращения к членам класса через указатель применяется стрелка (->).

Дружественные функции — это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

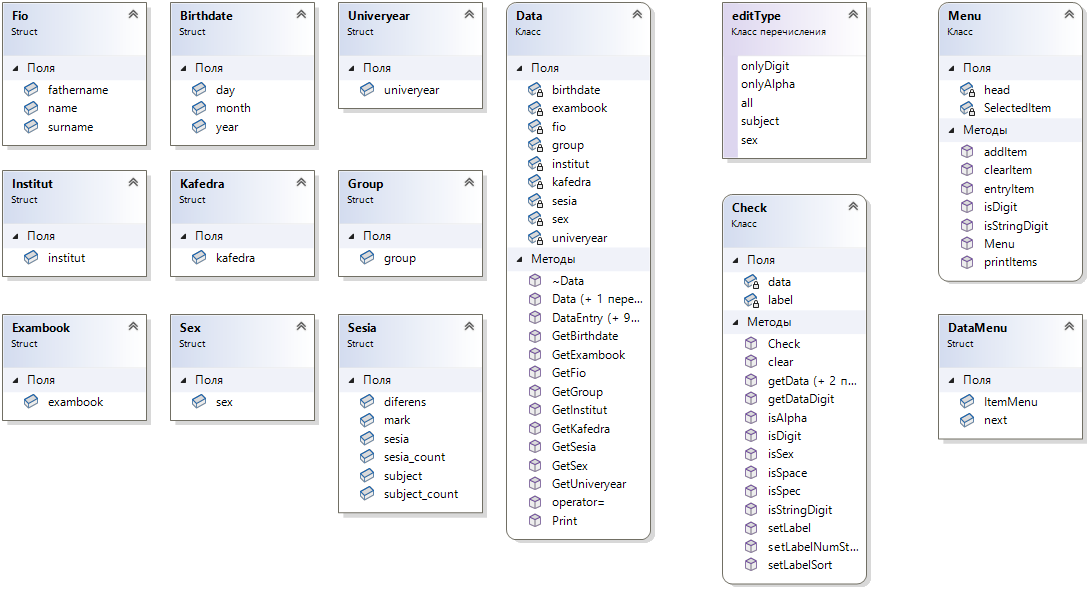
Для определения дружественных функций используется ключевое слово friend.

1.12.1 Описание классов и их методов

Классы в языке программировании С++ — это абстракция, которая описывает методы и свойства, ещё не существующих объектов. Объекты — конкретное представление абстракции, которые имеют свои свойства и методы. Свойства — это любые данные, которыми можно характеризовать объект класса. Методы — это функции, выполняющие различные действия над данными (свойствами) класса. Поле класса в объектно-ориентированном программировании — переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях.

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте.

Диаграмма классов:



2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Структура программы

Исходными данными для программы является информация о группе студентов из N человек, где запись о студенте содержит следующие данные (таблица 2.1.1).

Таблица 2.1.1 – Структура и типы данных

|  |  |
| --- | --- |
| Основная структура данных | Типы данных |
| Фамилия | string(строковый) |
| Имя | string(строковый) |
| Отчество | string(строковый) |
| День рождения | int(целочисленный) |
| Месяц рождения | int(целочисленный) |
| Год рождения | int(целочисленный) |
| Год поступления в институт | int(целочисленный) |
| Институт | string(строковый) |
| Кафедра | string(строковый) |
| Группа | string(строковый) |
| Номер зачётной книжки | string(строковый) |
| Пол студента | string(строковый) |
| Предмет | string(строковый) |
| Зачёты | string(строковый) |
| Оценка | unsigned int(беззнаковый целочисленный) |
| Номер сессии | unsigned int(беззнаковый целочисленный) |
| Количество предметов | short unsigned int(короткий беззнаковый целочисленный) |
| Количество сессий | short unsigned int(короткий беззнаковый целочисленный) |

По условиям курсовой работы, допустимо максимально 9 сессий и 10 предметов в каждом семестре, которые могут быть разными. Все данные являются форматными.

Для реализации соответствующей задачи были разработаны классы. В данных классах описаны методы решения поставленной задачи.

В представленной ниже таблице продемонстрированы роли используемых классов (таблица 2.1.2).

Таблица 2.1.2 – Классы

|  |  |
| --- | --- |
| Класс | Назначение |
| Data | Этот класс служит для инициализации структур, а также для отображения и ввода данных о студентах. |
| Check | Этот класс служит для форматного ввода с клавиатуры пользователя (на пример: только цифры или только буквы и т.д.) данных о студенте. |
| editType | Использование класса перечисления (enum class) позволяет избежать ошибок, связанных с неявным преобразованием элементов перечисления в целочисленные значения и обеспечивает более безопасное использование перечислений в программе. |
| Menu | Этот класс служит для отображения главного меню. |

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте (таблица 2.1.3).

Таблица 2.1.3 – Поля классов

|  |  |  |
| --- | --- | --- |
| Классы | Поле класса | |
| Тип данных | Название и характеристика |
| Data | Fio, Birthdate, Univeryear, Institut, Kafedra, Group, Exambook, Sex, Sesia | Fio — данные из структуры Fio (фамилия, имя, отчество); Birthdate — данные из структуры Birthdate (день, месяц, год рождения); Univeryear — данные из структуры Univeryear (год поступления в институт); Institut — данные из структуры Institut (название института); Kafedra — данные из структуры Kafedra (названия кафедры); Group — данные из структуры Group (название группы); Exambook — данные из структуры Exambook (номер зачётной книжки); Sex — данные из структуры (пол студента); Sesia — данные из структуры Sesia (предметы, зачёты, оценка, номер сессии, количество предметов и сессий). |
| Check | string label; string data | string label — строковое значение заголовка; string data — строковое значение вводимых данных. |
| editType | onlyDigit; onlyAlpha; all; subject; sex | onlyDigit — элемент для числовых значений; onlyAlpha — элемент для буквенных значений; |

Продолжение таблицы 2.1.3

|  |  |  |
| --- | --- | --- |
| Классы | Поле класса | |
| Тип данных | Название и характеристика |
| editType | onlyDigit; onlyAlpha; all; subject; sex | all — элемент для любых значений subject — элемент для значений, использующиеся при вводе названия предмета sex — элемент для значений, использующиеся при вводе названия пола |
| Menu | head; SelectedItem | head — элемент, использующийся для инициализации структуры DataMenu SelectedItem — строковое значение выбранного элемента. |

Далее методы классов, представленные в таблице 2.1.4.

Таблица 2.1.4 Методы классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Data | Data | Конструктор по умолчанию, создаёт пустой объект | - | - |
| Data | Конструктор с параметрами, создаёт объект с параметрами | Fio fio\_, Birthdate birthdate\_, Univeryear univeryear\_, Institut institut\_, Kafedra kafedra\_, Group group\_, Exambook exambook\_, Sex sex\_, Sesia sesia\_ | - |
| ~Data | Деструктор, удаляет объект | - | - |
| Print | Отображение объектов | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Data | DataEntry | Ввод данных в объекты | Fio fio\_, Birthdate birthdate\_, Univeryear univeryear\_, Institut institut\_, Kafedra kafedra\_, Group group\_, Exambook exambook\_, Sex sex\_, Sesia sesia\_ | - | |
| DataEntry | Ввод данных в объект ФИО | Fio fio\_ | - |
| DataEntry | Ввод данных в объект дата рождения | Birthdate birthdate\_ | - |
| DataEntry | Ввод данных в объект год поступления | Univeryear univeryear\_ | - |
| DataEntry | Ввод данных в объект названия института | Institut institut\_ | - |
| DataEntry | Ввод данных в объект название кафедры | Kafedra kafedra\_ | - |
| DataEntry | Ввод данных в объект название группы | Group group\_ | - |
| DataEntry | Ввод данных в объект номер зач. книжки | Exambook exambook\_ | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Data | DataEntry | Ввод данных в объект пола студента | Sex sex\_ | - |
| DataEntry | Ввод данных в объект | Sesia sesia\_ | - |
| GetFio | Вывод данных из объекта FIO | - | fio |
| GetBirthdate | Вывод данных из объекта | - | birthdate |
| GetUniveryear | Вывод данных из объекта | - | univeryear |
| GetInstitut | Вывод данных из объекта | - | institut |
| GetKafedra | Вывод данных из объекта | - | kafedra |
| GetGroup | Вывод данных из объекта | - | group |
| GetExambook | Вывод данных из объекта | - | exambook |
| GetSex | Вывод данных из объекта | - | sex |
| GetSesia | Вывод данных из объекта | - | sesia |
| operator= | Перегрузка оператора “=” | Data d\_o | - |
| Check | Check | Конструктор по умолчанию, создаёт пустой объект | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Check | setLabel | Устанавливает заголовок | string \_label = "Введите значение" | - |
| setLabelNumStud | Устанавливает заголовок для отображения номера студента | int n | - |
| isDigit | Проверяет вводимые данные, чтобы они соответствовали числовым | char ch | true / false |
| isStringDigit | Проверяет вводимую строку, чтобы она соответствовала числовой | string \_str | true / false |
| isAlpha | Проверяет вводимые данные, чтобы они соответствовали буквенным | int ch | true / false |
| isSpace | Проверяет вводимые данные, чтобы они соответствовали пробелу | char ch | true / false |
| isSpec | Проверяет вводимые данные, чтобы они соответствовали специальным символам | char ch | true / false |
| isSex | Проверяет вводимые данные, чтобы они соответствовали символам: “м” или “ж” | char ch | true / false |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Check | clear | Очищает буфер вводимых значений и заголовок | string \_data = "" | - |
| getData | Ввод символов, отображение заголовка и вводимых символов и сравнение с требуемым форматом | enum class editType et | string data |
| getDataDigit | Сравнение с форматом только числовые значение с указанием минимального | enum class editType et, int min | int num |
| getData | Сравнение с форматом только числовые значение с указанием минимального и максимального | enum class editType et, int min, int max | int num |
| getData | Сравнение с форматом значений с указанием длины | enum class editType et, int len | string data |
| setLabelSort | Устанавливает заголовок для сортировки | int n | - |
| Menu | Menu | Конструктор по умолчанию, создаёт пустой объект | - | - |
| isDigit | Проверяет вводимые данные, чтобы они соответствовали числовым | char ch | true / false |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Menu | isStringDigit | Проверяет вводимую строку, чтобы она соответствовала числовой | string \_str | true / false |
| addItem | Добавляет объект меню | string ItemMenu | - |
| printItems | Отображение пунктов меню | - | - |
| clearItem | Очищает буфер вводимых значений и заголовок | string \_SelectedItem = "" | - |
| entryItem | Ввод числового значения выбора пункта меню, с указанием минимального и максимального значения | int min, int max | int num |

Так же для разработки данного проекта понадобятся следующие структуры. В таблице ниже представлены и описаны поля структуры, используемые в проекте (таблица 2.1.5).

Таблица 2.1.5

|  |  |  |
| --- | --- | --- |
| Структура | Поле структуры | |
| Тип данных | Название и характеристика |
| Fio | string | surname — Фамилия;  name — Имя fathername —Отчество |
| Birthdate | int | day — День; month — Месяц; year — год |
| Univeryear | int | univeryear — Год поступления в институт |
| Institut | string | institut —Название института |
| Kafedra | string | kafedra — Название кафедры |
| Group | string | group —Название группы |
| Exambook | string | exambook — Номер зачётной книжки |
| Sex | string | sex — Пол студента |
| Sesia | string, unsigned int, short unsigned int | subject — Строковая матрица для названия предметов mark — Целочисленная матрица для оценок sesia — Целочисленная матрица для названия сессии subject\_count — Целочисленный массив для количества предметов в сессии sesia\_count — Целочисленная переменная для количества сессий deferens — Строковая матрица для зачётов или незачётов |
| DataMenu | string | ItemMenu — Пункт меню |

2.2 Алгоритм решения индивидуального задания

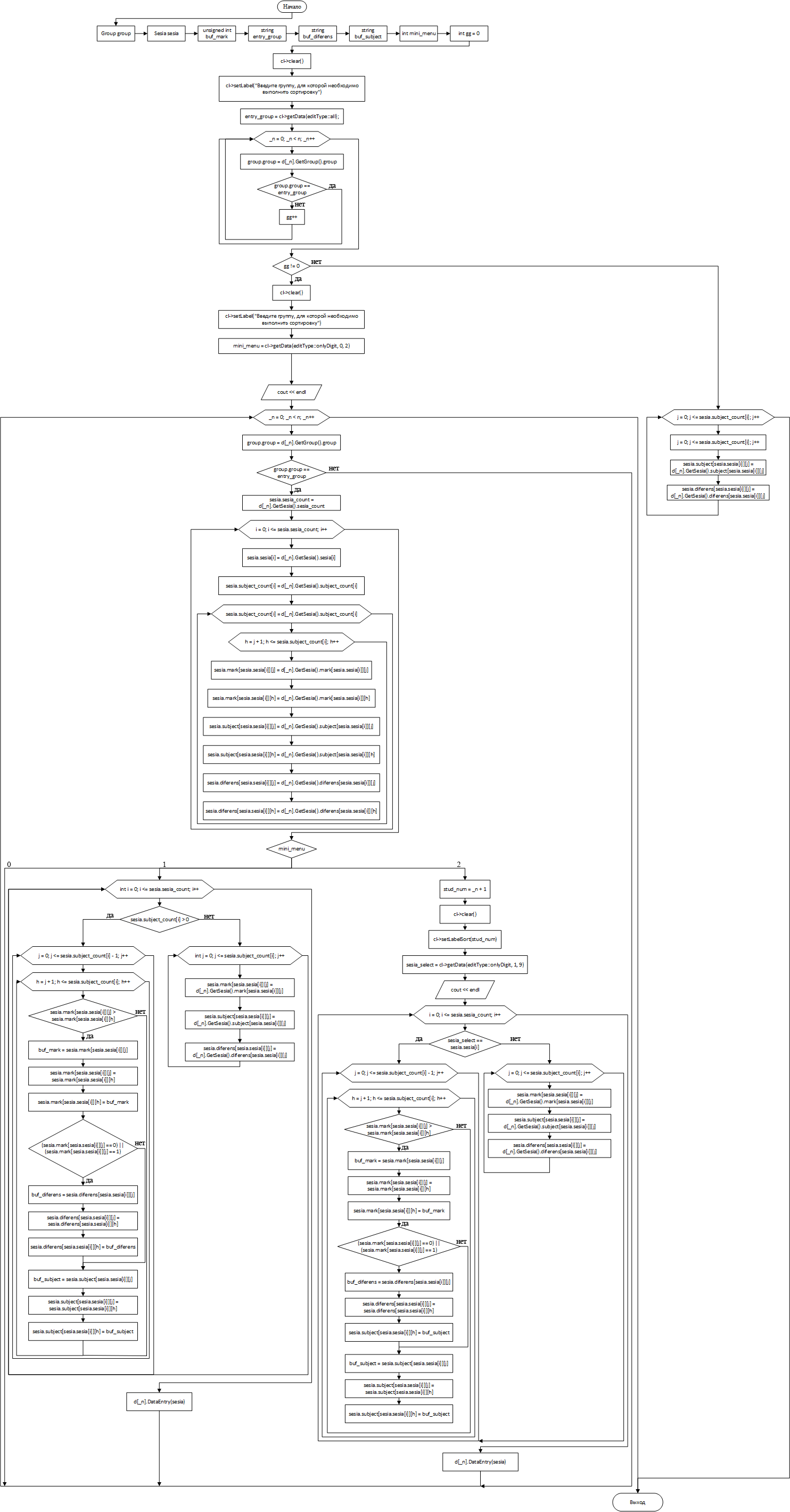
Индивидуальное задание для варианта №27 заключается в сортировке группы по успеваемости в каждой сессии или в сессии, которая вводится пользователем.

Сортировка в программе представлена в виде функции, в которую передаются данные из класса “Data”, а также общее количество студентов.

На первом этапе инициализируются локальные переменные, такие как буфер оценок, предметов введённая пользователем группа, выбор пункта меню и т.д.

На втором этапе от пользователя требуется ввести группу, в которой необходимо выполнить сортировку, а также проверяется существование такой группы в базе данных.

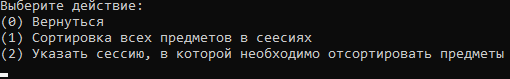
На третьем этапе проводится получение данных из базы (предметы, оценки, номер сессии и т.д.), а также сама сортировка по такой логике: сравнивается оценка, полученная в, к примеру, 1 сессии по такому-то предмету и оценка, полученная в, к примеру, 1 сессии по другому предмету и если первая оценка больше, чем вторая тогда первая оценка отправляется в буфер, а на ей присваивается значение второй оценки. Наконец второй оценке присваивается значение буфера. Если оценка ровна 0 или 1 (незачёт или зачёт), тогда, по аналогии с оценкой, буфер приравнивается к значению зачёт или незачёт первой оценки, первая переменная приравнивается ко второй, а вторая к первой. Аналогично происходит и с предметами. Если условие не выполняется, то с помощью методов класса “Data” переменным не присваиваются новые значения. По данной функции была построена блок схема на рисунке 2.2.1.



*Рисунок 2.2.1 – Блок схема индивидуального задания*

2.2.1 Пример решения индивидуального задания

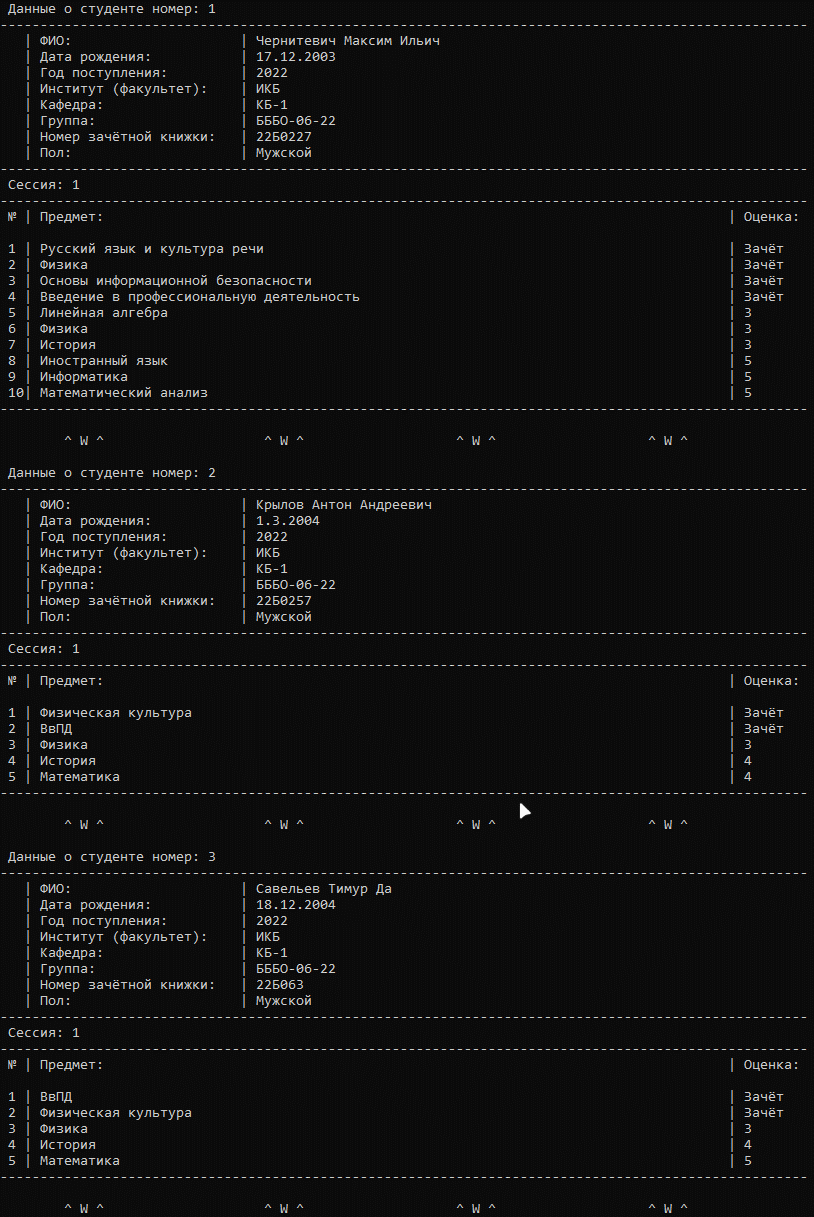
В примере используется база данных для 10 студентов, на примере группы БББО-06-22. Сортировка проводилась по всем сессиям. Что можно увидеть на рисунке 2.2.1.3, а также меню выбора метода сортировки данных на рисунке 2.2.1.1. Так же стоит отметить, что при введении несуществующей (в базе данных) группы, появится сообщение об ошибке, что можно увидеть на рисунке 2.2.1.2.



*Рисунок 2.2.1.1 – Выбор метода сортировки*



*Рисунок 2.2.1.2 – Сообщение об ошибке — несуществующая группа*



*Рисунок 2.2.1.3 – Пример выполнения сортировки*

2.3 Программная реализация задания

Программа представляет собой набор файлов:

* ListMenu.h — файл заголовков, который содержит динамический односвязный класс список, использующийся в файле “course\_work.cpp” для использования меню.
* course\_work.cpp — основной файл, который содержит функцию main, и является меню программы.
* Data.h — файл заголовков, содержащий структуры с именами и типами переменных, использующихся в курсовой работе, а так же класс “Data”, содержащий прототипы конструкторов (с аргументами и без), деструктора и функций. Так же содержит класс “Check”, содержащий две строковые переменные и функции, описанные в таблице 2.1.4.
* Data.cpp — файл, в котором описаны функции, про которых написано в предыдущем пункте.
* Func.h — файл заголовков, содержащий прототипы функций добавления, удаления, сохранения данных и т.д.
* Func.cpp — файл, в котором описаны функции, про которых написано в предыдущем пункте.

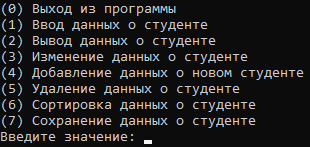
Программа умеет:

* считывать символы определённого типа (числовые, буквенные, все и т.д.) с клавиатуры пользователя, формировать строки из введённых символов и присваивать полученные значения переменным;
* изменять значения ранее полученных значений переменных;
* удалять значения, не принося изменений в остальные;
* сортировать полученные данные;
* записывать и считывать данные в бинарном файле, название которого вводится пользователям самостоятельно;

На рисунках 2.3.1.1 — 2.3.4. содержатся примеры выполнения каждого из вышеуказанных пунктов (все совпадения с реальными людьми случайны).

2.3.1 Пример реализации меню

В главном меню программы пользователю необходимо сделать выбор пункта меню, введя необходимую для этого цифру. Главное меню можно увидеть на рисунке 2.3.1.1 ниже.



*Рисунок 2.3.1.1 – Главное меню программы*

2.3.2 Пример ввода данных о студенте

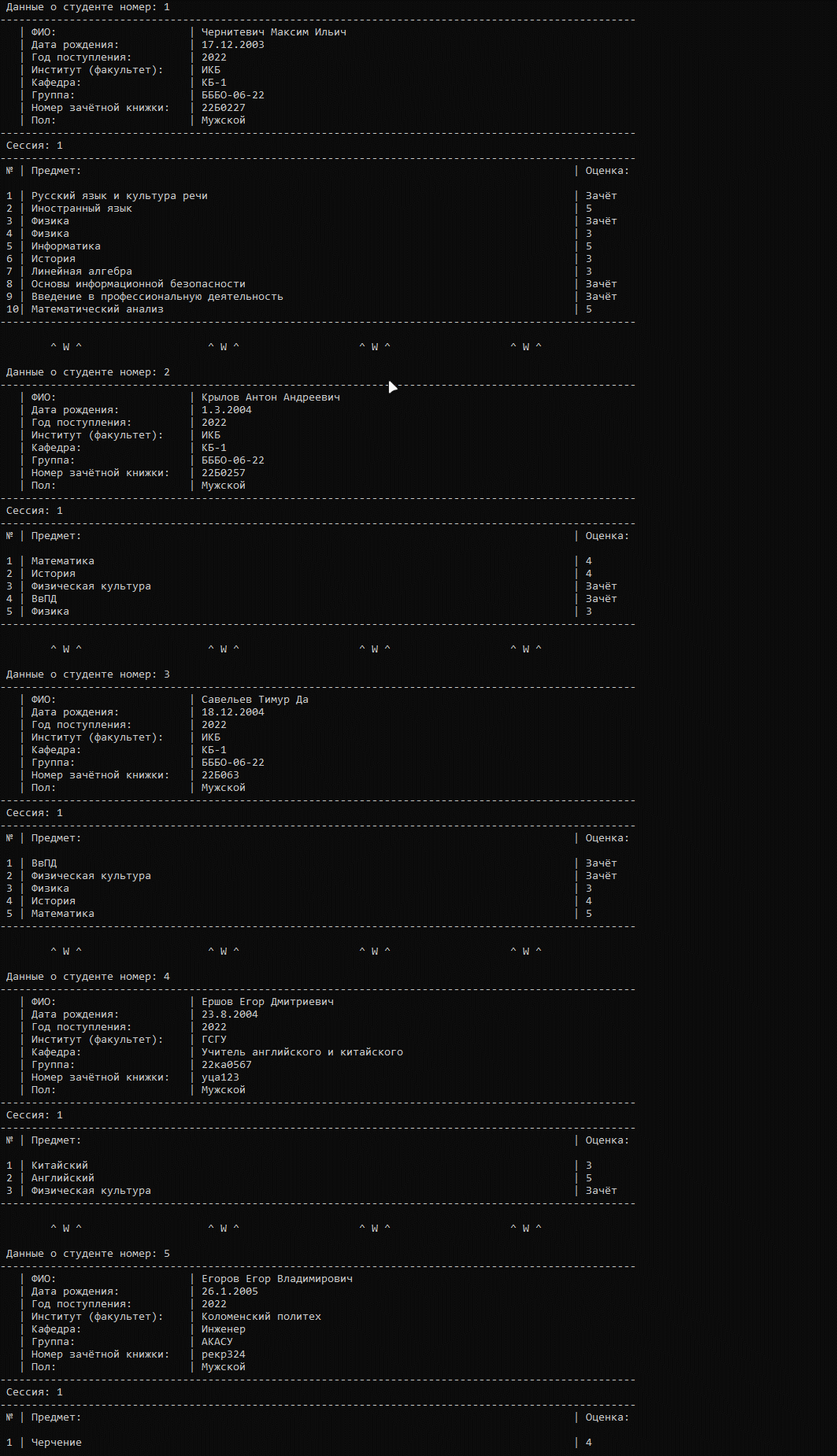
Ввод данных осуществляется либо пользователем, либо считывается из файла. В первом случаи изначально вводится количество студентов, затем фамилия, имя, отчество, день, месяц, год рождения, год поступления (стоит отметить, что пользователь не может ввести год поступление больше чем год рождения этого студента или равное, рисунок 2.3.2.1), институт (факультет), кафедру, группу, номер зачётной книжки, выбрать пол (м – мужской, ж – женский), указать количество сессий, ввести номер сессии, количество предметов, предмет, выбрать тип оценки (дифференцированный зачёт или зачёт/незачёт) и ввести оценку. Во втором случаи пользователь вводит имя файла.



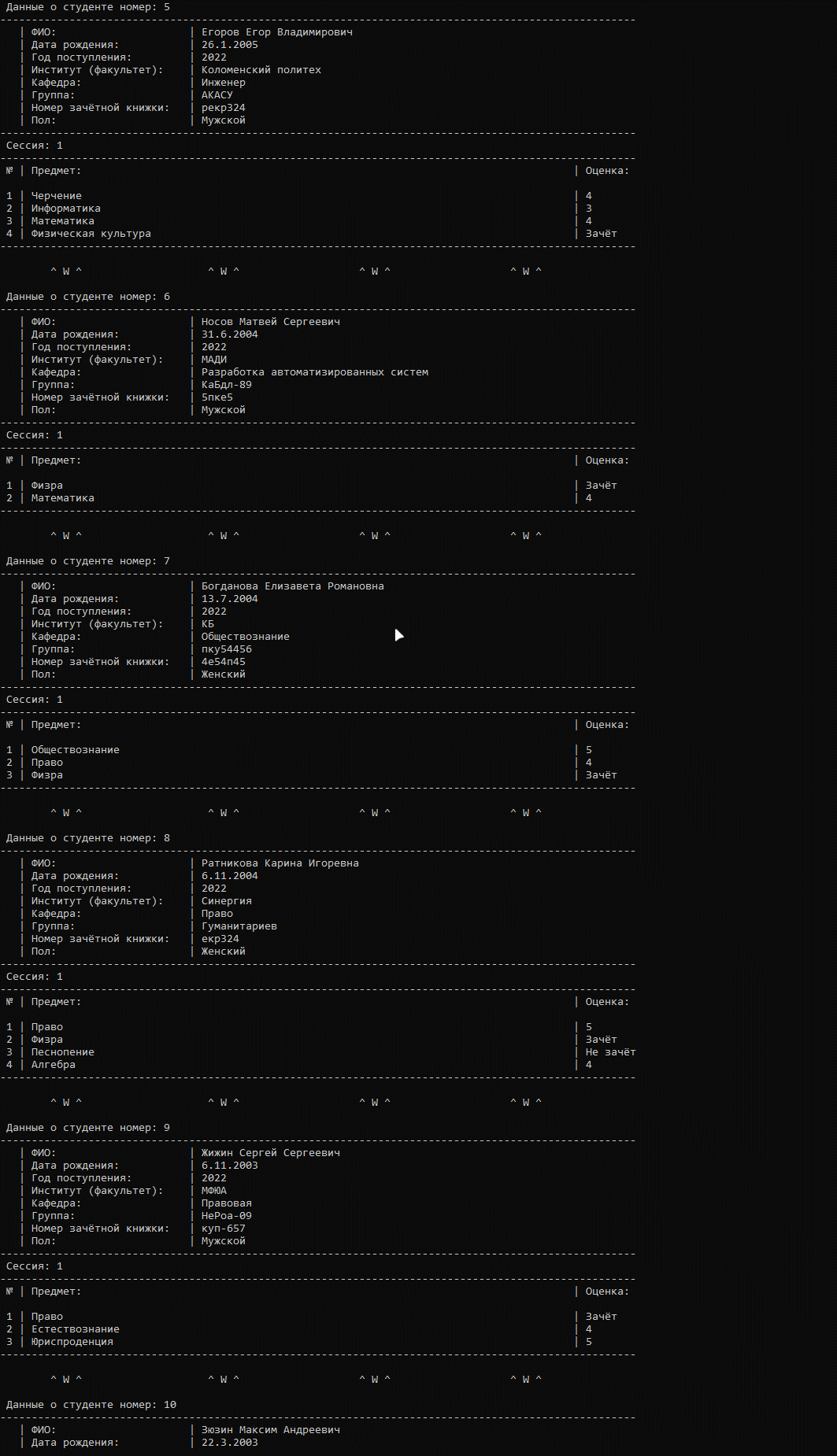
*Рисунок 2.3.2.1– Ошибка. Год рождения равен году поступления*

2.3.3 Отображение списка студентов

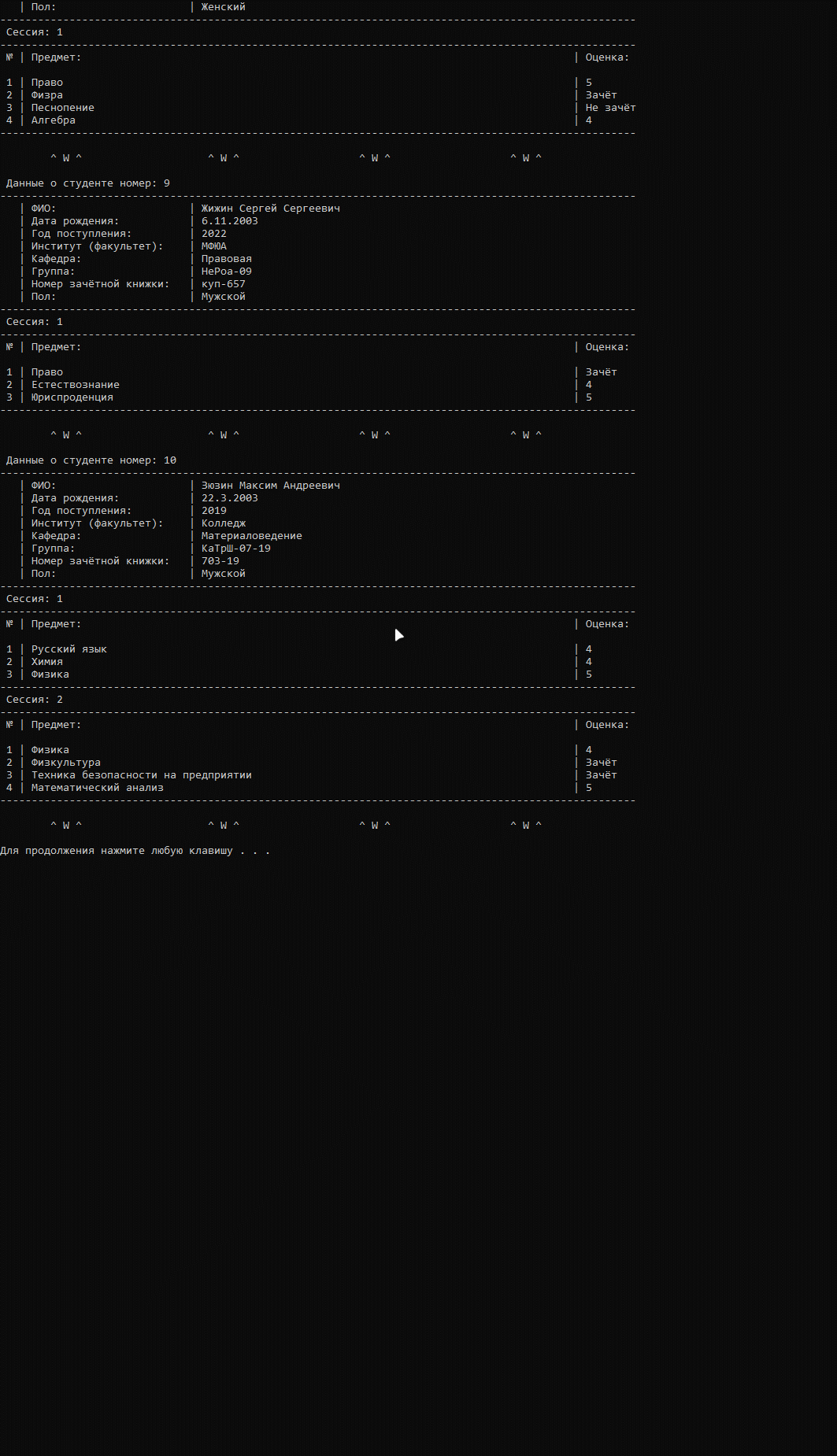
Список студентов содержит все данные, полученные от пользователя, студентов, в виде таблицы. Так же невозможно не упомянуть, что поля имеют свой фиксированный размер (в символах) и при вводе так же учитывается максимальное количество символов, которое может ввести пользователь. Пример такого списка, состоящего из 10 студентов, можно увидеть на рисунках 2.3.3.1 — 2.3.3.3.



*Рисунок 2.3.3.1 – Отображение списка студентов из 10 человек*



*Рисунок 2.3.3.2 – Отображение списка студентов из 10 человек*



*Рисунок 2.3.3.3 – Отображение списка студентов из 10 человек*

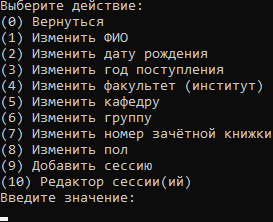
2.3.4 Изменение данных о студенте

Изменение данных о студенте начинается с поля, в которое пользователь вводит номер студента, данные которого необходимо изменить. Такое окно можно увидеть на рисунке 2.3.4.1.



*Рисунок 2.3.4.1 – Изменение данных о студенте*

Далее необходимо выбрать в меню пункт, отвечающий за изменение определённых данных о студенте. Такое меню можно увидеть на рисунке 2.3.4.2.



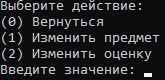
*Рисунок 2.3.4.2 – Меню изменения данных о студенте*

Отдельно будет рассмотрен 10 пункт меню — Редактор сессии. При выборе этого пункта пользователю необходимо ввести номер сессии, в которой необходимо изменить данные. Такое сообщение можно увидеть на рисунке 2.3.3.3.



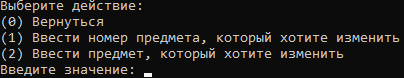
*Рисунок 2.3.4.3 – Поле для ввода сессии*

Дальше перед пользователем стоит выбор: либо вернуться в главное меню, либо изменить предмет, либо изменить оценку. Такое меню можно увидеть на рисунке 2.3.3.4.



*Рисунок 2.3.4.4 – Меню выбора изменения предмета или оценки*

При выборе изменения предмета или оценки пользователю предстоит выбор: либо вернуться в главное меню, либо ввести номер предмета, который необходимо изменить или оценку которого необходимо изменить либо ввести сам предмет. Пример такого меню можно увидеть на рисунке 2.3.3.5.



*Рисунок 2.3.4.5 – Меню выбора изменения предмета*

2.3.5 Добавление информации о новом студенте

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.6 Удаление данных о студенте

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.7 Сортировка данных о студенте

Сортировка данных о студенте является индивидуальным заданием и представлена в пункте 2.2, а пример в 2.2.1.

2.3.8 Сохранение данных о студенте

При сохранении данных пользователю необходимо ввести название файла, в который необходимо сохранить введённые значения, расширение “.bin”

2.4 Руководство пользователя

<ОПИСАНИЕ КАК ПОЛЬЗОВАТЬСЯ ВАШЕЙ ПРОГРАММОЙ>

ПРИМЕР:

При запуске программы на экране выводится консоль с меню, где пользователь может выбрать один из нескольких пунктов, таких как: ввести/добавить студентов, удалить студента, показать всех студентов в университете, изменить информацию о студенте или показать выборку. В зависимости от выбранного пользователем варианта, выполняется определенная часть программы.

При вводе «0» осуществляется выход из программы.

При вводе «1» программа предлагает два способа добавления студентов: либо загрузить из файла, если таковой файл существует, либо ввести студентов с клавиатуры. В этом случае пользователю предлагается внести в список желаемое количество студентов и конкретную информацию о студентах.

При вводе «2» программа запрашивает номер зачетной книжки, далее удаляет соответствующего этому номеру студента.

При вводе «3» программа выводит общий список студентов в университете, определенную информацию о них (ФИО, пол, дата дня рождения, год поступления, группа, номер зачетки, кафедра, факультет).

При вводе «4» запрашивается номер зачетной книжки для изменения определенной информации о студенте.

При вводе «5» программа предлагает указать интервал года рождения, а впоследствии распределяет студентов по группам.

2.8 Системные требования

Язык программирования: С++.

Операционная система: Windows 10 или выше

ОЗУ: 1 Гб и более.

Свободное место на диске: 200 Мб и более.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была написана программа, <ТУТ ВСТАВЛЯЕТЕ ЗАДАНИЕ ПО ВАШЕМУ ВАРИАНТУ>.

Кроме того, в процессе реализации программы мною были закреплены базовые навыки программирования, полученные при изучении дисциплины Языки программирования и языка программирования C++.

Были подробно рассмотрены теоретические выкладки, использованные в процессе написания программы.

Программа была реализована c использованием технологии ООП. Более того, была внедрена работа с динамической памятью, работа с файлами. Более того, детально были разобраны и применены на практике методы симметричного и ассиметричного шифрования и дешифрования файлов.

Дополнительно были созданы необходимые и достаточные условия для корректного выполнения индивидуального варианта задания.

По результату выполнения тестирования программного продукта, можно сделать вывод о том, что программа работает корректно и справляется с поставленной задачей.

Перспективы разработки данного продукта характеризуются высокой потребностью в универсальном и безопасном программном продукте, предназначенном для внедрения в средства автоматизации и информационные системы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Мерсов А. А. Основы объектно-ориентированного программирования на языке С++ [Электронный ресурс]: практикум / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2021. — Электрон. опт. диск (ISO)
2. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2022.
3. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2022.
4. Стенли Липпман Язык программирования С++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2023. 1104 c.
5. Страуструп Б. Дизайн и эволюция языка С++. – Litres, 2022.
6. Уильямс Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. – Litres, 2022.
7. Аммерааль Л. STL для программистов на C++. – Litres, 2022.
8. Дейл Н., Уимз Ч., Хедингтон М. Программирование на C++. – Litres, 2022.
9. Нефедов Д. Г., Русяк И. Г., Вавилова Д. Д. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. – 2020.
10. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – " Издательский дом"" Питер""", 2018.
11. Гудлиф П. Ремесло программиста. – 2009.

ПРИЛОЖЕНИЕ А Исходный код программы

< ВАШ КОД КУРСОВОЙ РАБОТЫ>

<весьма желательно чтобы исходный код был скопирован из Visual Studio 2022 c сохранением форматирования. Не допускается использование скриншотов>

<НАПРИМЕР>

**DB\_of\_students\_new23.cpp**

#include <iostream>

#include "ClassEditData.h"

#include <iostream>

#include <string>

#include "ClassMenu.h"

#include "ClassStudent.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251); // Ввод с консоли в кодировке 1251

SetConsoleOutputCP(1251);

ClassStudent\* st = new ClassStudent();

ClassMenu\* mainMenu = new ClassMenu("Основное меню");

mainMenu->addMenuItem("Выход"); // 0

mainMenu->addMenuItem("Отобразить/изменить данные о студентах"); // 1

mainMenu->addMenuItem("Добавить данные о новом студенте"); // 2

mainMenu->addMenuItem("Выполнить задание"); // 3

int selectedItem = -1;

while (selectedItem != 0){

selectedItem = mainMenu->run();

switch (selectedItem)

{

case 1:

st->getShortInfoFromFile();

break;

case 2:

st->editStudent();

st->addSt2File();

break;

deafault:

break;

}

}

}

**Файл такой то и далее….**

void setStudentData(int num) {

FILE\* binaryFile;

fopen\_s(&binaryFile, filename.c\_str(), "r");

fseek(binaryFile, num \* sizeof(st), SEEK\_SET);

fread\_s(&st, sizeof(st), sizeof(st), 1, binaryFile);

fclose(binaryFile);

}

void write2FileStudentData(int num) {

int size = countRecords();

FILE\* binaryFile;

FILE\* tmpFile;

fopen\_s(&binaryFile, filename.c\_str(), "r");

fopen\_s(&tmpFile, "tmp.txt", "w+");

StudentNode tmp\_st;

for (int i = 0; i < size; i++) {

fread\_s(&tmp\_st, sizeof(tmp\_st), sizeof(tmp\_st), 1, binaryFile);

if (i==num)

fwrite(&st, sizeof(st), 1, tmpFile);

else

fwrite(&tmp\_st, sizeof(tmp\_st), 1, tmpFile);

}

fclose(binaryFile);

fclose(tmpFile);

remove(filename.c\_str());

rename("tmp.txt", filename.c\_str());

}

</НАПРИМЕР>