СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc134171150)

[АННОТАЦИЯ 4](#_Toc134171151)

[ВВЕДЕНИЕ 5](#_Toc134171152)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc134171153)

[1.1 Структурное программирование 7](#_Toc134171154)

[1.2 Процедурное программирование 7](#_Toc134171155)

[1.3 Модульное программирование 8](#_Toc134171156)

[1.4 Ссылки 9](#_Toc134171157)

[1.5 Работа с памятью. Указатели 9](#_Toc134171158)

[1.6 Работа с файлами 10](#_Toc134171159)

[1.7 Динамические структуры данных 11](#_Toc134171160)

[1.8 Линейный список 11](#_Toc134171161)

[1.9 Функции 11](#_Toc134171162)

[1.10 Объектно-ориентированное программирование 12](#_Toc134171163)

[1.11 Конструкторы и деструкторы 13](#_Toc134171164)

[1.12 Классы и дружественные функции 14](#_Toc134171165)

[1.12.1 Описание классов и их методов 15](#_Toc134171166)

[1.13 Методы шифрования и дешифрования 15](#_Toc134171167)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 19](#_Toc134171168)

[2.1 Структура программы 19](#_Toc134171169)

[2.2 Алгоритм решения задачи 23](#_Toc134171170)

[2.3 Программная реализация задания 23](#_Toc134171171)

[2.3.1 Пример выполнения задания 23](#_Toc134171172)

[2.3.2 Отображение списка студентов 23](#_Toc134171173)

[2.3.3 Изменение данных о студенте 23](#_Toc134171174)

[2.3.3 Добавление информации о новом студенте 23](#_Toc134171175)

[2.3.4 Удаление данных о студенте 23](#_Toc134171176)

[2.3.5 Шифрование данных \* 23](#_Toc134171177)

[2.4 Руководство пользователя 23](#_Toc134171178)

[2.8 Системные требования 24](#_Toc134171179)

[ЗАКЛЮЧЕНИЕ 25](#_Toc134171180)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 26](#_Toc134171181)

[ПРИЛОЖЕНИЕ А Исходный код программы 27](#_Toc134171182)

АННОТАЦИЯ

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

Выполнил: Русаков Вячеслав Андреевич, БИСО-01-22

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: <ТУТ ВСТАВЛЯЕТЕ ЗАДАНИЕ ПО ВАШЕМУ ВАРИАНТУ>

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. Для работы с файлом в языке C++ необходима ссылка на файл. Для определения такой ссылки существует структура FILE, описанная в заголовочном файле stdio.h. Данная структура содержит все необходимые поля для управления файлами, например, текущий указатель буфера, текущий счетчик байтов, базовый адрес буфера ввода-вывода, номер файла.

Функция открытия файла.

При открытии файла (потока) в программу возвращается указатель на поток (файловый указатель), являющийся указателем на объект структурного типа FILE. Этот указатель идентифицирует поток во всех последующих операциях.

Функция закрытия файла.

Открытые на диске файлы после окончания работы с ними рекомендуется закрыть явно. Это является хорошим тоном в программировании.

Функция переименования файла.

Функция переименовывает файл; первый параметр – старое имя файла, второй – новое. Возвращает 0 при неудачном выполнении.

Функция контроля конца файла

Для контроля достижения конца файла есть функция feof. int feof(FILE \* filename);

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 для обащения 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.

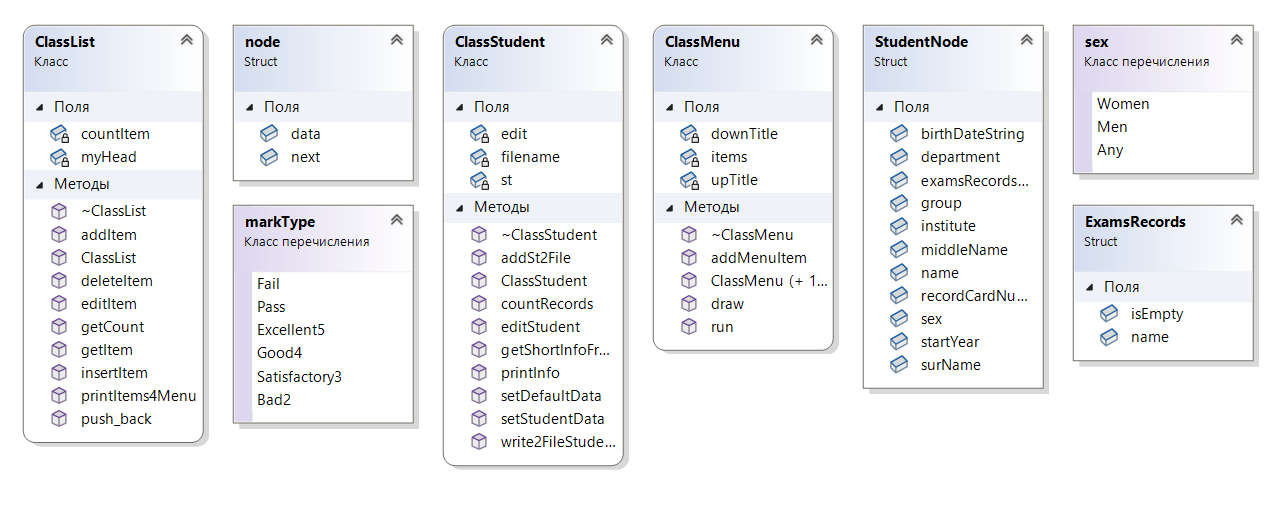
1.12.1 Описание классов и их методов

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>

<НАПРИМЕР>



</НАПРИМЕР>

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 рандомного 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) |
| Фамилия | Символьный массив из 30 элементов |
|  |  |
| Пол студента | int(целочисленный) |
| Число, месяц, год рождения | Date(дата) |
| Год поступления в институт | Date(дата) |
| Институт | Символьный массив из 30 элементов |
| Кафедра | string(строковый) |
| Группа | string(строковый) |
| Номер зачетной книжки | string(строковый) |
| Название предмета | string(строковый) |
| Оценка | 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 |
| Date | Является пользовательским типом данных. |
| Zachetka | Содержит в себе данные студента, а именно о его успеваемости за пройденные семестры. |
| Person | Хранит персональные данные студента (ФИО, дата рождения, пол) |
| Semestr | Содержит в себе информацию о дисциплинах семестра |
| Univer | Хранит в себе полную информацию о всех студентах |
| Subj | Содержит в себе оценке и название предмета |
| StudentArray | Хранит список студентов |
| Student | Содержит данные класса Person и сведения о дате зачисления, группе, факультете и кафедре |

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 |
| Date | int | day – день,  mon – месяц  year – год |
| Person | string | \_sname – Фамилия  \_fname – Имя  \_mname – Отчество |
| Date | \_bdate – Дата рождения |
| int | \_gender – Пол: 0=Ж, 1=М |
| Semestr | int | sc – количество имеющихся предметов |
| Subj\* | \_subjects[10] – массив указателей на 10 возможных предметов в семестре |
| Student | Date | \_edate – дата поступления |
| string | \_group – группа  \_fac – факультет  \_kaf – кафедра |
| Zachetka\* | \_zach – указатель на объект зачётной книжки |
| StudentArray | int | length – длина списка |
| Subj | string | title – название предмета |
| int | mark – оценка |
| Univer | string | \_title – название университета  \_students – указатель на список студентов |
| StudentArray\* |
| int | N – текущее кол-во студентов |
| Zachetka | string | \_num – номер зачётки |
| Semestr\* | \_sems[12] – массив указателей на 12 семестров |
| int | sc – кол-во имеющихся семестров |

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 |
| Person | fullIntroduce | Вывод информации о человеке | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Date | isValid | Проверка даты на корректность ввода (день, месяц, год) | day, mon, year (int) | int |
| isValid | Проверка даты на корректность ввода (строка формата) | sdate (string) | int |
| isValid | Проверка даты на корректность | Date | int |
| Semestr | hasSubj | Проверка на наличие предмета в семестре по названию | title (string) | int |
| getByTitle | Получение указателя на предмет по его названию | title (string) | Subj\* |
| addSubj | Добавить предмет в семестр | Subj\* | int |
| average | Считает среднее арифметическое оценок за семестр | - | float |
| Student | fullIntroduce | Вывести всю информацию о студенте | - | - |
| fullAvg | Среднее арифметическое всех оценок студента | - | float |
| dataForExport | Строка для экспорта данных студента в файл | - | string |
| StudentArray | addItem | Добавить указатель на студента в список | Student\* | - |
| removeItem | Удалить студента из списка | zach (string)  (номер зачётки) | - |
| Univer | addStudent | Добавление указателя на студента  в университет | Student\* | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Univer | removeStudent | Удаление студента | zach (string)  (номер зачётки) | - | |
| Zachetka | showAll | Вывести в окно консоли все семестры и предметы с оценками | - | - |
| addSem | Добавить указатель на семестр в зачётку | Semestr\* | int |
| average | Средняя оценка по всей зачётке | - | float |
| dump | Получить данные для экспорта в файл | - | string |
|  | changeStudent | Изменить информацию студента ро номеру зачётки | zach\_num, field (string, int) | - |
| showStudents | Вывод информации о каждом студенте университета | - | - |
| showStudentChangeDialog | Показать диалог изменения данных студента | zn (string)  (номер зачётки) | - |
| getByZach | Получение указателя на студента по номеру зачётки | zach\_num  (номер зачётки) | Student\* |
| hasStudent | Проверить наличие студента в университете | Student\* | int |
| hasStudent | Проверить наличие студента в университете по номеру зачётки | zach\_num (string)  (номер зачётки) | int |
| Var96query | Выполнить выборку, согласно условиям 96 варианта | se, ee, year  (int, int, int) | - |
| exportAll | Генерация строки для дальнейшего экспорта в файл | - | string |
| saveAll | Сохранить базу данных в файл | - | - |
| loadAll | Загрузить базу данных из файла |  | - |

2.2 Алгоритм решения задачи

<ВАШ ВАРИАНТ ЗАДАНИЯ КУРСОВОЙ РАБОТЫ>

<ПОЭТАПНОЕ ОПИСАНИЕ АЛГОРИТМА ВЫБОРКИ/СОРТИРОВКИ ДЛЯ ВЫПОЛНЕНИЯ ВАРИАНТА>

<ТУТ БЛОК СХЕМА ПРОГРАММЫ>

2.3 Программная реализация задания

2.3.1 Пример выполнения задания

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.2 Отображение списка студентов

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.3 Изменение данных о студенте

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.3 Добавление информации о новом студенте

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.4 Удаление данных о студенте

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

2.3.5 Шифрование данных \*

<СКРИНШОТ/ ПРИМЕР по данному пункту с описанием>

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.

2.5 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, <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. Языки программирования [Электронный ресурс]: методические рекомендации по выполнению курсовой работы / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2022. — Электрон. опт. диск (ISO)
2. 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)
3. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2022.
4. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2022.
5. Стенли Липпман Язык программирования С++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2023. 1104 c.
6. Страуструп Б. Дизайн и эволюция языка С++. – Litres, 2022.
7. Уильямс Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. – Litres, 2022.
8. Аммерааль Л. STL для программистов на C++. – Litres, 2022.
9. Дейл Н., Уимз Ч., Хедингтон М. Программирование на C++. – Litres, 2022.
10. Нефедов Д. Г., Русяк И. Г., Вавилова Д. Д. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. – 2020.
11. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – " Издательский дом"" Питер""", 2018.
12. 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….**

</НАПРИМЕР>