СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc136295293)

[АНОТАЦИЯ 4](#_Toc136295294)

[ВВЕДЕНИЕ 5](#_Toc136295295)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc136295296)

[1.1 Структурное программирование 7](#_Toc136295297)

[1.2 Процедурное программирование 7](#_Toc136295298)

[1.3 Модульное программирование 8](#_Toc136295299)

[1.4 Ссылки 9](#_Toc136295300)

[1.5 Работа с памятью. Указатели 10](#_Toc136295301)

[1.6 Работа с файлами 10](#_Toc136295302)

[1.7 Динамические структуры данных 11](#_Toc136295303)

[1.8 Линейный список 12](#_Toc136295304)

[1.9 Функции 12](#_Toc136295305)

[1.10 Объектно-ориентированное программирование 13](#_Toc136295306)

[1.11 Конструкторы и деструкторы 14](#_Toc136295307)

[1.12 Классы и дружественные функции 14](#_Toc136295308)

[1.12.1 Описание классов и их методов 15](#_Toc136295309)

[1.13 Методы шифрования и дешифрования 16](#_Toc136295310)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 20](#_Toc136295311)

[2.1 Структура программы 20](#_Toc136295312)

[2.2 Алгоритм решения задачи 26](#_Toc136295313)

[2.3 Программная реализация задания 31](#_Toc136295314)

[2.3.1 Пример выполнения задания 31](#_Toc136295315)

[2.3.2 Отображение списка студентов 32](#_Toc136295316)

[2.3.3 Изменение данных о студенте 33](#_Toc136295317)

[2.3.4 Добавление информации о новом студенте 34](#_Toc136295318)

[2.3.5 Удаление данных о студенте 36](#_Toc136295319)

[2.3.6 Шифрование данных 37](#_Toc136295320)

[2.4 Руководство пользователя 37](#_Toc136295321)

[2.5 Системные требования 40](#_Toc136295322)

[ЗАКЛЮЧЕНИЕ 41](#_Toc136295323)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 42](#_Toc136295324)

[ПРИЛОЖЕНИЕ А Исходный код программы 43](#_Toc136295325)

АНОТАЦИЯ

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

Выполнил: ФИО, НОМЕР ГРУППЫ

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: 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.

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. Структурное программирование

Структурное программирование отражает принципы системного подхода к созданию и использованию программного обеспечения для компьютеров. В основе структурного программирования лежат следующие простые принципы:

Алгоритмы и программы разрабатываются поэтапно.

1. Сложные задачи разбиваются на более простые части, каждая из которых имеет один вход и один выход.
2. Логика алгоритмов и программ опирается на минимальное количество базовых управляющих структур.

Принципы структурного программирования включают:

1. Программа состоит из трех базовых управляющих конструкций: последовательность, ветвление и цикл.
2. Базовые управляющие конструкции могут быть вложены друг в друга произвольным образом.
3. Повторяющиеся фрагменты программы оформляются в виде подпрограмм, таких как процедуры и функции.
4. Указанные конструкции имеют один вход и один выход.
5. Разработка программы осуществляется пошагово, используя метод "сверху вниз".

1.2 Процедурное программирование

Процедурное программирование представляет собой подход к программированию на императивных языках, где последовательность операторов может быть объединена в подпрограммы - целостные блоки кода, с помощью встроенных механизмов языка. Процедурное программирование отражает архитектуру традиционных ЭВМ

Особенности процедурного программирования:

1. Возможность использования предопределенных функций. Предопределенные функции — это инструкции, идентифицируемые по имени. Обычно они встроены в языки программирования более высокого уровня и берутся из библиотек или реестра, а не из самой программы.
2. Использование локальных переменных. Локальная переменная — это переменная, объявленная в основной структуре метода и ограниченная локальной областью видимости, которую она определяет. Локальная переменная может использоваться только в том методе, где она определена.
3. Использование глобальных переменных. Глобальная переменная — это переменная, объявленная вне любой другой функции в коде. Глобальные переменные могут использоваться во всех функциях, в отличие от локальных переменных.
4. Модульность. Модульность — это когда различные системы имеют свои собственные задачи, но они группируются вместе, чтобы сначала выполнить более общую задачу. Каждая группа систем выполняет свои задачи последовательно, 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 представляет собой фрагмент кода, который имеет определенное функциональное значение и обладает логической завершенностью. Модульное программирование, в свою очередь, представляет собой подход к созданию программ путем объединения модулей в единую структуру

Основные концепции модульного программирования включают:

1. Каждый модуль имеет только одну точку входа и выхода, что обеспечивает ясность и предсказуемость взаимодействия с другими модулями.
2. Размер модуля должен быть минимизирован насколько это возможно, что способствует повышению читаемости и облегчает понимание функциональности модуля.
3. Вся система программы строится из модулей, которые выполняют определенные функции, обеспечивая модульность и разделение ответственности.
4. Каждый модуль должен быть независим от реализации других модулей, что позволяет изменять и модифицировать модули независимо друг от друга.

Использование модульного программирования упрощает тестирование программы и обнаружение ошибок. Кроме того, аппаратно-зависимые подзадачи могут быть строго отделены от остальных задач, что повышает переносимость создаваемых программ.

1.4 Ссылки

Ссылка в программировании представляет собой псевдоним для другой переменной. Они объявляются с использованием символа "&". Ссылки должны быть проинициализированы при объявлении и инициализируются только один раз.

При объявлении ссылка сразу же инициализируется. Инициализация ссылки выполняется следующим образом:

int i = 0;

int& iref = i;

Физически ссылка (iref) представляет собой константный указатель на int - переменную типа int\* const.

Поскольку ссылка является псевдонимом, при передаче объекта в функцию по ссылке его можно изменять внутри этой функции. Ссылки не могут ссылаться на другие ссылки или на поле битов. Массивы ссылок или указателей на ссылки также не допускаются. Ссылка может быть использована для возврата результата из функции. Возврат значения по ссылке означает возврат самого объекта, а не указателя на объект или его значения.

1.5 Работа с памятью. Указатели

Указатели являются одним из ключевых понятий языка Си. Они позволяют хранить адреса памяти, чаще всего адреса начального элемента динамического массива.

Важно учитывать следующее:

1. Указатель — это переменная, которая содержит адрес другой переменной.
2. При объявлении указателя необходимо указать тип переменных, на которые он будет указывать, и поставить знак "\*" перед именем.
3. Знак "&" перед именем переменной обозначает ее адрес.
4. Знак "\*" перед указателем в рабочей части программы (не в объявлении) обозначает значение ячейки памяти, на которую указывает указатель.
5. Нельзя записывать значения по указателю, который указывает на неопределенную область памяти, так как это может вызвать сбой программы.
6. Для обозначения недействительного указателя используется константа NULL.
7. При изменении значения указателя на "n", он фактически сдвигается на "n" следующих элементов данного типа. Например, для указателей на целые числа это будет сдвиг на "n\*sizeof(int)" байт.
8. Указатели обычно выводятся с использованием формата "%p".

1.6 Работа с файлами

Файл — это именованная область памяти, в которой хранятся данные одного типа. Файл обладает следующими характеристиками: уникальное имя, однородность данных и произвольную длину, ограниченную только размером диска. В языке C++ для работы с файлами необходима ссылка на файл. Для этой цели используется структура FILE, определенная в заголовочном файле stdio.h. Эта структура содержит все необходимые поля для управления файлами, такие как указатель на текущую позицию буфера, счетчик байтов, базовый адрес буфера ввода-вывода и номер файла.

Функция открытия файла.

При открытии файла (потока) в программе возвращается указатель на поток (файловый указатель), который является указателем на объект структуры FILE. Этот указатель идентифицирует поток для всех последующих операций.

Функция закрытия файла.

Рекомендуется явно закрывать открытые файлы на диске после завершения работы с ними. Это считается хорошим стилем программирования.

Функция переименования файла.

Функция позволяет переименовать файл, принимая в качестве параметров старое и новое имя файла. Она возвращает 0 в случае неудачного выполнения.

Функция контроля конца файла.

Для проверки достижения конца файла существует функция

feof.int feof(FILE \* filename);

1.7 Динамические структуры данных

В таких случаях применяются динамические структуры данных, такие как связанные списки или деревья. Эти структуры позволяют эффективно управлять данными переменного размера и структуры, связывая их с помощью ссылок.

Например, для анализа текста и подсчета количества встречающихся слов, можно использовать связанный список, где каждый элемент списка представляет отдельное слово, а связи между элементами образуют связь в порядке алфавитного упорядочивания. При обработке текста, новые слова могут быть добавлены в список и вставлены на правильное место с учетом их лексикографического порядка.

Таким образом, использование динамических структур данных позволяет эффективно работать с данными переменного размера и структуры, обеспечивая связь между элементами через ссылки.

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

Функции играют важную роль в программировании, поскольку они определяют действия, которые программа выполняет. Они позволяют группировать инструкции и дать им имена, чтобы их можно было вызывать многократно из разных частей программы. Функции можно вызывать из любого места в программе, и для этого передаются значения в виде аргументов, которые должны соответствовать типам параметров, указанным в определении функции.

Длина функции практически не ограничена, но для достижения максимальной эффективности кода рекомендуется использовать функции, которые выполняют отдельные, четко определенные задачи. Если это возможно, сложные алгоритмы следует разбивать на более короткие и понятные функции. Также функции могут быть перегружены, что означает, что разные версии функции могут иметь одно и то же имя, но отличаться числом и типом параметров.

Определение функции состоит из объявления и тела функции, которые заключены в фигурные скобки. В теле функции содержатся объявления переменных, операторы и выражения, необходимые для выполнения задачи функции.

1.10 Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) представляет собой подход, при котором программа организуется вокруг взаимодействующих объектов. ООП помогает структурировать код, улучшить его читабельность и облегчить понимание логики программы. Этот подход также способствует повышению безопасности кода и уменьшению его дублирования.

Преимущества ООП проявляются при построении сложных систем, так как предметная область разбивается на объекты, и каждый объект имеет слабую связанность с другими. Это достигается путем соблюдения трех основных принципов: инкапсуляции, наследования и полиморфизма.

Инкапсуляция: объект скрывает свое внутреннее поведение от других объектов. Например, объект "водитель" не требует знания о том, как работает объект "машина" для осуществления своих действий. Это ключевой принцип ООП.

Наследование: объекты могут наследовать свойства и поведение других объектов. Например, объекты "человек" и "водитель" могут иметь общие атрибуты. Наследование позволяет выделить общие аспекты в более общий объект (например, "человек"), а затем определить "водителя" как специализированный объект с дополнительными свойствами и/или поведением.

Полиморфизм: объекты могут переопределять свое поведение. Например, объекты "человек" и "водитель" могут иметь схожее поведение, но реализованное по-разному: один перемещается пешком, а другой - на машине.

ООП упрощает сложные системы, разбивая их на более маленькие и простые объекты. Такой подход позволяет нескольким разработчикам работать над программой, каждый в своем блоке. Большинство современных языков программирования поддерживают ООП, и понимание основных принципов ООП позволяет легко освоить несколько языков программирования.

1.11 Конструкторы и деструкторы

Конструктор в объектно-ориентированном программировании является функцией-членом класса, которая имеет то же имя, что и сам класс, и служит для инициализации переменных-членов и выделения памяти для их хранения.

Деструктор, также являющийся функцией-членом класса, имеет имя класса и предназначен для уничтожения переменных и освобождения памяти.

В языке C++ создание объектов является одной из основных операций, которую вы будете выполнять в своих программах. Для упрощения процесса инициализации данных объектов используется конструктор, который вызывается при создании каждого объекта. Деструктор, в свою очередь, вызывается при уничтожении объекта.

Конструктор представляет собой метод класса, который облегчает инициализацию полей объекта при его создании. Он имеет то же имя, что и класс, и не имеет возвращаемого значения. Конструкторы являются частью интерфейса класса, позволяя создавать объекты данного класса из внешней части программы.

Деструктор, в отличие от конструктора, не может быть перегружен и должен существовать только в одном экземпляре в классе. Он вызывается автоматически при уничтожении объекта и выполняет необходимые операции для освобождения ресурсов, выделенных объектом.

1.12 Классы и дружественные функции

Класс в C++ представляет собой составной тип данных, который может содержать в себе другие типы. Классы и объекты являются основными понятиями в объектно-ориентированном программировании (ООП).

Класс предназначен для описания определенного типа объектов. Он является абстрактным планом или шаблоном объекта. Сам объект представляет конкретное воплощение класса, его экземпляр. Объекты имеют свои свойства (поля) и методы (функции). Свойства класса представляют данные, которыми можно характеризовать объект, а методы выполняют операции и действия над данными.

Поля класса определяются программистом при создании класса и используются для хранения состояния объекта. Все данные, относящиеся к объекту, хранятся в его полях. Класс также может содержать переменные и константы для хранения состояния объекта, а также функции для определения его поведения.

В C++ можно создавать указатели на объекты классов, так же, как и на объекты других типов. Через указатель можно обращаться к членам класса - его переменным и методам. Однако для доступа к членам класса через указатель используется оператор "->", в отличие от обращения через обычную переменную с использованием оператора "точка".

Дружественные функции — это функции, которые не являются членами класса, но имеют доступ к его закрытым членам, таким как приватные переменные и функции. Они объявляются с использованием ключевого слова "friend". Дружественные функции позволяют расширить доступ к закрытым данным класса для определенных функций, сохраняя принципы инкапсуляции и защиты данных.

1.12.1 Описание классов и их методов

В языке программирования C++, классы представляют собой абстракцию, описывающую методы и свойства будущих объектов. Объекты, в свою очередь, являются конкретным воплощением класса и имеют свои собственные свойства и методы. Свойства объекта представляют любые данные, которые могут описывать его характеристики. Методы, с другой стороны, являются функциями, выполняющими различные операции над данными (свойствами) класса.

В объектно-ориентированном программировании поле класса представляет собой переменную, которую программист определяет при создании класса. Поля класса служат для хранения данных объекта. Все данные, относящиеся к объекту, хранятся в его полях.

Классы в C++ позволяют создавать абстрактные модели объектов, определять их свойства и методы, а затем создавать конкретные экземпляры объектов на основе этих классов. Это позволяет структурировать код, повысить его читаемость и повторное использование, а также обеспечивает принципы инкапсуляции и абстракции, характерные для объектно-ориентированного программирования.

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

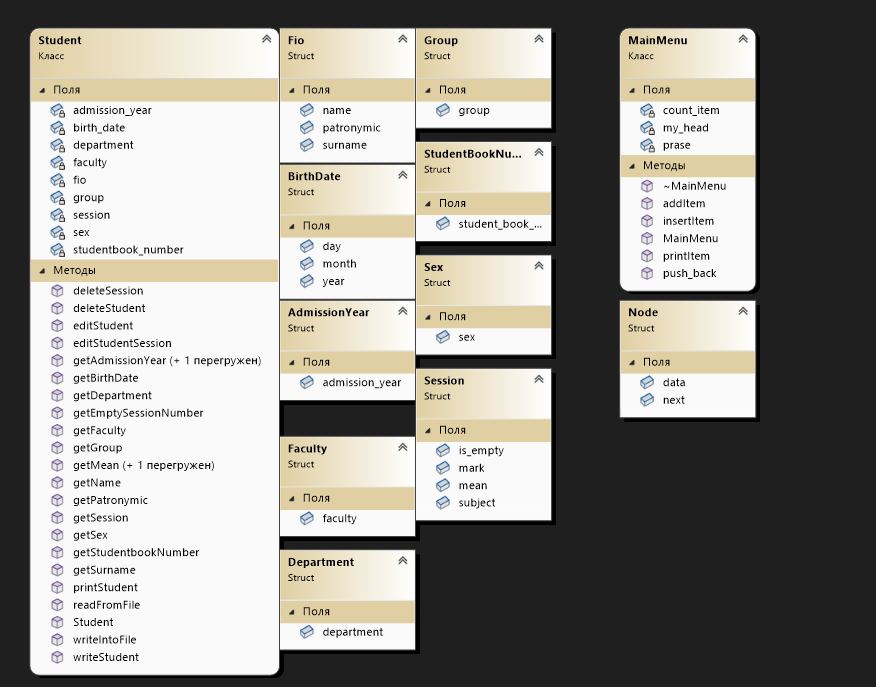


рисунок 1.12.1(1) – диаграмма классов

1.13 Методы шифрования и дешифрования

Шифрование – это процесс кодирования информации с целью предотвращения несанкционированного доступа.

Дешифрование – процесс, обратный процессу шифрованию.

OpenSSL – криптографическая библиотека с открытым исходным кодом. Библиотека поддерживает почти все низкоуровневые алгоритмы хеширования и шифрования, а также реализует большинство популярных криптографических стандартов, в том числе: позволяет создавать ключи RSA, DH, DSA, шифровать данные.

Для начала реализуем функцию Crypt, для этого поэтапно реализуем следующее:

1) генерацию случайного пароля;

2) шифрование алгоритмом AES базы данных;

3) удаление не зашифрованной базы данных;

4) запись пароля в файл;

5) шифрование алгоритмом RSA файла с паролем с помощью публичного ключа;

6) удаление не зашифрованного файла с паролем.

Листинг 1.13.1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 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;  case 2:  pass[i] = rand() % 26 + 'a';  }  } |

Листинг 1.13.2

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

Листинг 1.13.3

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

Листинг 1.13.4

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

Листинг 1.13.5

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

Листинг 1.13.6

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

Для реализации функции Decrypt поэтапно реализуем следующее:

1) Расшифровка зашифрованного файла с паролем с помощью приватного ключа и RSA;

2) Удаление зашифрованного файла с ключом;

3) Считывание ключа из файла;

4) Удаление файла с ключом;

5) Расшифровка зашифрованной базы данных с помощью ключа и алгоритма AES;

6) Удаление зашифрованной базы данных.

Листинг 1.13.7

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

Листинг 1.13.8

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

Листинг 1.13.9

|  |  |
| --- | --- |
| 1  2  3  4  5 | char\* pass = new char[64];  ifstream file;  file.open("key.txt", ios::binary);  file.read(pass, 65);  file.close(); |

Листинг 1.13.10

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

Листинг 1.13.11

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

Листинг 1.13.12

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

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 |
| Ф.И.О. студента | string (строковый) |
| Число, месяц, год рождения | int (целочисленный) |
| Год поступления в институт | int (целочисленный) |
| Институт | string (строковый) |
| Кафедра | string (строковый) |
| Группа | string (строковый) |
| Номер зачетной книжки | string (строковый) |
| Название предмета | string (строковый) |
| Оценка | int (целочисленный) |
| Пустая ли запись | bool (логический) |
| Средний балл сессии | int (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 |
| Student | Содержит в себе данные студента, а именно о его фамилии, имени, отчестве, дате рождения, годе поступления, факультете, кафедре, номере группы, номере зачётной книжки, поле, а также результатах сессий |
| MainMenu | Класс реализации динамического списка, содержит в себе указатель на структуру Node, строковое значение элемента, а также количество этих элементов |

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 |
| Student | int | birth\_date.day – день,  birth\_date.mon – месяц  birth\_date.year – год  admission\_year.admission\_year – год поступления  session.mark – статический двумерный массив оценок |
| string | fio.surname – фамилия  fio.name – имя  fio.patronymic – отчество  faculty.faculty – факультет  department.department – кафедра  group.group – номер группы  studentbook\_number.student\_book\_number – номер зачётной книжки  sex.sex – пол  session.subject – название предмета сессии |
| float | session.mean – средний балл за сессию |
| bool | session.is\_empty – статический двумерный массив, обозначающий, заполнена ли запись о сессии |
| MainMenu | struct \*Node | my\_head – указатель на следующий элемента динамического списка |
| string | phrase – значение (что будем выводить) |
| int | count\_item – количество элементов в динамическом списке |

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 |
| Student | Student | Конструктор по умолчанию | - | - |
| setSurname | Установить значение фамилии | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Student | setName | Установить значение имени | - | - |
| setPatronymic | Установить значение отчества | - | - |
| setBirthDate | Установить значение даты рождения | - | - |
| setAdmissionYear | Установить значение года поступления | - | - |
| setFaculty | Установить значение факультета | - | - |
| setDepartment | Установить значение кафедры | - | - |
| setGroup | Установить значение номера группы | - | - |
| setStudentbookNumber | Установить значение номера зачётной книжки | - | - |
| setSex | Установить значение пола | - | - |
| setSession | Установить значение записи сессии | - | - |
| setMean | Установить значение среднего балла сессии | - | - |
| getAdmissionYear | Возвращает значение года поступления | - | int |
| getMean | Возвращает значение среднего балла сессии | - | float |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Student | getEmptySessionNumber | Возвращает количество записей в сессии студента | session\_num  (int) | int |
| addStudent | Добавляет запись о студенте в динамический массив | - | - |
| printCompact | Выводит компактную информацию о студенте | student\_num  (int) | - |
| printStudent | Выводит информацию о студенте в виде таблицы | student\_num  (int) | - |
| writeIntoFile | Записывает переданные данные в файл | fio\_, birth\_date\_, admission\_year, faculty\_, department\_, group\_, studentbook\_number\_, sex\_, session\_  (Fio (string), BirthDate (int), AdmissionYear (int), Faculty (string), Department (string), Group (string), StudentBookNumber (string), Sex (string), Session (string, int, bool, float)) | - |
| readFromFile | Читает данные из файла | required\_number  (int) | - |
| editStudent | Изменяет данные о студенте | \*student, student\_count  (Student, int) | - |
| editStudentSession | Изменяет данные о сессии студента | required\_student  (int) | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| MainMenu | deleteStudent | Удаляет запись о студенте | \*student, student\_count  (Student, int) | int |
| deleteSession | Удаляет запись сессии студента | required\_student  (int) | - |
| MainMenu | Конструктор с параметром | phrase  (string) | - |
| ~MainMenu | Деструктор по умолчанию | - | - |
| addItem | Добавляет элемент в начало динамического списка | inp\_data  (string) | - |
| insertItem | Добавляет элемент в конец динамического списка | index, inp\_data  (int, string) | - |
| push\_back | Добавляет элемент в конец динамического списка | inp\_data  (string) | - |
| printItem | Вывод элементов | - | - |
| Основная программа | main | Точка входа в программу. В ней происходит обращение к функциям в зависимости от выбора пользователя в главном меню | - | int |
| getAmountOfStudents | Возвращает количество студентов в базе данных | - | int |
| initStudent | Инициализирует данные о студентах из базы данных | \*student, student\_count  (Student, int) | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Основная программа | mainMenu | Устанавливает значения пунктов главного меню и возвращает действие, которое в дальнейшем будет выполнено программой | - | int |
| checkForValue | Проверка, попадает ли число в указанный интервал | min, value, max  (int, int, int) | bool |
| checkForValue | Проверка, строки на длину | value, max  (string, int) | bool |
| getAlpha | «Умный» ввод текста (невозможность ввода цифр и других символов) | what\_to\_enter  (string) | string |
| getDigit | «Умный» ввод числа (невозможность ввода букв и других символов) | what\_to\_enter  (string) | int |
| bubbleSort | Осуществляет пузырьковую сортировку элементов массива | \*student, \*&arr, len  (Student, int, int) | - |
| task | Осуществляет выполнение задания согласно варианту | \*student  (Student) | - |
| Crypt | Осуществляет шифрование файла базы данных с использованием OpenSSL | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Основная программа | Decrypt | Осуществляет дешифрование файла базы данных с использованием OpenSSL | - | - |

2.2 Алгоритм решения задачи

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.

Описание алгоритма сортировки варианта:

Программа заходит в бесконечный цикл while, в котором, с помощью функции, которая не даёт вводить буквы, запрашивает у пользователя год поступления, по которому будет осуществляться сортировка. Далее она проверяет, чтобы год поступления находился в интервале от 1900 до 2022. Если введённый год находится в указанном интервале, программа выходит из бесконечного цикла, в противном случае цикл продолжается до тех пор, пока пользователь не введёт подходящий год поступления. Далее инициализируются 3 переменные: s\_len = 0, us\_len = 0 и student\_count, которая равна количеству студентов в базе данных. Для этого программа расшифровывает файл и считает количество строк, которыми обозначается конец записи о студенте в базе данных, после чего программа обратно шифрует файл и присваивает значение переменной student\_count. Далее создаётся 2 динамических массива целых чисел с помощью оператора new: suitable и unsuitable, оба размерностью ноль. После создания динамических массивов запускается цикл for с переменной i, изменяющейся от нуля до значения переменной student\_count и шагом +1 за 1 итерацию. В цикле for находится условие (1) «если год поступления i-того студента равен введённому году поступления», то создаётся новый динамический массив new\_suitable, размерностью на единицу больше, значения переменной s\_len. Далее запускается цикл for, в котором i-тому элементу массива new\_suitable присваивается значение i-того элемента массива suitable. После окончания работы цикла for, последнему элементу массива new\_suitable присваивается индекс, далее с помощью оператора delete происходит удаление элементов массива suitable, после этого указателю на массив suitable присваивается указатель на массив new\_suitable. Программа увеличивает значение переменной s\_len, которая отвечает за размерность динамического массива. Если условие (1) не выполняется, то проделывается то же самое, но создаётся и заполняется динамический массив new\_unsuitable. Таким образом, после окончания цикла for, в программе есть 2 динамических массива, в одном из которых содержатся индексы студентов, поступивших в введённый год, а в другом массиве – студентов, поступивших в другие годы. После того, как массивы заполнены, вызывается функция «bubbleSort», в которой происходит последовательное сравнение среднего балла по всем сессиям 2-ух студентов. Если средний балл i-того студента меньше среднего балла i+1-того студента, то программа меняет местами i и i+1-ых студентов в массиве. После сортировки пузырьком, программа «проходит» по индексам сначала массива suitable, а потом и по индексам массива unsuitable и выводит студента по индексу. Таким образом, в консоли выводятся студенты, отсортированные по среднему баллу, поступившие в указанном году.

Ниже представлена схема работы алгоритма решения задачи.

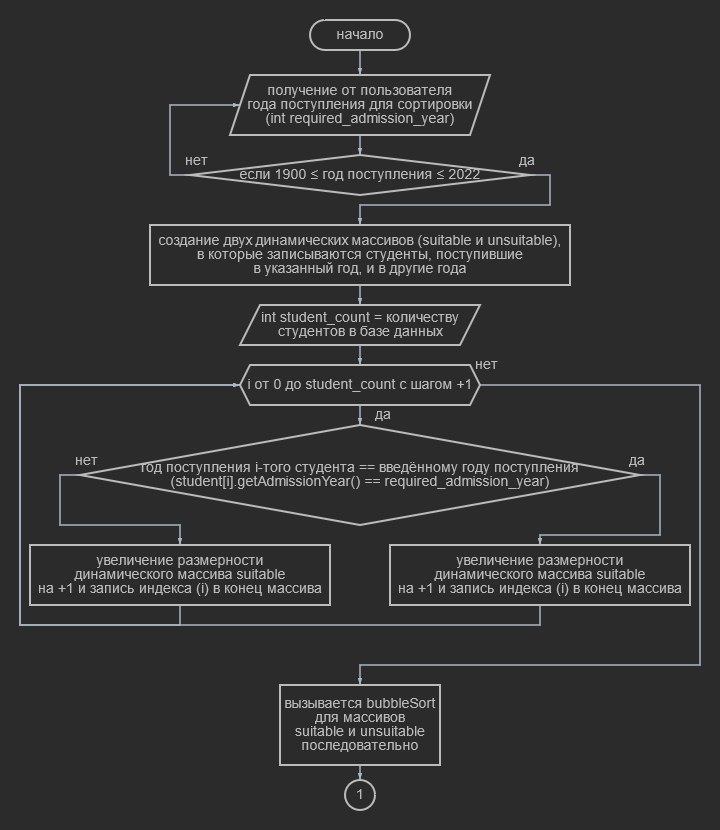


рисунок 2.2(1) – схема алгоритма выполнения индивидуального задания, часть 1

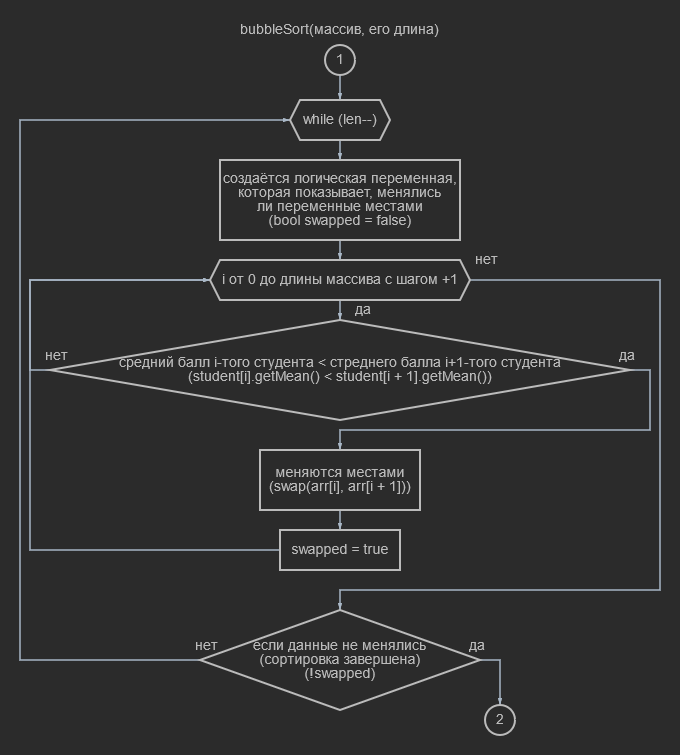


рисунок 2.2(2) – схема алгоритма выполнения индивидуального задания, часть 2

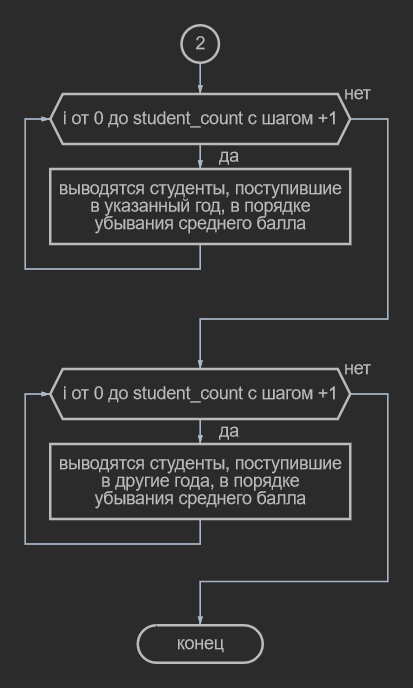


рисунок 2.2(3) – схема алгоритма выполнения индивидуального задания, часть 3

2.3 Программная реализация задания

2.3.1 Пример выполнения задания

СКРИН

рисунок 2.1.3(1) – пример выполнения задания

2.3.2 Отображение списка студентов

СКРИН

рисунок 2.3.2(1) – отображение списка студентов

2.3.3 Изменение данных о студенте

СКРИН

рисунок 2.3.3(1) – ввод порядкового номера студента при изменении данных

СКРИН

рисунок 2.3.3(2) – выбор параметра студента при изменении данных

СКРИН

рисунок 2.3.3(3) – ввод новых данных студента при их изменении

СКРИН

рисунок 2.3.3(4) – окончание работы функции изменения данных

2.3.4 Добавление информации о новом студенте

СКРИН

рисунок 2.3.4(1) – ввод фамилии при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(2) – ввод имени при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(3) – ввод отчества при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(4) – ввод даты рождения при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(5) – ввод месяца рождения при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(6) – ввод года рождения при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(7) – ввод года поступления при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(8) – ввод факультета при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(9) – ввод кафедры при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(10) – ввод номера группы при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(11) – ввод номера зачётной книжки при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(12) – ввод пола при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(13) – ввод номера сессии при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(14) – ввод названия предмета при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(15) – ввод оценки при добавлении информации о новом студенте

СКРИН

рисунок 2.3.4(16) – конец добавления информации о новом студенте

2.3.5 Удаление данных о студенте

СКРИН

рисунок 2.3.5(1) – ввод порядкового номера при удалении информации о студенте

СКРИН

рисунок 2.3.5(2) – окончание работы функции удаления студента

2.3.6 Шифрование данных

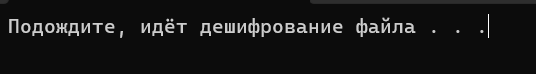


рисунок 2.3.6(1) – отображение процесса дешифрования в консоли



рисунок 2.3.6(2) – файл базы данных, зашифрованный с помощью OpenSSL



рисунок 2.3.6(2) – файл с ключом от базы данных, зашифрованный с помощью OpenSSL

2.4 Руководство пользователя

При запуске программы пользователю выводится номер пункта и его название (функционал). Среди пунктов есть: «Добавить запись», «Вывод данных», «Изменение данных», «Удалить запись», «Выполнить 24 вариант», «Выход». Пользователю предлагается ввести порядковый номер меню для осуществления предписанных ей действий.

При вводе пользователем «1» и нажатии клавиши «Enter», реализовывается функционал добавления записи о студенте. Пользователю предлагается пошагово вводить данные с клавиатуры.

Сначала необходимо ввести фамилию, затем имя и отчество студента. При случайном нажатии пользователем клавиш цифрового блока или спецсимволов, программа не считывает их. Это демонстрирует защиту от случайных нажатий. Так же, длина и фамилии, и имени, и отчества не должна быть равной нулю или превышать 20 символов.

После ввода ФИО студента, пользователю необходимо ввести последовательно сначала дату, а затем месяц и год рождения студента. Эти поля также защищены от случайных нажатий, т. е. пользователь может вводить только цифры, используя цифровой блок клавиатуры. Помимо защиты от случайных нажатий, есть и проверка на ввод «разумной» даты рождения (не менее 1900, не более 2005).

Далее пользователю предлагается ввести год поступления. На этом этапе есть проверка на ввод только цифр, а также проверка на то, находится ли указанный год в интервале от года рождения плюс 13 лет до 2022 года.

Далее пользователю предлагается ввести факультет. На этом этапе есть 2 проверки: ввод только букв и длину, которая не должна превышать 60 символов.

Далее пользователю предлагается ввести название кафедры. На данном этапе есть лишь проверка на длину, которая не должна превышать 60 символов.

Далее пользователю предлагается ввести номер группы. На данном этапе есть лишь проверка на длину, которая не должна превышать 60 символов.

Далее пользователю предлагается ввести номер зачётной книжки. На данном этапе есть лишь проверка на длину, которая не должна превышать 60 символов.

Далее пользователю предлагается ввести пол студента. Ввод «1» приравнивается к мужскому полу, «0» – к женскому. На данном этапе есть проверка на ввод пользователем только цифр и на соответствие ввода «1» или «0».

Далее пользователю предлагается ввести номер сессии, данные о которой он хочет добавить. Далее пользователю предлагается ввести однословное или многословное название предмета (есть проверка на то, есть ли ещё место для записи), а затем ввести соответствующую ему оценку (есть проверка на ввод только цифр и попадет ли введённая цифра в интервал от 0 до 5). Для ввода дифференцированной оценки, необходимо просто ввести её, а для того, чтобы поставить «зачёт» необходимо ввести «1», для «незачёт» – «0». Программа будет спрашивать пользователя номер сессии до тех пор, пока пользователь не введёт «0». Тогда программа сохранит изменения в базе данных и откроет главное меню.

При вводе пользователем «2» на экран выводится информация о студентах в порядке их добавления. Вывод оформлен в виде таблицы с отображением результатов сессии. При нажатии любой клавиши, программа открывает главное меню.

При вводе пользователем «3», пользователю выводится краткий список студентов, содержащихся в базе данных. Пользователю предлагается ввести номер студента, данные которого он бы хотел изменить (есть проверка на ввод только цифр и попадает ли введённое число в интервал от 1 до количества студентов). После ввода номера студента, пользователю предлагается список данные, которые можно изменить. После ввода номера изменяемого параметра, пользователю предлагается изменить этот самый параметр. Этот процесс идентичен процессу добавления студента (сохраняются проверки на ввод), но, в отличие от него, при изменении данных, пользователь вводит только выбранный параметр. После изменения параметра, программа записывает изменения в файл и открывает главное меню.

При вводе пользователем «4», пользователю выводится краткий список студентов, содержащихся в базе данных. Пользователю предлагается ввести номер студента, данные которого он бы хотел удалить (есть проверка на ввод только цифр и попадает ли введённое число в интервал от 1 до количества студентов). После ввода порядкового номера, студент с таким номером удаляется из базы данных, а программа открывает главное меню.

При вводе «5», программа предлагает пользователю ввести год поступления (есть проверка на ввод только цифр и попадает ли введённое число в интервал от 1913 до 2022), по которому будет осуществляться выполнение варианта. После ввода года поступления, по нему будет осуществлена сортировка. Сначала появятся данные о студентах, поступивших в указанный год, а затем, поступившие в другие года. Оба списка отсортированы по убыванию среднего балла по всем предметам. После нажатия любой клавиши, программа отрывает главное меню.

При вводе «6», программа завершает свою работу.

2.5 Системные требования

Язык программирования: С++.

Операционная система: Linux Mint 20.1 выше или аналог, Windows 10.

ОЗУ: 1 Гб и более.

Свободное место на диске: 200 Мб и более.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была написана программа, в которой было реализовано задание: 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.

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.
12. Курсовая работа medwu [Электронный ресурс]. - URL: https://github.com/medwuu/course-work (Дата обращения: 19.5.2023).

ПРИЛОЖЕНИЕ А Исходный код программы

**КОД**