**

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка абстрактного типа данных «Длинное число»»

Выполнил студент

Мукоедов Максим Андреевич

Группа П2-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

Королёв 2020 г

**Оглавление**

[Задание на выполнение курсового проекта 4](#_Toc44579721)

[Введение 5](#_Toc44579722)

[1. Теоретическая часть 6](#_Toc44579723)

[1.1. Этапы разработки программы 6](#_Toc44579724)

[1.1.1. Язык программирования C++ 8](#_Toc44579725)

[2. Проектная часть 9](#_Toc44579726)

[2.1. Диаграмма прецедентов 9](#_Toc44579727)

[2.2. Выбор инструментария 10](#_Toc44579728)

[2.3. Проектирование сценария 11](#_Toc44579729)

[2.3.1. Достоинства и недостатки языка C++ 12](#_Toc44579730)

[2.4. Основные требования к проекту 13](#_Toc44579731)

[2.5. Выбор инструментария для написания проекта 13](#_Toc44579732)

[2.6. Программа 14](#_Toc44579733)

[2.6.1. Основы 14](#_Toc44579734)

[2.7. Разработка классов и функций для каждого пункта меню 15](#_Toc44579735)

[2.7.1. Подключение файлов к программе 16](#_Toc44579736)

[2.7.2. Разработка панели управления 16](#_Toc44579737)

[3. Эксплуатационная часть 18](#_Toc44579738)

[3.1. Состав функций 18](#_Toc44579739)

[3.1.1. Условия выполнения программы 18](#_Toc44579740)

[3.2. Выполнение программы 19](#_Toc44579741)

[Заключение 21](#_Toc44579743)

[Список используемой литературы и интернет-источников 22](#_Toc44579744)

[Приложение 1. Листинг главного модуля main.cpp 23](#_Toc44579745)

[Приложение 2. Листинг summ.h 26](#_Toc44579746)

[Приложение 3. Листинг razn.h 28](#_Toc44579747)

[Приложение 4. Листинг multiplication.h 30](#_Toc44579748)

[Приложение 5. Листинг division.h 31](#_Toc44579749)

[Приложение 6. Листинг sqrt.h 33](#_Toc44579750)

[Приложение 7. Листинг settings.h 35](#_Toc44579751)

# Задание на выполнение курсового проекта

Разработать программу для работы с абстрактным типом данных «Длинное число» на языке программирования C++.

# Введение

Данный курсовой проект посвящен созданию программы для работы с длинными числами на языке C++.

Цель курсового проекта заключалась в понимании основных принципов работы с многофайловым проектом, работы с классами и файлами.

# Теоретическая часть

# Этапы разработки программы

**Абстрактный Тип Данных (АТД)**

Что означает слово «абстрактный»? В первую очередь понятие «абстрактность» означает сосредоточение внимания на чем-то важном и, при этом, нам нужно отвлечься от неважных, на данный момент, деталей. Определение абстрактности хорошо раскрыто в книге Гради Буча [Grady Booch].

Само определение звучит следующим образом:

**Абстракция** – это выделение и придание совокупности объектов общих свойств, которые определяют их концептуальные границы и отличают от всех других видов объектов.

Иными словами, абстракция позволяет «пролить свет» на нужные нам данные объектов и, при этом «затенить» те данные, которые нам не важны.

Что же будет, если слить понятия «тип данных» и «абстракция» воедино? Мы получим тип данных, который предоставляет нам некий набор операций, обеспечивающих поведение объектов этого типа данных, а также этот тип данных будет скрывать те данные, с помощью которых реализовано данное поведение.

**АТД** – это такой тип данных, который скрывает свою внутреннюю реализацию от клиентов.

**Многофайловый проект (заголовочные файлы)**

В программировании **заголовочный файл** (англ. *header file)* или **подключаемый файл** – файл, содержимое которого автоматически добавляется препроцессором в исходный текст в том месте, где располагается некоторая директива (в Си++ это #include <file.h>).

В языках программирования Си и C++ заголовочные файлы – основной способ подключить к программе типы данных, структуры, прототипы функций, перечисляемые типы и макросы, используемые в другом модуле. По умолчанию используется расширение *.h*; иногда для заголовочных файлов языка C++ используют расширение *.hpp*.

**Классы в C++**

Классы в C++ - это абстракция описывающая методы, свойства, ещё не существующих объектов. Объекты – конкретное представление абстракции, имеющие свои свойства и методы. Созданные объекты на основе одного класса называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса. В ООП существует три основных принципа построения классов:

* + - 1. **Инкапсуляция** – это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя.
      2. **Наследование** – это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку.
      3. **Полиморфизм** – свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Структура объявления класса выглядит следующим образом:

Class /\*имя класса\*/

{

private:

/\*список свойств и методов для использования внутри класса\*/

public:

/\*список методов доступных другим функциям и объектам программы\*/

protected:

/\*список средств, доступных при наследовании\*/

};

**Работа с файлами в C++**

Для работы с файлами необходимо подключить заголовочный файл <fstream>. В <fstream> определены несколько классов и подключены заголовочные файлы <ifstream> - файловый ввод и <ofstream> - файловый вывод.

Например, необходимо создать текстовый файл и записать в него строку «Работа с файлами в C++». Для этого необходимо проделать следующие шаги:

1. Создать объект класса ofstream;
2. Связать объект класса с файлом, в который будет производиться запись;
3. Записать строку в файл;
4. Закрыть файл.

Для записи в файл используется объект класса ofstream, а для чтения информации из файла используется ifstream.

Программа, описанная чуть выше будет выглядеть так:

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

ofstream file\_input(“file.txt”); // создаем объект класса

// ofstream для записи и связываем его с файлом file.txt

file\_input<<”Работа с файлами в C++”; // запись строки в файл

file\_input.close(); // закрываем файл

return 0;

}

Закрытие файла является обязательной процедурой. Если не закрыть файл после всех операций с этим файлом, то могут случиться непредвиденные ошибки с этим самым файлом (например, удаление всей информации из файла, неправильное отображение информации и т.д.).

Также есть режимы открытия файлов:

|  |  |
| --- | --- |
| ios\_base::in | Открыть файл для чтения |
| ios\_base::out | Открыть файл для записи |
| ios\_base::ate | При открытии файла переместить указатель в конец файла |
| ios\_base::app | Открыть файл для записи в конец файла |
| ios\_base::trunc | Удалить всё содержимое файла в том случае, если он существует |
| ios\_base::binary | Открытие файла в двоичном режиме |

Режимы открытия файлов можно комбинировать с помощью поразрядной логической операции **или |**, например: ios\_base::out | ios\_base::trunc – открытие файла для записи, предварительно очистив его.

# Язык программирования C++

**C++** (читается си-плюс-плюс) - компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. **C++** сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником - языком **C**, - наибольшее внимание уделено поддержке объектно-ориентированного и обобщенного программирования.

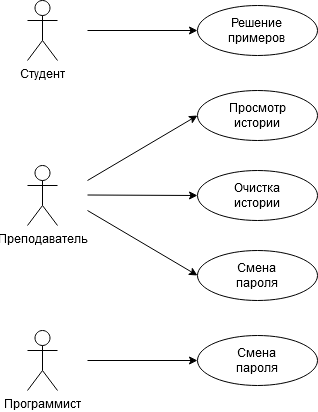
**C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка **C++**, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это **GCC**, **Visual C++**, **Intel C++ Compiler**, **Embarcadero (Borland) C++ Builder** и другие. **C++** оказал огромное влияние на другие языки программирования, в первую очередь на **Java** и **C#**.

Синтаксис **C++** унаследован от языка **C**. Одним из принципов разработки было сохранение совместимости с **C**. Тем не менее, **C++** не является в строгом смысле надмножеством **C**; множество программ, которые могут одинаково успешно транслироваться как компиляторами **C**, так и компиляторами **C++**, довольно велико, но не включает все возможные программы на **C**.

# Проектная часть

# Диаграмма прецедентов

По указанию преподавателя диаграмма прецедентов была спроектирована для калькулятора. Диаграмма прецедентов представлена на рисунке 1.



**Рисунок 1**. Диаграмма прецедентов

# Выбор инструментария

При написании курсового проекта были выбраны следующие программные решения: среда разработки Code::Blocks и язык программирования C++.

Таблица 1. Критерии выбора инструментов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Простота и понимание синтаксиса | Документация на русском | Скорость разработки | Удобство разработки |
|  | Высокое | Среднее | Высокая | Высокое |

Сравнение языков программирования представлено в таблице.

Таблица 2. Оценка языков программирования

|  |  |  |
| --- | --- | --- |
|  | C | C++ |
| Простота и понимание синтаксиса | 7 | 9 |
| Документация на русском | 6 | 8 |
| Скорость разработки | 8 | 10 |
| Итог: | 21 | 27 |

Таблица 3. Оценка сред разработки

|  |  |  |
| --- | --- | --- |
|  | Visual Studio | Code::Blocks |
| Интерфейс | 8 | 8 |
| Скорость разработки | 8 | 7 |
| Документация на русском | 7 | 7 |
| Удобство разработки | 4 | 10 |
| Итог? | 27 | 32 |

Вывод: в совокупности всех факторов, был выбран язык программирования C++ и среда разработки Code::Blocks.

Среда разработки Code::Blocks:

Использовалась для написания кода.

# Проектирование сценария

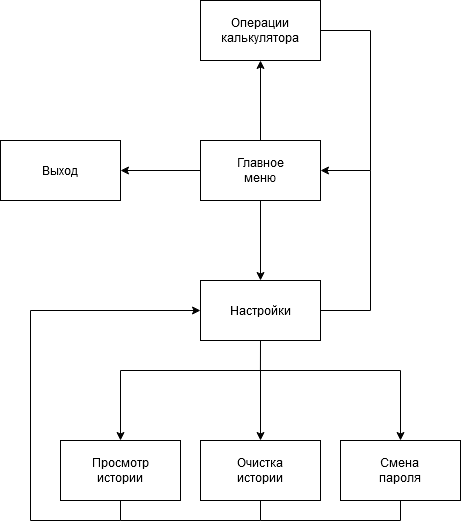
После запуска программы пользователь имеет несколько вариантов действий:

* Решение примеров
* Авторизация в меню настроек

После авторизации:

* Просмотр истории
* Очистка истории
* Смена пароля

Диаграмма сценария представлена на рисунке 2.

****

**Рисунок 2.** Диаграмма сценария

# Достоинства и недостатки языка C++

К многочисленным преимуществам этого языка относятся:

* Высокая совместимость с языком Си: код на Си может быть с минимальными переделками скомпилирован компилятором C++. Внешнеязыковой интерфейс является прозрачным, так что библиотеки на Си могут вызываться из C++ без дополнительных затрат, и более того — при определённых ограничениях код на C++ может экспортироваться внешне не отличимо от кода на Си (конструкция extern "C").
* Как следствие предыдущего пункта — вычислительная производительность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Один из базовых принципов C++ — «не платишь за то, что не используешь» (см. Философия C++) — то есть ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования. Имеется возможность работы с памятью на низком уровне.
* Поддержка различных стилей программирования: традиционное императивное программирование (структурное, объектно-ориентированное), обобщённое программирование, функциональное программирование, порождающее метапрограммирование.
* Автоматический вызов деструкторов объектов в адекватном порядке (обратном вызову конструкторов) упрощает и повышает надёжность управления памятью и другими ресурсами (открытыми файлами, сетевыми соединениями, соединениями с базами данных и т. п.).
* Перегрузка операторов позволяет кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.
* Имеется возможность управления константностью объектов (модификаторы const, mutable, volatile). Использование константных объектов повышает надёжность и служит подсказкой для оптимизации. Перегрузка функций-членов по признаку константности позволяет определять выбор метода в зависимости цели вызова (константный для чтения, неконстантный для изменения). Объявление mutable позволяет сохранять логическую константность при виде извне кода, использующего кэши и ленивые вычисления.
* Шаблоны C++ дают возможность построения обобщённых контейнеров и алгоритмов для разных типов данных. Попутно шаблоны дают возможность производить вычисления на этапе компиляции.
* Возможность встраивания предметно-ориентированных языков программирования в основной код. Такой подход использует, например библиотека Boost.Spirit, позволяющая задавать EBNF-грамматику парсеров прямо в коде C++.
* Доступность. Для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки. Язык имеет высокий порог вхождения, но среди всех языков такого рода обладает наиболее широкими возможностями.

К числу обычно упоминаемых недостатков языка можно отнести:

* Отсутствие системы модулей. C++ унаследовал от Си подключение заголовочных файлов с помощью препроцессора. Это вынуждает дублировать описания объектов, порождает неочевидные требования к коду (см. правило одного определения) и увеличивает объём компилируемого текста, а значит и время компиляции.
* Наличие более чем одного механизма для выполнения одних и тех же задач, что усложняет язык и приводит к неоптимальному и небезопасному кодированию.
* Унаследованные от Си опасные и провоцирующие ошибки возможности (макроопределения, адресная арифметика, неявное приведение типов, возможность прямого управления распределением памяти).
* Отсутствие встроенных механизмов статической валидации времени жизни объектов, приводящее к внезапному краху программ из-за обращения к уничтоженной переменной, или из-за неправильной многопоточной работы с объектами.
* Шаблоны порождают объёмный и не всегда оптимальный код. Частичное определение шаблонов усложняет как сам язык, так и программы, где оно используется.
* Множественное (в том числе виртуальное) наследование приводит к созданию громоздких иерархий классов, которые при любом изменении требований к программе могут потребовать серьёзного пересмотра.
* Сложный синтаксис и объёмная спецификация языка затрудняют его изучение.
* Язык не поощряет создание надёжного, легко читаемого и удобного в сопровождении кода, вместо этого зачастую предлагая выбор между короткими и простыми, но опасными средствами, унаследованными от Си, и новыми, объёмными и сложными, но более безопасными механизмами.
* Сложная и постоянно разрастающаяся стандартная библиотека, затрудняющая изучение языка.

# Основные требования к проекту

Проект должен составлять в себе многофайловую программу и классы.

От себя я решил добавить элементы управления с возможностью просмотра истории выполненных операций, очистки истории и вход в панель управления по паролю.

**Стадии и этапы разработки:**

* Написание основного меню программы;
* Создание заголовочных файлов;
* Подключение работы с файлами с помощью библиотеки fstream;
* Создание меню настроек и входа в меню по паролю;
* Реализация операций в меню настроек (просмотр истории, очистка истории, смена пароля).

# Выбор инструментария для написания проекта

Для написания проекта мной был выбран язык C++ и среда разработки Code::Blocks, так как в этой среде опыт у меня уже имеется.

# Программа

# Основы

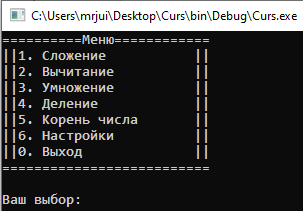
Проект в Code::Blocks делится на файлы расширений .cpp (исходный файл) и .h (заголовочный файл).

.cpp содержит в себе:

* Главное меню
* Функции из заголовочных файлов

.h содержит в себе:

* Класс, в зависимости от цели применения заголовочного файла
* Переменные и функции, определенные в классе

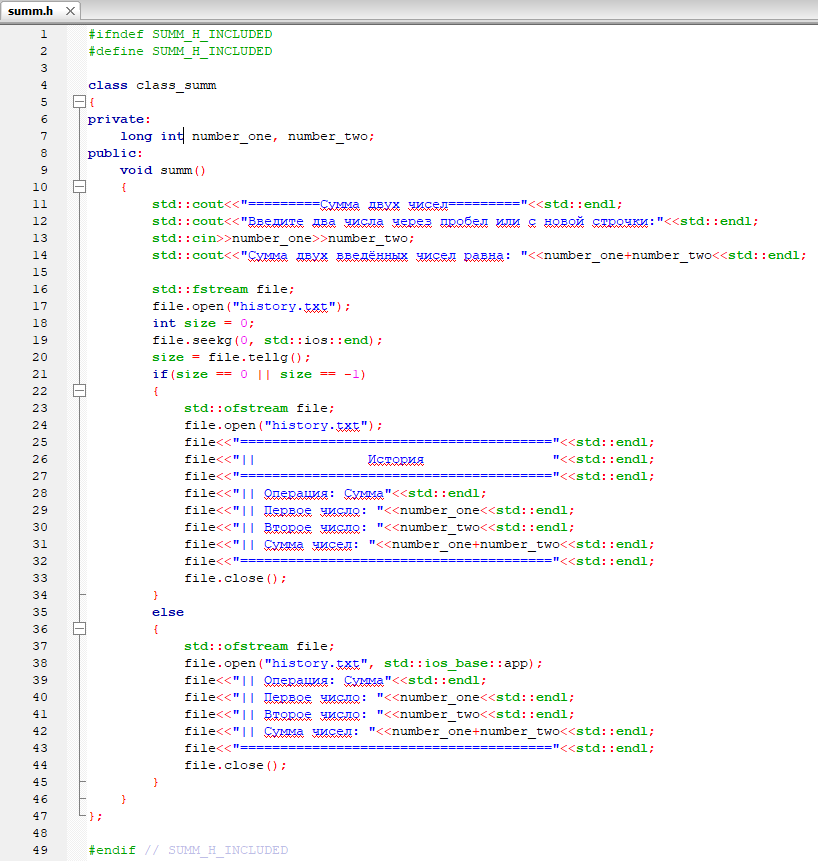


**Рисунок 3**. Главное меню программы

# Разработка классов и функций для каждого пункта меню

В первую очередь необходимо создать заголовочный файл для операции «Сложение», в котором будут определены класс и необходимые функции.

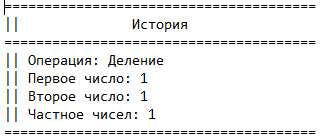
По такому же принципу будут разработаны следующие, заголовочные файлы для каждого пункта меню.



**Рисунок 4.** Вид готового заголовочного файла для пункта меню «Сложение»

# Подключение файлов к программе

Далее необходимо, чтобы каждая операция из меню записывалась в файл. Причём с таким условием, что если файл пуст, то сначала будет следующая строчка в текстовом документе:



**Рисунок 5.** Вид текстового документа после выполнения программы

Для проверки текстового документа на пустоту выполняется следующий кусочек кода:

std::fstream file;

file.open(“history.txt”);

int size = 0;

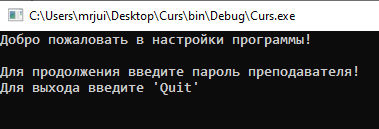
size = file.seekg(0, std::ios::end);

size = file.tellg();

В том случае, если в файле уже что-то есть, в файл будут записаны: операция, введенные числа, и результат операции без строчки «История».

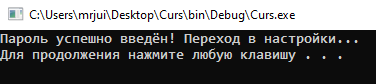
# Разработка панели управления

После ввода в главном меню значения «6» и нажатия на клавишу «Enter», программа направляет Вас к вводу пароля для доступа к панели управления.

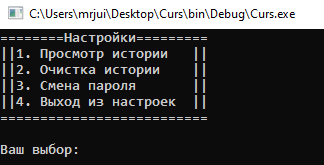


**Рисунок 6**. Вход в панель управления программой

Сразу же после удачно введенного пароля, который считывается с клавиатуры а потом сверяется с паролем из другого текстового документа под названием «password.txt», Вас перекидывает уже в саму панель управления программой.



**Рисунок 7.** Сообщение о удачно введенном пароле



**Рисунок 8.** Меню панели управления программой

С помощью панели управления программой у нас появляются такие возможности, как: просмотр истории работы программы, очистка истории и смена пароля.

# Эксплуатационная часть

**Функциональное назначение программы**

Основной функцией проекта «Калькулятор» является решение примеров для студентов. Программа помогает преподавателю следить за использованием программы студентом.

**Эксплуатационное назначение программы**

Программа может быть использована на любом компьютере.

# Состав функций

**Функции решения примеров**

Эти функции позволяют студентам решать поставленные перед ними задачи.

**Функции записи истории в текстовый документ**

Эти функции позволяют записать историю использования программы в текстовый документ.

**Функция авторизации в настройках**

Эта функция помогает преподавателю авторизоваться для доступа к меню настроек.

**Функция просмотра/очистки истории**

Функция просмотра истории позволяет преподавателю просмотреть историю использования программы студентом.

Функция очистки истории очищает текстовый документ с историей.

**Функция смены пароля**

Данная функция позволяет преподавателю изменить пароль, которвый необходим для доступа к меню настроек.

# Условия выполнения программы

**Минимальный состав программных средств**

Для функционирования проекта «Калькулятор» необходимо иметь программу, файл *history.txt* и *password.txt* в одной директории.

**Требование к персоналу (пользователю)**

Конечный пользователь должен обладать элементарными навыками работы с компьютером.

Данная программа предназначена для:

* Студенов, которые будут пользоваться своеобразным «калькулятором»;
* Преподователей, которые будут иметь собственную панель управления, в которой они смогут просматривать историю использования «калькулятора» студентом, очищать историю и менять свой пароль;
* Программистов, которые будут как дополнять программу, так и исправлять так называемые «баги».

# Выполнение программы

**Загрузка и запуск программы**

Загрузка и запуск проекта «Калькулятор» осуществляется в следующем порядке:

Необходимо скачать проект с GitHub’а и распаковать его в папку.

Для запуска необходимо:

* Скачать любую среду разработки для C++
* Открыть проект в среде разработки
* Запустить проект кнопкой «Run»

При запуске пользователь попадает в главное меню.

Главное меню изображено на рисунке 7.

При выборе одного из пунктов меню, программа укажет, что нужно ввести для того, чтобы появился результат программы на экран.

Пример одного из пунктов меню изображен на рисунке 10.

При выборе пункта меню настройки, пользователю необходимо ввести пароль для доступа к меню настроек.

Примеры ввода пароля и меню настроек изображены на рисунках 14 и 17.

**Выполнение программы**

Далее будут рассмотрены действия, которые доступны студенту:

1. Выбор математической операции

Пользователь может выбрать одну из операций: сложение, вычитание, умножение, деление и корень.

1. Выход из программы.

Для выхода из программы необходимо в главном меню выбрать пункт «0».

А сейчас будут рассмотрены действия, которые доступны пользователю, который владеет паролем от настроек (преподавателю):

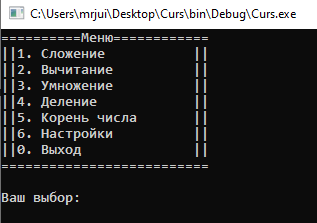
1. Все те же действия, что и у студента.

Преподавателю также доступны действия калькулятора.

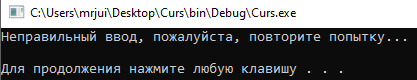
1. Настройки.

Для доступа к настройкам необходимо ввести пароль.

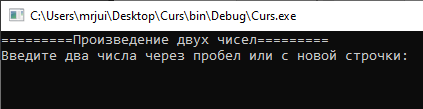
1. Просмотр истории.
2. Очистка истории.
3. Смена пароля.



**Рисунок 9.** Главное меню программы



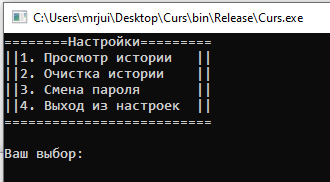
**Рисунок 10.** Сообщение о неправильном вводе



**Рисунок 9.** Поведение программы при выборе пункта "Умножение"

# 

**Рисунок 10.** Окно программы при выборе пункта "Настройки"



**Рисунок 11.** Меню панели управления

# Заключение

В данном проекте была проделана работа с многофайловым проектом в C++, работа с файлами и классами.

В качестве развития программы, я выделяю несколько тем:

* Звездочки при вводе пароля;
* Добавление новых операций с числами;
* Сделать так, чтобы в файле «password.txt» отображался закодированный пароль, который будет опознавать программа в самом коде.

Использование данного продукта может использоваться в любом учебном учреждении.

Разработка данной программы принесла достаточно опыта в сфере разработки простых приложений для учеников/преподавателей учебных учреждений.

При разработке программы использовался язык программирования C++.

# Список используемой литературы и интернет-источников

1. <https://habr.com/ru/post/216725/>
2. <https://ru.wikipedia.org/wiki/Заголовочный_файл>
3. <http://cppstudio.com/post/439/>
4. <http://cppstudio.com/post/446/>
5. <https://ru.wikipedia.org/wiki/C>++

# Приложение 1. Листинг главного модуля main.cpp

#include <iostream>

#include <cstdlib>

#include <math.h>

#include <fstream>

#include <locale>

#include <conio.h>

#include <string>

#include "summ.h"

#include "razn.h"

#include "multiplication.h"

#include "division.h"

#include "sqrt.h"

#include "settings.h"

using namespace std;

int main()

{

setlocale(LC\_ALL,"Russian");

int input\_number;

begin:

cout<<"==========Меню============"<<endl;

cout<<"||1. Сложение ||"<<endl;

cout<<"||2. Вычитание ||"<<endl;

cout<<"||3. Умножение ||"<<endl;

cout<<"||4. Деление ||"<<endl;

cout<<"||5. Корень числа ||"<<endl;

cout<<"||6. Настройки ||"<<endl;

cout<<"||0. Выход ||"<<endl;

cout<<"=========================="<<endl;

cout<<endl;

cout<<"Ваш выбор: ";

cin>>input\_number;

switch(input\_number)

{

case 1:

{

system("cls");

class\_summ Summ\_func;

Summ\_func.summ();

system("pause");

system("cls");

break;

}

case 2:

{

system("cls");

class\_razn Razn\_func;

Razn\_func.razn();

system("pause");

system("cls");

break;

}

case 3:

{

system("cls");

class\_multiplication Mult\_func;

Mult\_func.multiplication();

system("pause");

system("cls");

break;

}

case 4:

{

system("cls");

class\_division Div\_func;

Div\_func.division();

system("pause");

system("cls");

break;

}

case 5:

{

system("cls");

class\_sqrt Sqrt\_func;

Sqrt\_func.Sqrt();

system("pause");

system("cls");

break;

}

case 6:

{

system("cls");

settings Sett\_func;

Sett\_func.open\_settings();

system("cls");

break;

}

case 0:

{

system("cls");

cout<<"Выход из программы..."<<endl;

return 0;

}

default:

{

system("cls");

cout<<"Неправильный ввод, пожалуйста, повторите попытку..."<<endl<<endl;

system("pause");

system("cls");

break;

}

}

goto begin;

}

# Приложение 2. Листинг summ.h

#ifndef SUMM\_H\_INCLUDED

#define SUMM\_H\_INCLUDED

class class\_summ

{

private:

long int number\_one, number\_two;

public:

void summ()

{

std::cout<<"=========Сумма двух чисел========="<<std::endl;

std::cout<<"Введите два числа через пробел или с новой строчки:"<<std::endl;

std::cin>>number\_one>>number\_two;

std::cout<<"Сумма двух введённых чисел равна: "<<number\_one+number\_two<<std::endl;

std::fstream file;

file.open("history.txt");

int size = 0;

file.seekg(0, std::ios::end);

size = file.tellg();

if(size == 0 || size == -1)

{

std::ofstream file;

file.open("history.txt");

file<<"======================================="<<std::endl;

file<<"|| История "<<std::endl;

file<<"======================================="<<std::endl;

file<<"|| Операция: Сумма"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Сумма чисел: "<<number\_one+number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

else

{

std::ofstream file;

file.open("history.txt", std::ios\_base::app);

file<<"|| Операция: Сумма"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Сумма чисел: "<<number\_one+number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

}

};

#endif // SUMM\_H\_INCLUDED

# Приложение 3. Листинг razn.h

#ifndef RAZN\_H\_INCLUDED

#define RAZN\_H\_INCLUDED

class class\_razn

{

private:

long int number\_one, number\_two;

public:

void razn()

{

std::cout<<"=========Разность двух чисел========="<<std::endl;

std::cout<<"Введите два числа через пробел или с новой строчки:"<<std::endl;

std::cin>>number\_one>>number\_two;

std::cout<<"Разность двух введённых чисел равна: "<<number\_one-number\_two<<std::endl;

std::fstream file;

file.open("history.txt");

int size = 0;

file.seekg(0, std::ios::end);

size = file.tellg();

if(size == 0 || size == -1)

{

std::ofstream file;

file.open("history.txt");

file<<"======================================="<<std::endl;

file<<"|| История "<<std::endl;

file<<"======================================="<<std::endl;

file<<"|| Операция: Вычитание"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Разность чисел: "<<number\_one-number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

else

{

std::ofstream file;

file.open("history.txt", std::ios\_base::app);

file<<"|| Операция: Разность"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Разность чисел: "<<number\_one-number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

}

};

#endif // RAZN\_H\_INCLUDED

# Приложение 4. Листинг multiplication.h

#ifndef MULTIPLICATION\_H\_INCLUDED

#define MULTIPLICATION\_H\_INCLUDED

class class\_multiplication

{

private:

long int number\_one, number\_two;

public:

void multiplication()

{

std::cout<<"=========Произведение двух чисел========="<<std::endl;

std::cout<<"Введите два числа через пробел или с новой строчки:"<<std::endl;

std::cin>>number\_one>>number\_two;

std::cout<<"Произведение двух введённых чисел равна: "<<number\_one\*number\_two<<std::endl;

std::fstream file;

file.open("history.txt");

int size = 0;

file.seekg(0, std::ios::end);

size = file.tellg();

if(size == 0 || size == -1)

{

std::ofstream file;

file.open("history.txt");

file<<"======================================="<<std::endl;

file<<"|| История "<<std::endl;

file<<"======================================="<<std::endl;

file<<"|| Операция: Умножение"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Произведение чисел: "<<number\_one\*number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

else

{

std::ofstream file;

file.open("history.txt", std::ios\_base::app);

file<<"|| Операция: Умножение"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Произведение чисел: "<<number\_one\*number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

}

};

#endif // MULTIPLICATION\_H\_INCLUDED

# Приложение 5. Листинг division.h

#ifndef DIVISION\_H\_INCLUDED

#define DIVISION\_H\_INCLUDED

class class\_division

{

private:

long int number\_one, number\_two;

public:

void division()

{

begin:

std::cout<<"=========Частное двух чисел========="<<std::endl;

std::cout<<"Введите два числа через пробел или с новой строчки:"<<std::endl;

std::cin>>number\_one>>number\_two;

if(number\_one == 0 && number\_two == 0)

{

std::cout<<"Нельзя делить ноль на ноль!"<<std::endl;

std::cout<<"Повторите ещё раз..."<<std::endl;

system("pause");

system("cls");

goto begin;

}

else

{

std::cout<<"Частное двух введённых чисел равна: "<<number\_one/number\_two<<std::endl;

}

std::fstream file;

file.open("history.txt");

int size = 0;

file.seekg(0, std::ios::end);

size = file.tellg();

if(size == 0 || size == -1)

{

std::ofstream file;

file.open("history.txt");

file<<"======================================="<<std::endl;

file<<"|| История "<<std::endl;

file<<"======================================="<<std::endl;

file<<"|| Операция: Деление"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Частное чисел: "<<number\_one/number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

else

{

std::ofstream file;

file.open("history.txt", std::ios\_base::app);

file<<"|| Операция: Деление"<<std::endl;

file<<"|| Первое число: "<<number\_one<<std::endl;

file<<"|| Второе число: "<<number\_two<<std::endl;

file<<"|| Частное чисел: "<<number\_one/number\_two<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

}

};

#endif // DIVISION\_H\_INCLUDED

# Приложение 6. Листинг sqrt.h

#ifndef SQRT\_H\_INCLUDED

#define SQRT\_H\_INCLUDED

class class\_sqrt

{

private:

long int number\_one;

public:

void Sqrt()

{

std::cout<<"=========Корень числа========="<<std::endl;

std::cout<<"Введите число, корень которого Вам необходимо найти:"<<std::endl;

std::cin>>number\_one;

if(number\_one < 0)

{

number\_one = abs(number\_one);

std::cout<<"Корень числа равен: "<<sqrt(number\_one)<<std::endl;

}

else

{

std::cout<<"Корень числа равен: "<<sqrt(number\_one)<<std::endl;

}

std::fstream file;

file.open("history.txt");

int size = 0;

file.seekg(0, std::ios::end);

size = file.tellg();

if(size == 0 || size == -1)

{

std::ofstream file;

file.open("history.txt");

file<<"======================================="<<std::endl;

file<<"|| История "<<std::endl;

file<<"======================================="<<std::endl;

file<<"|| Операция: Корень"<<std::endl;

file<<"|| Число: "<<number\_one<<std::endl;

file<<"|| Корень: "<<sqrt(number\_one)<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

else

{

std::ofstream file;

file.open("history.txt", std::ios\_base::app);

file<<"|| Операция: Корень"<<std::endl;

file<<"|| Число: "<<number\_one<<std::endl;

file<<"|| Корень: "<<sqrt(number\_one)<<std::endl;

file<<"======================================="<<std::endl;

file.close();

}

}

};

#endif // SQRT\_H\_INCLUDED

# Приложение 7. Листинг settings.h

#ifndef SETTINGS\_H\_INCLUDED

#define SETTINGS\_H\_INCLUDED

class settings

{

private:

std::string pass, new\_pass;

public:

void open\_history()

{

std::cout<<"Открытие истории..."<<std::endl;

std::cout<<"После того, как Вы закроете историю, Вы будете перенаправлены в меню настроек."<<std::endl;

system("history.txt");

system("cls");

}

void clear\_history()

{

std::cout<<"Проводится очистка истории..."<<std::endl;

std::ofstream clear\_file("history.txt", std::ios\_base::trunc);

clear\_file.close();

}

void change\_password()

{

std::ofstream file("password.txt", std::ios\_base::trunc);

std::cout<<"Введите новый пароль:"<<std::endl;

std::cin>>pass;

file<<pass;

file.close();

std::cout<<"Пароль успешно изменён!"<<std::endl;

}

int settings\_menu()

{

begin:

int choise;

std::cout<<"========Настройки========="<<std::endl;

std::cout<<"||1. Просмотр истории ||"<<std::endl;

std::cout<<"||2. Очистка истории ||"<<std::endl;

std::cout<<"||3. Смена пароля ||"<<std::endl;

std::cout<<"||4. Выход из настроек ||"<<std::endl;

std::cout<<"=========================="<<std::endl;

std::cout<<std::endl;

std::cout<<"Ваш выбор: ";

std::cin>>choise;

switch(choise)

{

case 1:

{

system("cls");

open\_history();

goto begin;

}

case 2:

{

system("cls");

clear\_history();

system("pause");

system("cls");

goto begin;

}

case 3:

{

system("cls");

change\_password();

system("pause");

system("cls");

goto begin;

}

case 4:

{

return 0;

}

default:

{

system("cls");

std::cout<<"Неправильный ввод, пожалуйста, повторите попытку..."<<std::endl<<std::endl;

system("pause");

system("cls");

goto begin;

}

}

}

void open\_settings()

{

std::ifstream file("password.txt");

while(file)

{

file>>pass;

}

std::string entry;

std::cout<<"Добро пожаловать в настройки программы!"<<std::endl;

std::cout<<std::endl;

std::cout<<"Для продолжения введите пароль преподавателя!"<<std::endl;

retry:

std::cout<<"Для выхода введите 'Quit'"<<std::endl;

std::cin>>entry;

if(entry == pass)

{

system("cls");

std::cout<<"Пароль успешно введён! Переход в настройки..."<<std::endl;

system("pause");

system("cls");

settings\_menu();

}

else if(entry == "Quit")

{

}

else

{

system("cls");

std::cout<<"Пароль введён неверно! Повторите попытку..."<<std::endl<<std::endl;

goto retry;

}

}

};

#endif // SETTINGS\_H\_INCLUDED