Министерство образования и науки Российской Федерации

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский нижегородский

государственный университет им. Н.И. Лобачевского»

институт Информационных Технологий Математики и Механики

Отчет к лабораторным работам

за весенний семестр

Выполнил:

студент группы 381703-2

Мазенков Михаил Александрович

Проверил:

доцент каф. МОСТ

Сысоев А.В.

Нижний Новгород

2018

**Оглавление**

[Введение. 2](#_Toc515522504)

[Лабораторная работа 1. 3](#_Toc515522505)

[Руководство пользователя: 3](#_Toc515522506)

[Руководство программиста 4](#_Toc515522507)

[Эксперимент 5](#_Toc515522508)

[Список литературы 5](#_Toc515522509)

[Лабораторная работа 2. 5](#_Toc515522510)

[Руководство пользователя 6](#_Toc515522511)

[Руководство программиста. 9](#_Toc515522512)

[Эксперимент 9](#_Toc515522513)

[Список литературы 9](#_Toc515522514)

[Лабораторная работа 3. 9](#_Toc515522515)

[Руководство пользователя 10](#_Toc515522516)

[Руководство программиста 13](#_Toc515522517)

[Эксперимент 13](#_Toc515522518)

[Список литературы 13](#_Toc515522519)

[Лабораторная работа 4. 14](#_Toc515522520)

[Руководство пользователя 15](#_Toc515522521)

[Руководство программиста 16](#_Toc515522522)

[Список литературы 18](#_Toc515522523)

[Лабораторная работа 5. 19](#_Toc515522524)

[Руководство пользователя 20](#_Toc515522525)

[Эксперимент 25](#_Toc515522526)

[Лабораторная работа 6. 26](#_Toc515522527)

[Руководство пользователя 26](#_Toc515522528)

[Руководство программиста 28](#_Toc515522529)

[Эксперимент 29](#_Toc515522530)

[Приложение 30](#_Toc515522531)

# **Введение.**

Целью данных работ является обучение и совершенствование навыков разработки и использования классов и их систем в языке программирования С++.

# **Лабораторная работа 1.**

Была поставлена задача разработать класс, которая выполняет функции конвертера температур:

* Класс должен хранить температуру в градусах Цельсия и предоставлять методы по его преобразованию в другие единицы измерения (Фаренгейт, Кельвин, Ранкин, …).
* Класс должен предоставлять операции: 1) установить текущую температуру в градусах Цельсия, 2) узнать текущую температуру в градусах Цельсия, 3) узнать текущую температуру в выбранной единице измерения (из списка поддерживаемых).
* Класс должен содержать все необходимые конструкторы, оператор присваивания, а также «уметь» выводить себя на консоль.

# **Руководство пользователя:**

1. При запуске программа предложит пользователю ввести необходимую ему температуру(рис.1)

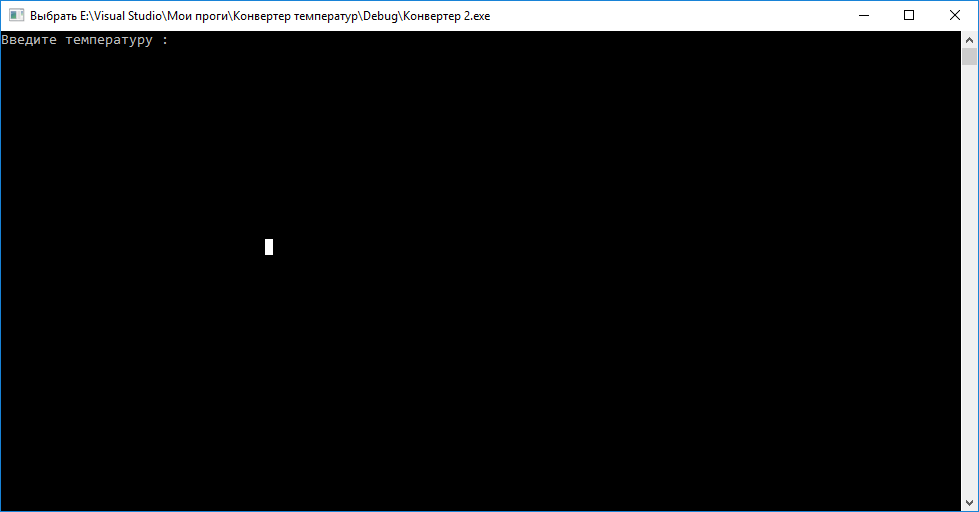


Рис.1

1. После ввода интересующей температуры, программа проведёт необходимые расчёты и выведет информацию на экран консоли (рис. 2)

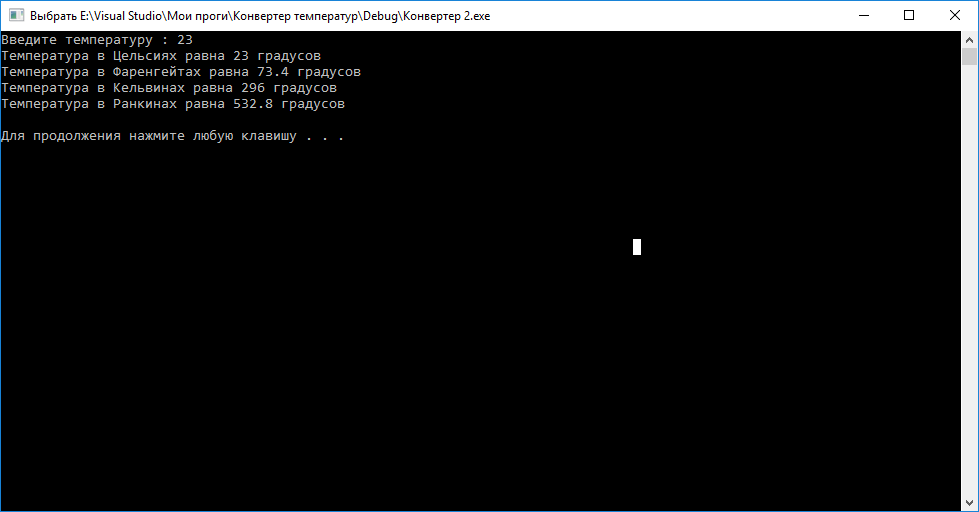


Рис.2

# **Руководство программиста**

Класс ConverterTemperature имеет методы, которые реализуют перевод температуры в градусах Цельсия в Фаренгейты, Кельвины и Ранкины.

Методы, выводящие значения на экран консоли:

**void CelsShow()**

**void FarShow()**

**void KelvShow()**

**void RanShow()**

Методы, проводящие перевод в нужные температуры:

**double Farengeit()**

**double Kelvin()**

**double Rankin()**

# **Эксперимент**

В результате эксперимента, показанного на рис.2, мы видим, что на экран консоли правильно выводится температура во всех интересующих нас величинах. Из чего следует вывод, что программа работает правильно

**Список литературы**

* http://cppstudio.com/post/6964/
* https://code-live.ru/post/cpp-classes/
* http://codelessons.ru/cplusplus/klassy-v-c-rukovodstvo-dlya-nachinayushhix.html

# **Лабораторная работа 2.**

Была поставлена задача разработать класс, являющийся календарём событий

* Класс должен позволять сохранять даты заданных событий в формате: год, месяц, день (в диапазоне от 1 января 1 года до 31 декабря 2020 года), наименование события. На каждый день может приходиться только одно событие. Общее число событий – не более 30.
* Класс должен предоставлять операции: 1) установить событие, 2) узнать дату выбранного события, 3) вычислить разницу между заданной датой и датой события (в годах, месяцах, днях), 4) сформировать новое событие, сдвинув выбранное существующее событие на заданное смещение (в годах, месяцах, днях) в меньшую и в большую сторону.
* Класс должен содержать все необходимые конструкторы, деструктор, оператор присваивания, а также «уметь» выводить себя на консоль.

# **Руководство пользователя**

1. Первым делом программа спросит пользователя, сколько событий он хочет ввести, попросит дать номер и название событию, а так же дату его проведения(рис.1).

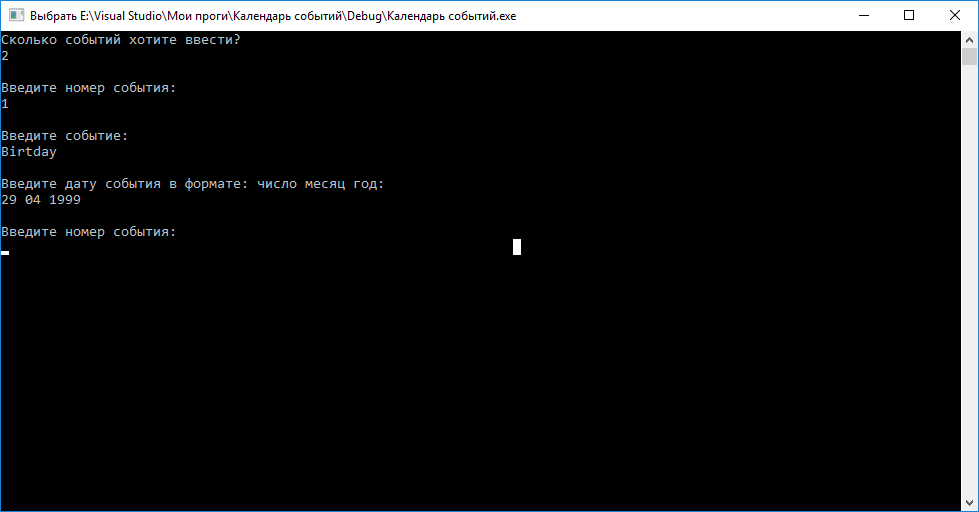


Рис.1

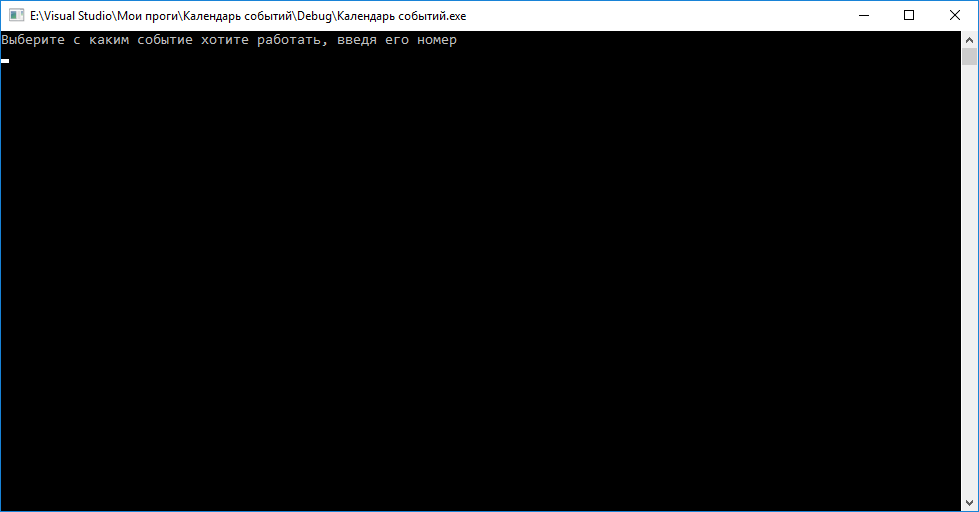
1. Далее, программа попросит пользователя ввести номер события, с которым он хочет работать. (рис.2) 

Рис.2

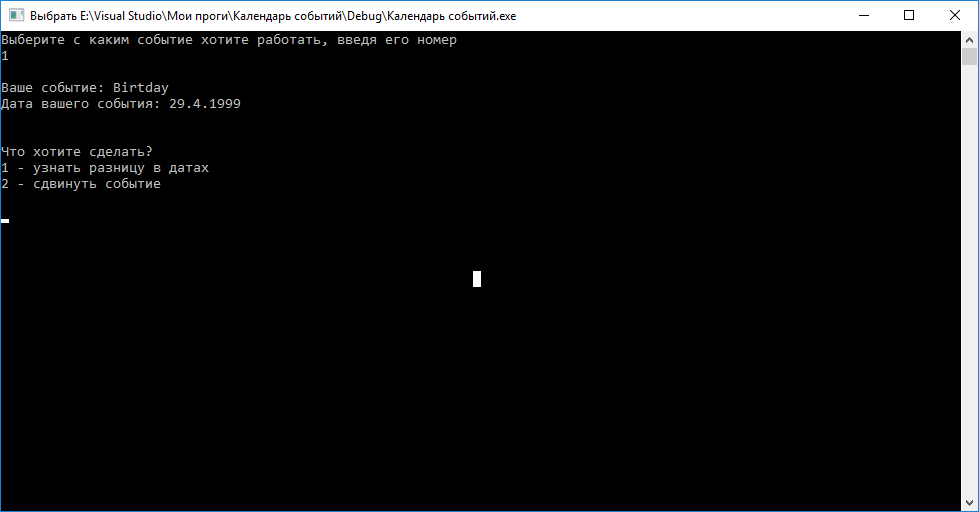
1. Введя номер события пользователь увидит информацию о нём и предложение поработать с ним. (рис.3) 

Рис 3

1. При выборе первого пункта меню программ предложит ввести дату, разницу с которой необходимо узнать и программ выдаст эту разницу. (рис.4)

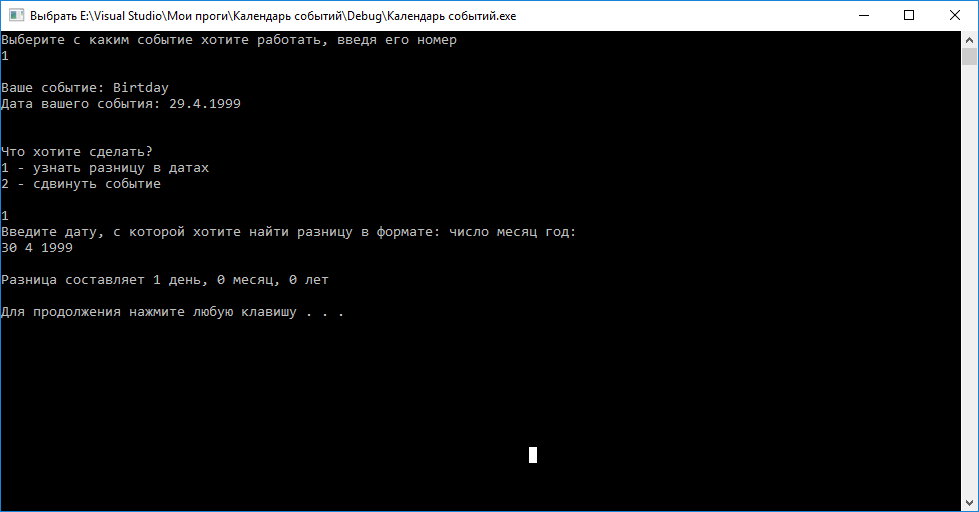


Рис.4

1. При выборе второго пункта меню пользователю будет предложено ввести количество дней, месяц и лет, на которое нужно будет подвинуть дату события и предложено выбрать в какую сторону производить сдвиг, вперёд или назад. В итоге программа установит для вашего события новую дату.(рис.5)

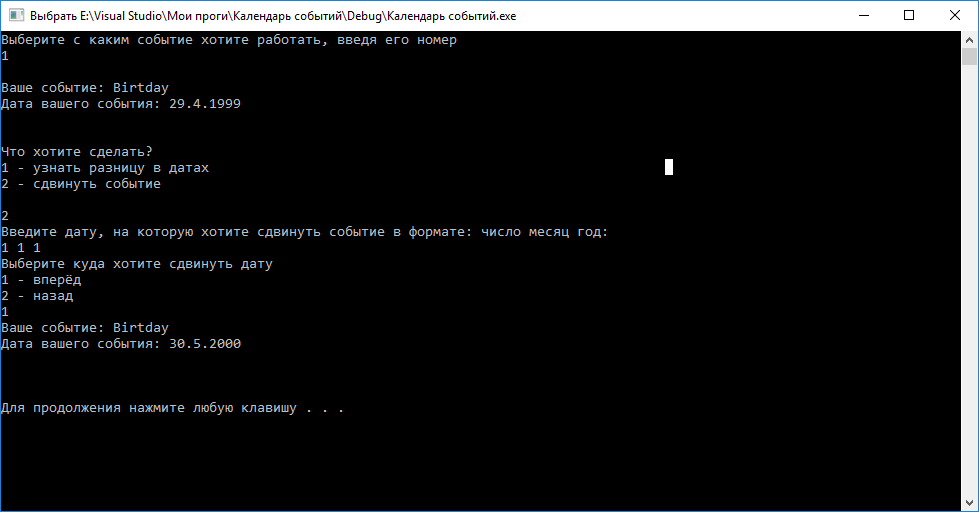


Рис.5

# **Руководство программиста.**

Основные методы:

**void ChooseEvent ()**

Данный метод принимает на вход число, определяющее количество необходимых нам событй.

**void ShowEven()**

Данный метод выводит на экран консоли информацию о событие.

**void CheckDate()**

Данный метод считает разницу в датах

**void SetNewDate()**

Данный метод выполняет сдвиг нашей даты

# **Эксперимент**

В результате эксперимента, показанного на последующих рисунках 1-5, мы видим, что справляется с поставленной перед нами задачей. Программа функционирует и работает правильно.

# **Список литературы**

* http://cppstudio.com/post/6964/
* https://code-live.ru/post/cpp-classes/
* http://codelessons.ru/cplusplus/klassy-v-c-rukovodstvo-dlya-nachinayushhix.html
* http://cppstudio.com/post/432/
* http://codelessons.ru/cplusplus/dinamicheskie-massivy-i-peremennye-vse-samoe-glavnoe.html

# **Лабораторная работа 3.**

Была поставлена задача разработать класс, выполняющий табулирование функций.

* Класс должен позволять выполнять табулирование произвольной функции одной переменной, заданной в виде функции языка C++.
* Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
* Класс должен предоставлять следующие операции: 1) задать текущую функцию, 2) задать текущее число точек табулирования, 3) узнать текущее число точек табулирования, 4) задать отрезок табулирования, 5) узнать отрезок табулирования, 6) выполнить табулирование функции, 7) выдать результаты табулирования, 8) сохранить результаты табулирования в файл.

# **Руководство пользователя**

1. При запуске программа покажет на экране консоли функции, табулирование которых она способна выполнять(рис.1).

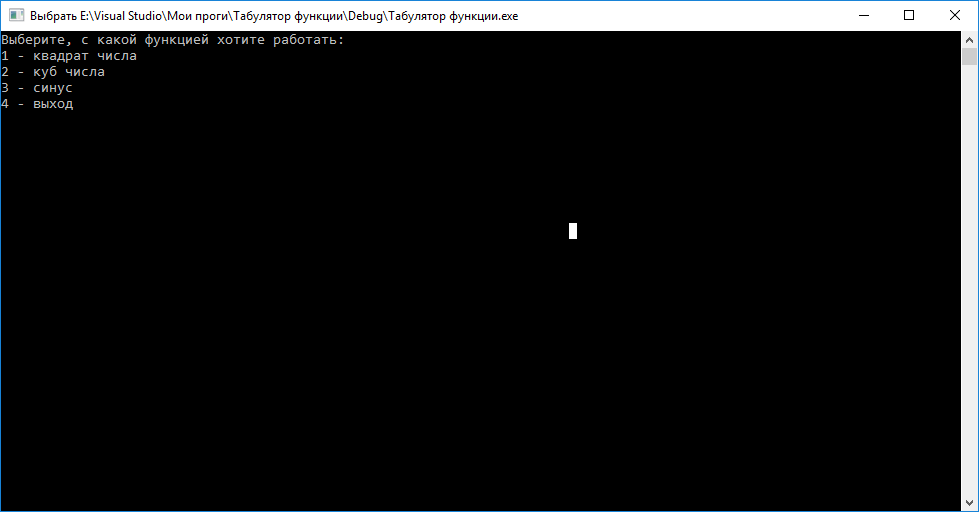


Рис.1

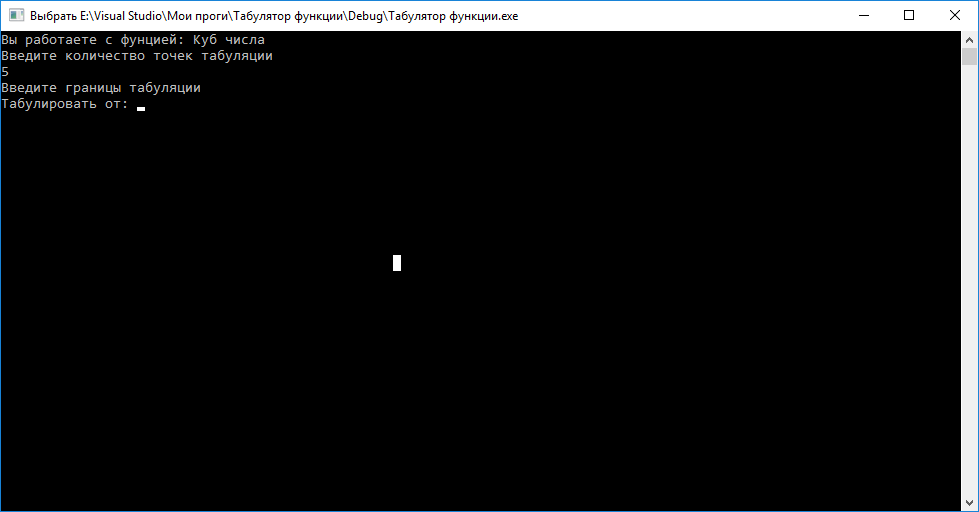
1. Для примера выберем второй пункт. На первом шаге программа попросит пользователя ввести количество точек табуляции. (рис.2) 

Рис.2

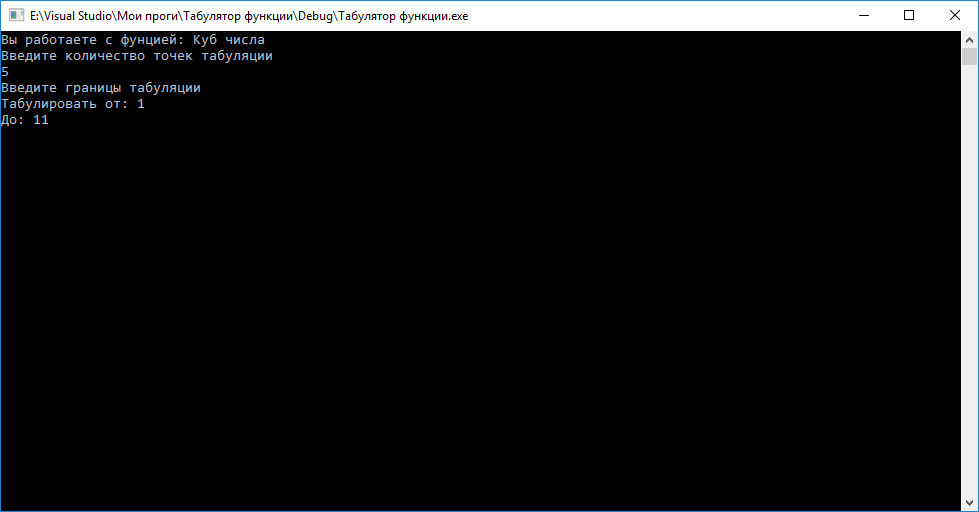
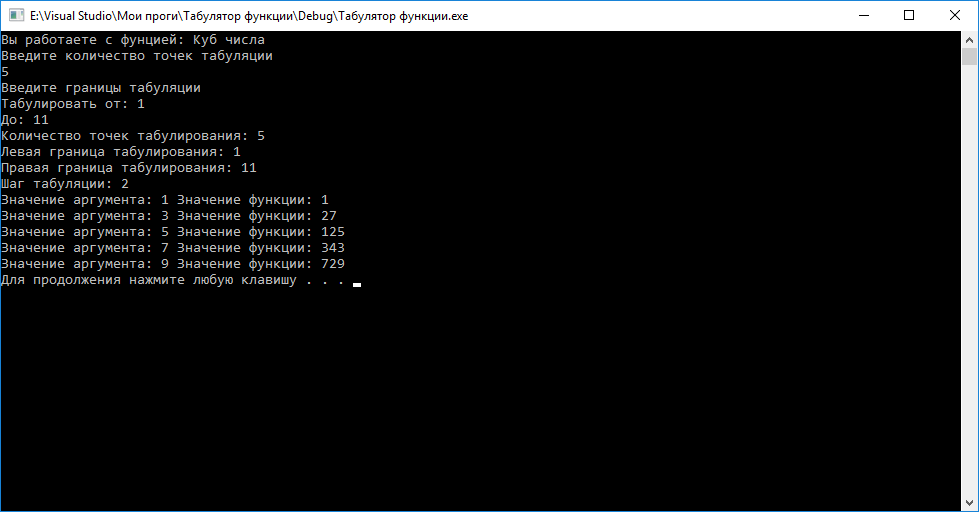
1. На втором шаге попросит ввести границы табуляции. (рис.3) 

Рис.3

1. На последнем же шаге программа выдаст результаты табулирования и сохранит данные в файл с название «Отчёт» в папку с проектом. (рис.4-5)

Рис.4

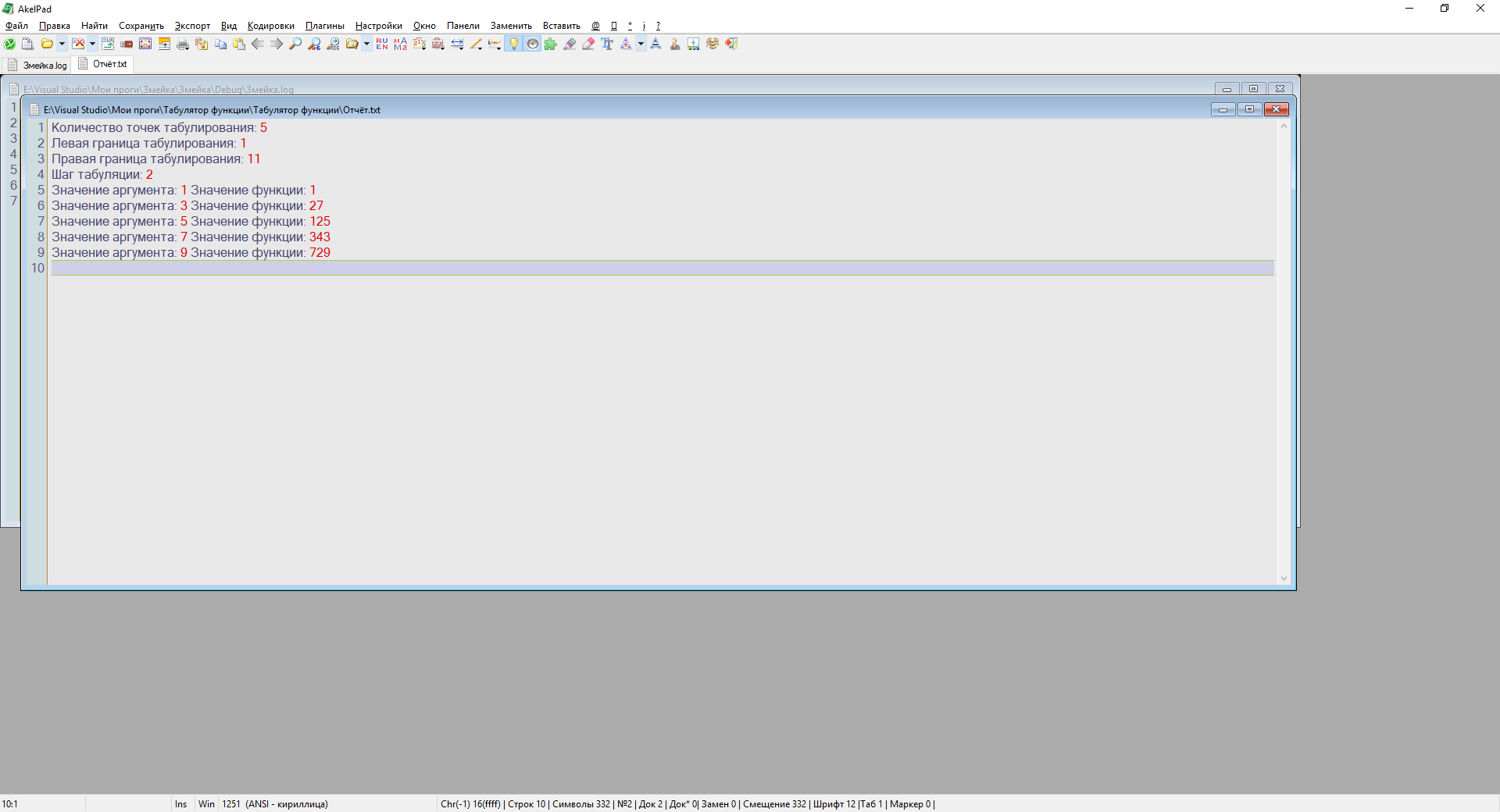


Рис.5

# **Руководство программиста**

Основные методы:

**void SetFunction()**

Данный метод устанавливает функцию, с которые пользователю предстоит работать.

**void SetBorder()**

Данный метод реализует установку границ табулирования.

**void Tabulate()**

Данный метод выполняет непосредственное табулирование функции.

**void ShowResult()**

Данный метод выдаёт всю информацию о табулировании (количество точек табулирования, границы и шаг)

**void WorkWithFile()**

Данный метод сохраняет результаты табулирования в файл.

# **Эксперимент**

В результате эксперимента, показанного на рисунках 1-5 мы можем сделать вывод, что программа функционирует правильно.

# **Список литературы**

* http://cppstudio.com/post/6964/
* https://code-live.ru/post/cpp-classes/
* http://codelessons.ru/cplusplus/klassy-v-c-rukovodstvo-dlya-nachinayushhix.html
* http://cppstudio.com/post/432/
* http://codelessons.ru/cplusplus/dinamicheskie-massivy-i-peremennye-vse-samoe-glavnoe.html
* http://cppstudio.com/post/446/
* http://purecodecpp.com/archives/2751

# **Лабораторная работа 4.**

Передо мной была поставлена задача написать класс, который бы выполнял функции фильмотеки. Этот класс должен выполнять следующие функции:

* Хранить информацию о фильмах. Каждый фильм описывается следующими данными: название, режиссер, сценарист, композитор, дата выхода в прокат (день, месяц, год), сборы (в рублях). Фильмы хранятся упорядоченно по названию и годам. Данные вводятся на русском языке.
* Содержать необходимые служебные методы.
* Предоставлять следующие операции: 1) добавить фильм, 2) изменить данные выбранного фильма, 3) найти фильм по названию и году, 4) выдать все фильмы заданного режиссера, 5) выдать все фильмы, вышедшие в прокат в выбранном году, 6) выдать заданное число фильмов с наибольшими сборами, 7) выдать заданное число фильмов с наибольшими сборами в выбранном году, 8) узнать текущее число фильмов, 9) удалить фильм, 10) сохранить фильмотеку в файл, 11) считать фильмотеку из файла.

# **Руководство пользователя**

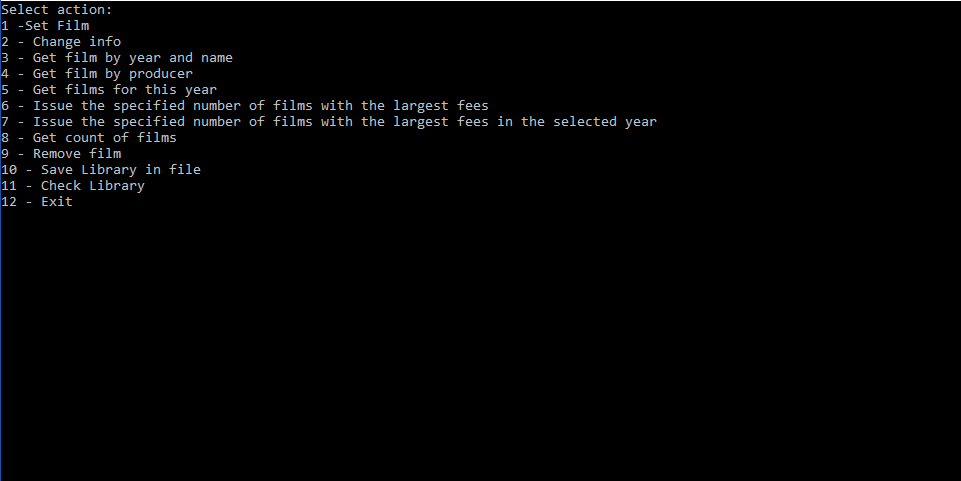
1. При запуске программы перед Вами появится меню(рис.1)

Рис.1

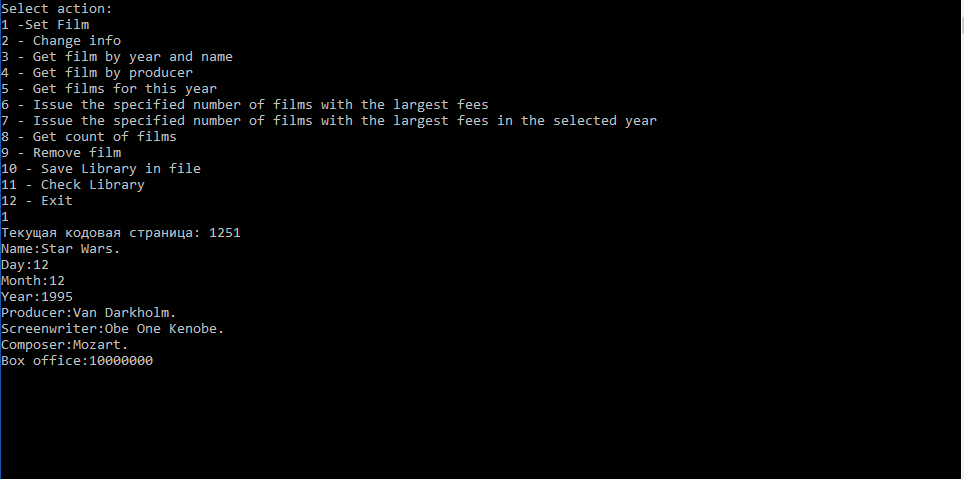
1. Выберите пункт.
   1. Если Вы выбрали пункт “Set Film”, то далее нужно ввести по очереди: название, день, месяц, год, продюсера, сценариста, композитора и кассовые сборы(рис.2).

Рис.2

* 1. Если Вы выбрали пункт “Change Info”, то введите название фильма и год выпуска, а далее информацию, которую хотите поменять об этом фильме.
  2. Если Вы выбрали пункт “Get film by name and year”, то введя название и год выпуска, Вы получите список фильмов с таким названием и годом выпуска.
  3. Если Вы выбрали пункт “Get film by producer”, то введя продюсера, Вы получите список фильмов введенного продюсера.
  4. Если Вы выбрали пункт “Get film for this year”, то введя год выпуска, Вы получите список фильмов, выпущенных в данном году.
  5. Если Вы выбрали пункт “Issue the specified number of films with the largest fees”, то введя желаемое количество фильмов, программа выведет заданное количество фильмов с наибольшими кассовыми сборами.
  6. Если Вы выбрали пункт “Issue the specified number of films with the largest fees in the selected year”, то введя количество и год выпуска, программа выведет заданное количество фильмов с наибольшими кассовыми сборами в заданном году
  7. Если Вы выбрали пункт “Get count of films”, ты Вы получите количество фильмов в фильмотеке.
  8. Если Вы выбрали пункт “Remove film”, то нужно будет ввести название и год выпуска фильма, который Вы хотите удалить.
  9. Если Вы выбрали пункт “Save library in file”, то все фильмы, введенные в данном запуске, сохранятся в файл FilmLibrary.txt.
  10. Если Вы выбрали пункт “Check Library”, то программа считает список фильмов из файла для дальнейшей работы.

# **Руководство программиста**

**struct Film –** структура хранящая данные о фильме, с перегруженным выводом.

**class FilmLibrary –** класс в котором хранятся методы, необходимые для правильной работы.

**vector <Film> SortLib()**

Вспомогательный метод, сортирующий вектор фильмов по имени и году выпуска.

**vector <Film> SortFees()**

Вспомогательный метод, сортирующий вектор по кассовым сборам.

**void AddFilm()**

Метод, принимающий на вход 3 вектора: композитор, продюсер и название и 4 целых значения: кассовые сборы, день, месяц, год.

**void ChangeInfo()**

Метод, принимающий на вход целое значение, отвечающее за выбор изменяемой информации о фильме, вектор названия и год, по которому происходит поиск фильма, и указатель на массив символов. Данный метод меняет информацию о фильме в зависимости от выбора.

**Film GetFilm()**

Данный метод принимает на вход строку – название фильма и целое значение – год выпуска. Метод возвращает фильм с данным названием и годом выпуска.

**vector <Film> GetFilmProd()**

Данный метод принимает на вход строку – продюсера и возвращает фильмы данного продюсера.

**vector <Film> GetFilmYear()**

Данный метод принимает на вход целое значение – год выпуска. Метод возвращает список фильмов, выпущенных в введенном году.

**vector <Film> GetFilmsWithMaxFees()**

Данный метод принимает на вход целое значение – количество. Метод выводит список, введенной вам длины, фильмов с наибольшими кассовыми сборами.

**vector <Film> GetFilmsWithMaxFeesInYear()**

Данный метод работает так же, как и описанный выше метод, но выводит список, заданной длины с наибольшими кассовыми сборами в введённым Вами году.

**int CountFilms()**

Данный метод возвращает количество фильмов в фильмотеке.

**void DeleteFilm()**

Данный метод принимает на вход целое значение – год выпуска и строку – название фильма. Метод удаляет фильм из фильмотеки.

**void SaveInFile()**

Данный метод сохраняет текущую фильмотеку в файл FilmLibrary.txt

**void CheckFile()**

Данный метод считывает фильмотеку из файла FilmLibrary.txt для дальнейшей работы с ней.

**void PrintLib()**

Данный метод выводит текущую фильмотеку на консоль.

# **Список литературы**

* http://cppstudio.com/post/6964/
* https://code-live.ru/post/cpp-classes/
* http://codelessons.ru/cplusplus/klassy-v-c-rukovodstvo-dlya-nachinayushhix.html
* http://cppstudio.com/post/432/
* http://codelessons.ru/cplusplus/dinamicheskie-massivy-i-peremennye-vse-samoe-glavnoe.html
* http://cppstudio.com/post/446/
* http://purecodecpp.com/archives/2751
* http://cppstudio.com/post/9033/

# **Лабораторная работа 5.**

Была поставлена задача разработать два взаимодействующих между собой класса Железнодорожная касса и Горьковская железная дорога, которые отвечают следующим требованиям:

* Класс Железнодорожная касса должен имитировать работу кассы по продаже билетов на поезда Нижний Новгород – Москва. Считать, что продажа билетов проводится на поезда в пределах 30 дней от текущей даты. Считать, что по маршруту Нижний Новгород – Москва курсирует три скоростных поезда «Ласточка», один фирменный и один скорый поезд в сутки в каждом направлении. Все вагоны в поездах «Ласточка» однотипны и содержат по 100 сидячих мест. В фирменном и скором поездах вагоны трех типов: плацкартные (27 верхних, 27 нижних мест), купейные (18 верхних, 18 нижних мест), СВ (18 нижних мест). Число вагонов в поездах «Ласточка» – 8. В фирменном поезде – 2 вагона СВ, 6 купейных вагонов, 4 плацкартных вагона. В скором поезде – 4 купейных вагона, 8 плацкартных вагонов. Поезда идентифицируются номерами (десять номеров из диапазона от 1 до 100), вагоны – номерами (целые числа от 1 до 12), места – номерами (целые числа от 1 до максимума для данного типа вагона).
* Информация о всех поездах и всех проданных билетах хранится в классе Горьковская железная дорога. Для каждого поезда, каждого типа вагона и каждого типа места установлена стоимость билета. Считать, что все поезда не делают промежуточных остановок по маршруту.
* Класс Железнодорожная касса должен предоставлять следующие операции: 1) принять данные покупателя: дату, поезд, тип вагона (если есть выбор), количество билетов каждого возможного вида (если есть выбор), ФИО пассажиров 2) проверить наличие свободных мест по запросу покупателя (при невозможности выдать все билеты в одном вагоне, считать заказ невыполнимым), 3) зарезервировать места, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, номер поезда, номер вагона, номер места, ФИО пассажира, станция отправления, станция прибытия).
* Класс Горьковская железная дорога должен использоваться для поддержки работы класса Железнодорожная касса и может быть разработан в минимально-необходимом варианте.

# **Руководство пользователя**

1. При запуске программа попросит пользователя ввести ФИО. (рис.1)

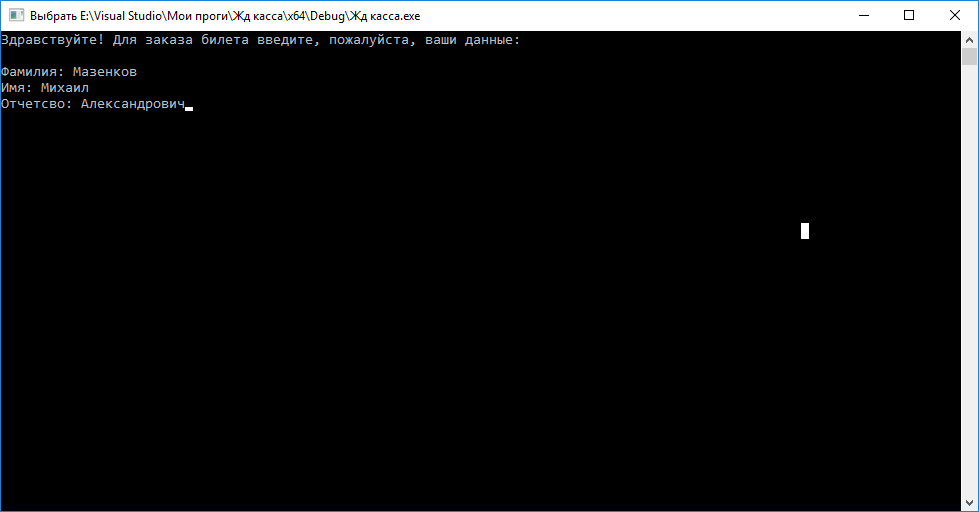


Рис.1

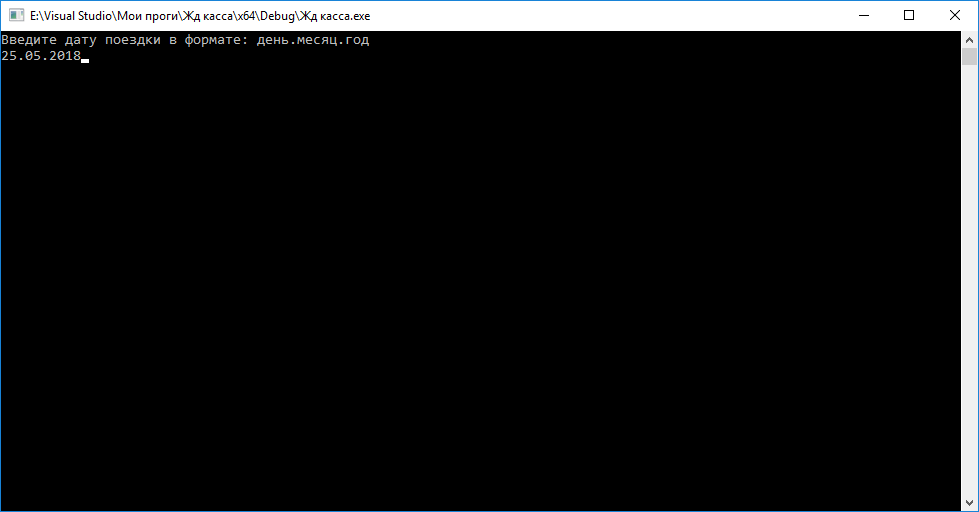
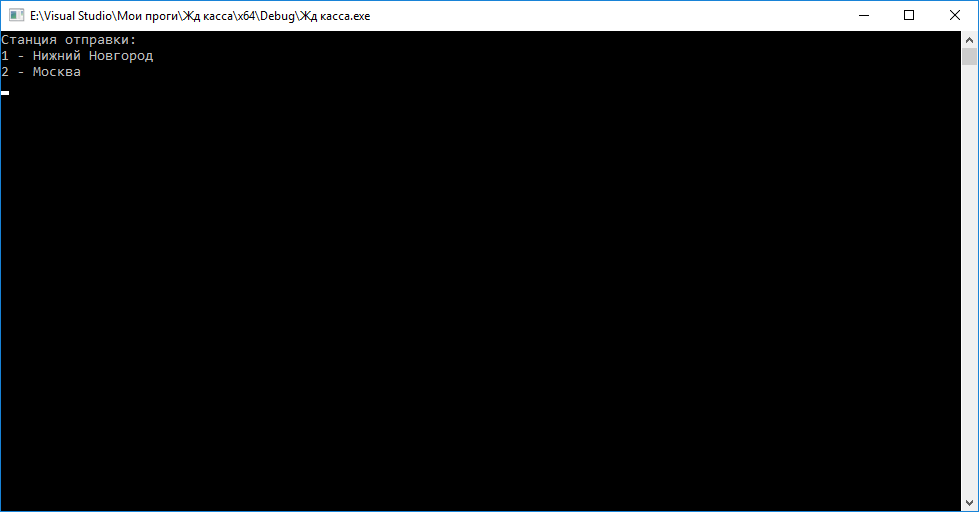
1. Далее программа попросит ввести дату поездки. (рис.2) 
2. На третьем шаге программа попросит выбрать станцию отправки (рис.3) и тип поезда (рис.4) 

Рис.3

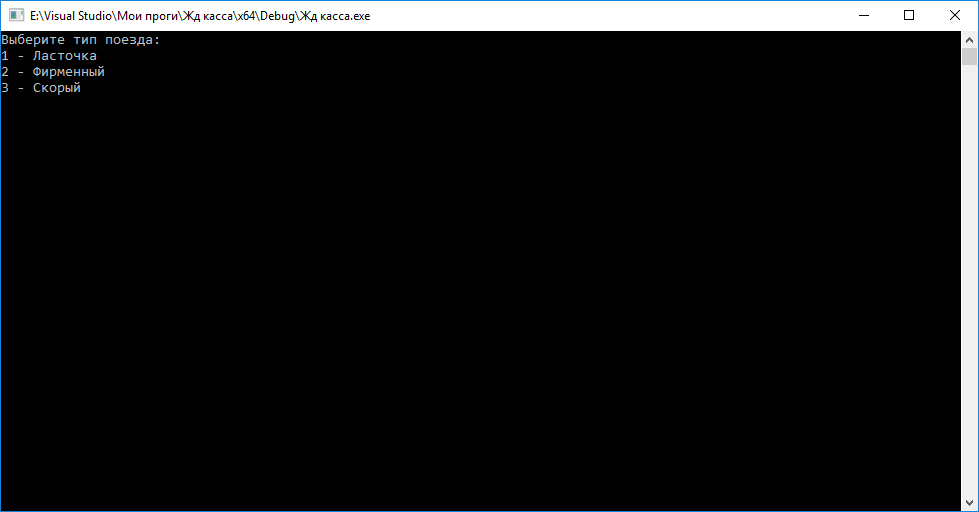


Рис.4

1. Далее пользователь увидит меню. (рис.5)

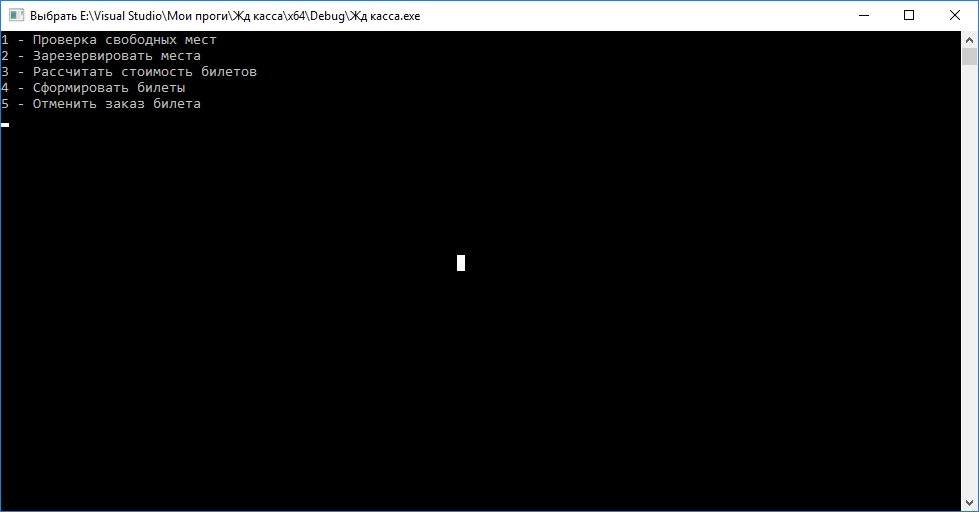


Рис.5

1. Выбрав пункт меню 2, пользователь увидит количество возможных мест и сможет выбрать номер нужного. (рис.6)

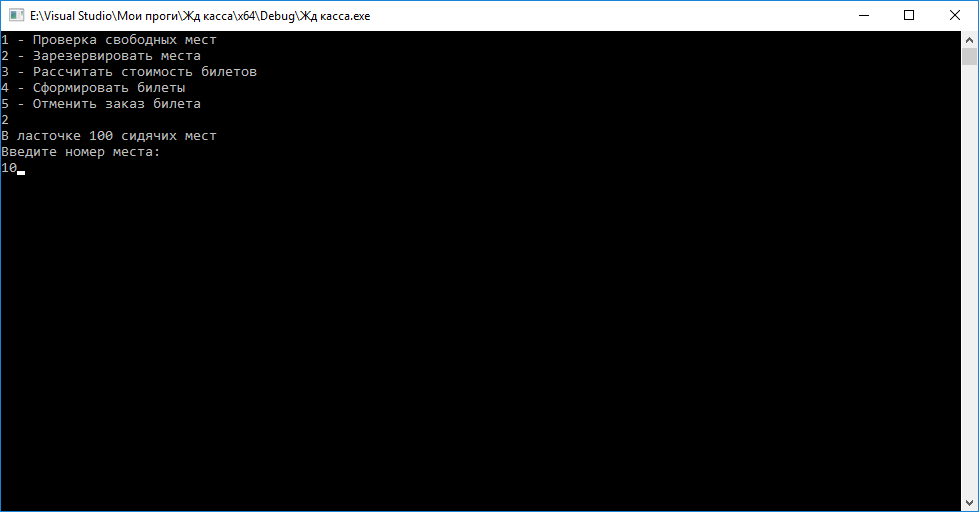


Рис.6

1. Выбрав пункт меню 1, пользователь увидит зарезервированные места. (рис.7)

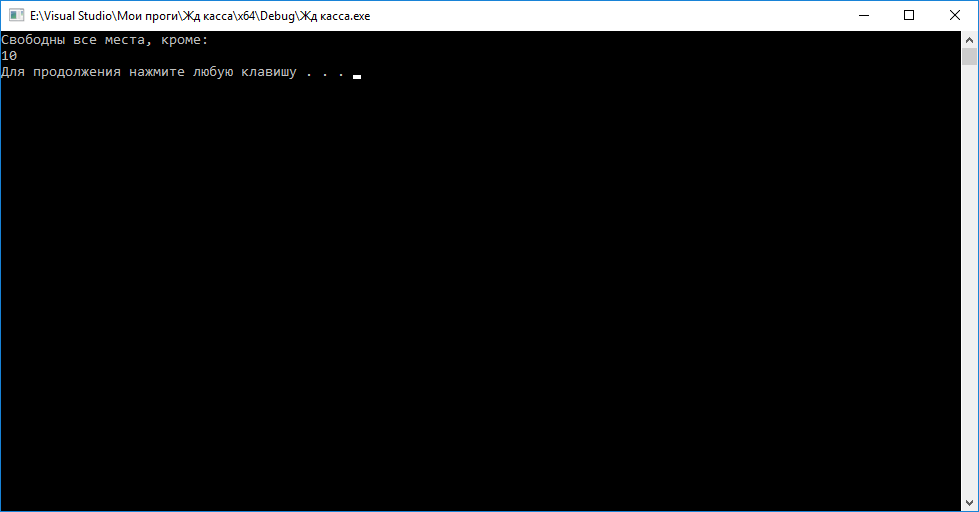


Рис.7

1. Выбрав пункт меню 3, программа рассчитает стоимость вашего билета. (рис.8)

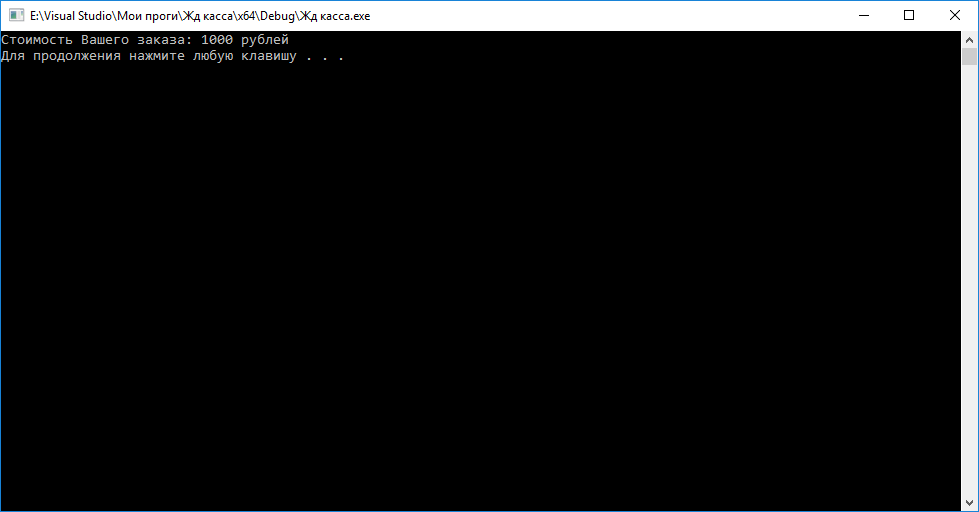


Рис.8

1. Выбрав пункт меню 4, программа выдаст пользователю информацию о его билете. (рис.9)

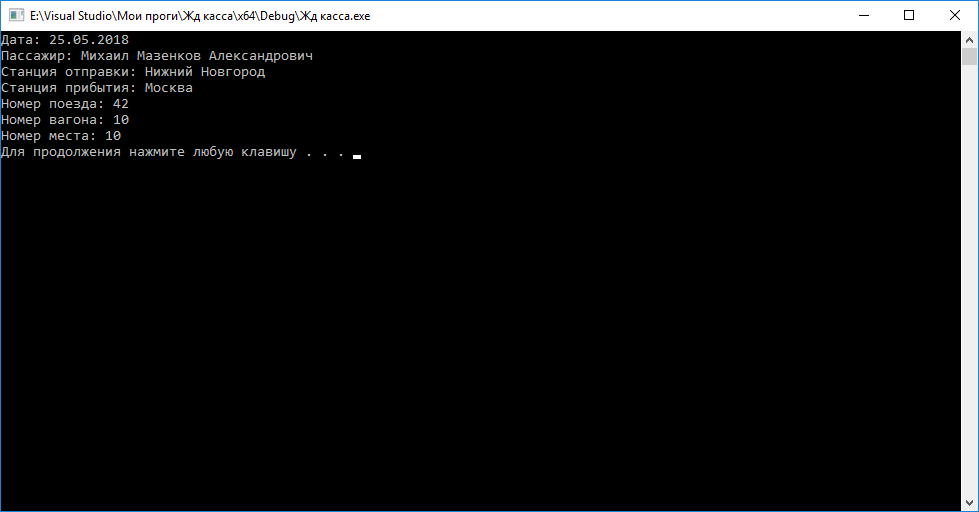


Рис.9

1. Выбрав пункт меню 5, программа очистит все выбранные места.

**Руководство программиста**

Основные методы:

**void ReservPlace()**

Данный метод резервирует выбранные места.

**void Calculate()**

Данный метод высчитывает стоимость билета.

**void CheckPlace()**

Данный метод проверяет наличие свободных мест.

**void CancelOrder()**

Данный метод очищает зарезервированные места.

**void CreateTicket()**

Данный метод показывает пользователю информацию по его билету.

# **Эксперимент**

В результате эксперимента, показанного на рисунках 1-9 мы можем сделать вывод, что программа функционирует правильно.

# **Лабораторная работа 6.**

Была поставлена задача разработать класс или систему классов и реализовать с её помощью игры «быки и коровы»

Суть игры:

* Играют два игрока (человек и компьютер).
* Игрок выбирает длину загадываемого числа – n.
* Компьютер «задумывает» n-значное число с неповторяющимися цифрами.
* Игрок делает попытку отгадать число – вводит n-значное число с неповторяющимися цифрами.
* Компьютер сообщает, сколько цифр угадано без совпадения с их позициями в загаданном числе (то есть количество коров) и сколько угадано вплоть до позиции в загаданном числе (то есть количество быков).
* Игрок делает попытки, пока не отгадает всю последовательность.

# **Руководство пользователя**

1. При запуске программа выдаст пользователю правила игры и спросит о его готовности играть. (рис.1)

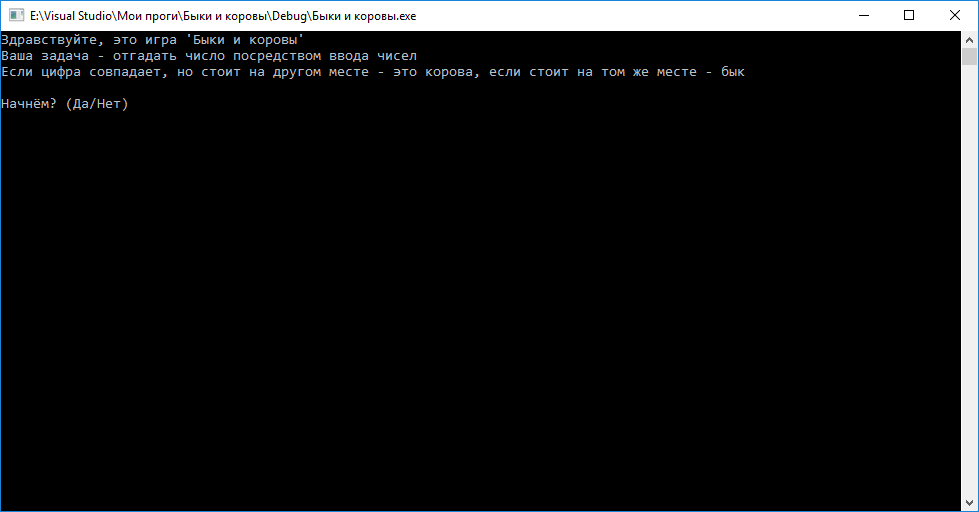


Рис.1

1. Далее программа попросит пользователя ввести длину числа. (рис.2)

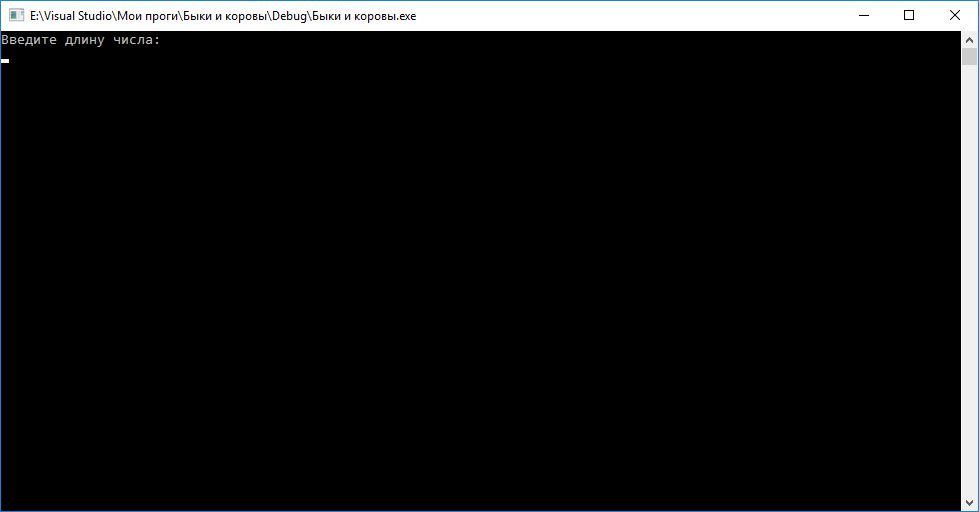


Рис.2

1. После этого начинается сам процесс игры. Число выведено для экран консоли для наглядности. (рис.3)

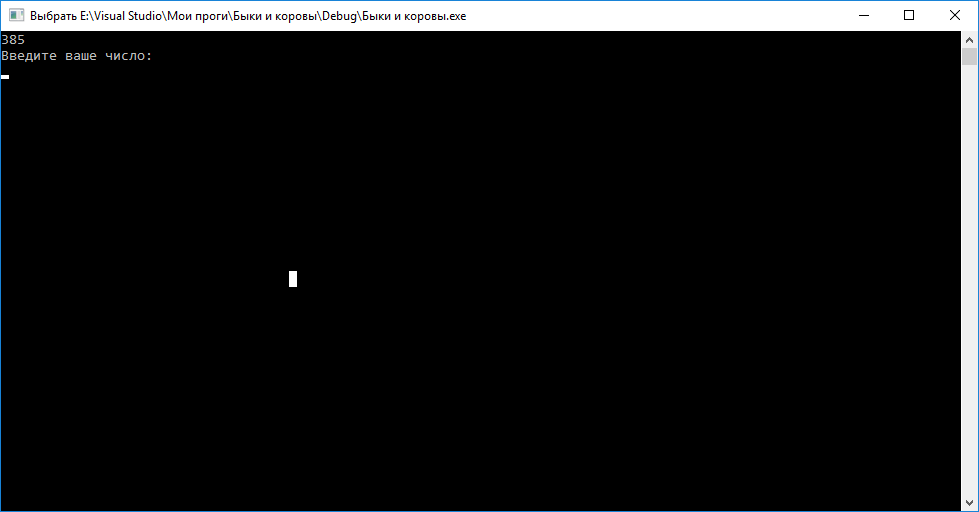


Рис.3

1. После того как пользователь угадает число, программа поздравит его и выведен на экран консоли количество шагов, которое потратил пользователь для того, чтобы угадать число. (рис.4)

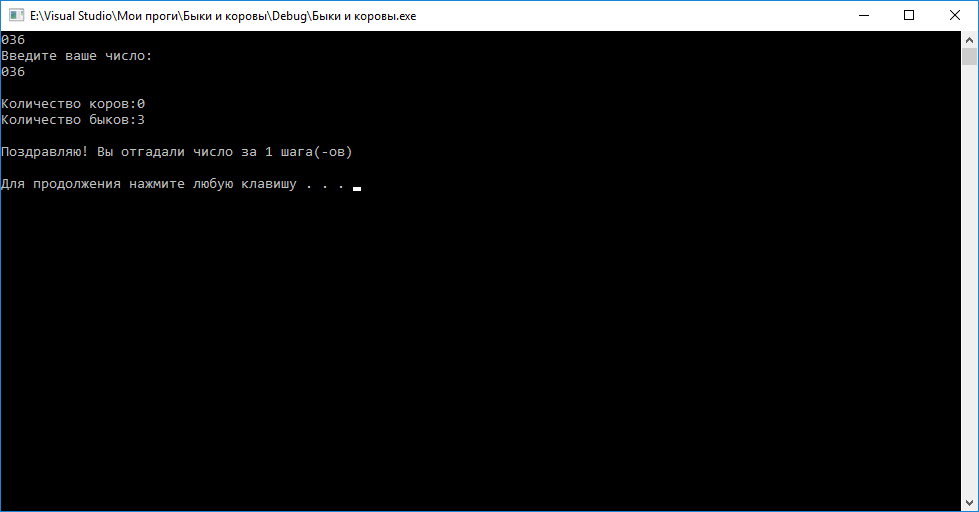


Рис.4

# **Руководство программиста**

Основные методы:

**void SetLenght()**

Данный метод устанавливает длину угадываемого числа.

**vector <int> RandomNamber()**

Данный метод генерирует случайное число для игры.

**void CheckBulls()**

Данный метод подсчитывает количество быков.

**void CheckBulls ()**

Данный метод подсчитывает количество коров.

**bool CheckWin()**

Данный метод проверяет условие победы в игре.

**void ClearCowBulls ()**

Данный метод обнуляет счётчик быков и коров.

# **Эксперимент**

Выберем длину числа в четыре символа и проверим правильно подсчёта быков и коров для разных комбинаций. (рис.5)

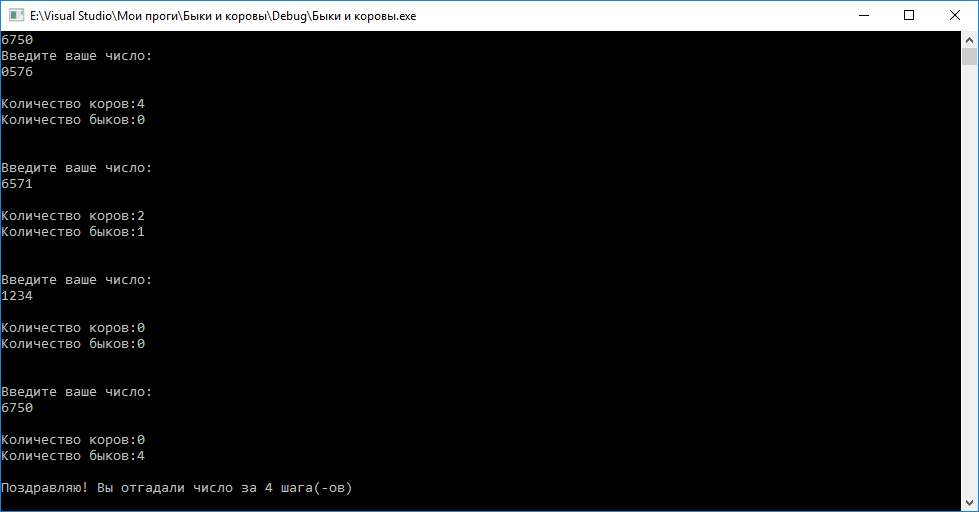


Рис.5

Как видно из рисунка, подсчёт быков, коров и шагов работает верно. Так же правильно функционирует определение условия победы. Из всего этого можно сделать вывод, что задача была реализована верно.

# Приложение

#define \_CRT\_SECURElenghtO\_WARNINGS

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <string>

#include <algorithm>

#include <vector>

#include <time.h>

#include <Windows.h>

using namespace std;

class BullsAndCows

{

int n;

int cows;

int bulls;

vector <int> Player;

vector <int> Chislo;

public:

BullsAndCows()

{

n = 0;

cows = 0;

bulls = 0;

}

void SetLenght(int lenght)

{

n = lenght;

}

vector<int> RandomNumber(int n)

{

vector<int> tmp(10);

srand(time(0));

for (int i = 0; i < 10; i++)

tmp[i] = i;

random\_shuffle(tmp.begin(), tmp.end());

for (int i = 0; i <= n; i++)

Chislo.push\_back(tmp[i]);

return Chislo;

}

bool GetNumber(string igrok)

{

if (n == igrok.size())

{

for (int i = 0; i < igrok.size(); i++)

{

char tmp[2];

tmp[0] = igrok[i];

Player.push\_back(atoi(tmp));

}

return true;

}

else

return false;

}

int CheckCows()

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if ((Chislo[i] == Player[j]) && (i != j))

cows++;

}

}

return cows;

}

int CheckBulls()

{

for (int i = 0; i < n; i++)

{

for (int j = i; j < n; j++)

{

if (Chislo[i] == Player[i] && (i == j))

bulls++;

}

}

return bulls;

}

bool CheckWin(bool BullsAndCowsWin)

{

if (bulls == n)

{

BullsAndCowsWin = true;

}

return BullsAndCowsWin;

}

void ClearCowsBulls()

{

cows = 0;

bulls = 0;

Player.clear();

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

BullsAndCows BullsAndCows;

int lenght, \_steps;

string(menu);

bool BullsAndCowsWin = false;

string num;

cout << "Здравствуйте, это игра 'Быки и коровы'" << endl;

cout << "Ваша задача - отгадать число посредством ввода чисел" << endl;

cout << "Если цифра совпадает, но стоит на другом месте - это корова, если стоит на том же месте - бык" << endl;

cout << endl;

cout << "Начнём? (Да/Нет)" << endl;

cin >> menu;

if (menu == "Да" || menu == "ДА" || menu == "да")

{

system("cls");

\_steps = 0;

cout << "Введите длину числа:" << endl;;

cin >> lenght;

system("cls");

for (int i = 0; i < lenght; i++)

cout << BullsAndCows.RandomNumber(lenght)[i];

BullsAndCows.SetLenght(lenght);

m1:

cout << endl;

cout << "Введите ваше число:" << endl;

cin >> num;

cout << endl;

if (BullsAndCows.GetNumber(num) == 1)

{

\_steps++;

cout << "Количество коров:" << BullsAndCows.CheckCows() << endl;

cout << "Количество быков:" << BullsAndCows.CheckBulls() << endl;

cout << endl;

if (BullsAndCows.CheckWin(BullsAndCowsWin) == true)

{

cout << "Поздравляю! Вы отгадали число за " << \_steps << " шага(-ов)" << endl;

cout << endl;

system("pause");

system("cls");

return 0;

}

else

{

BullsAndCows.ClearCowsBulls();

goto m1;

}

}

}

else

{

return 0;

}

}