**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ**

**ПО НИЖЕГОРОДСКОЙ ОБЛАСТИ**

**УНИВЕРСИТЕТ ИМ.Н.И.ЛОБАЧЕВСКОГО**

**ИНСТУТ ИТММ.**

**Отчет по лабораторным работам.**

**Выполнил:**

студент группы 1703-2

Лугин М.Д.

**Проверил:**

доцент кафедры МОСТ

Сысоев А.В.

Нижний Новгород

2018

**Содержание:**

Программа №13

1.1 Постановка задачи3

1.2 Руководство пользователя4

1.3 Руководство программиста 5

1.4 Эксперимент6

Список литературы7

**Программа №2**8

2.1 Постановка задачи8

2.2 Руководство пользователя9

2.3 Руководство программиста 10

Список литературы11

**Программа №312**

3.1 Постановка задачи12

3.2 Руководство пользователя13

3.3 Руководство программиста 15

3.4 Эксперимент16

Список литературы17

**Программа №418**

4.1 Постановка задачи18

4.2 Руководство пользователя19

4.3 Руководство программиста 22

Список литературы24

**Программа №525**

5.1 Постановка задачи25

5.2 Руководство пользователя26

5.3 Руководство программиста 30

Список литературы32

**Программа №633**

6.1 Постановка задачи33

6.2 Руководство пользователя34

6.3 Руководство программиста 35

Список литературы36

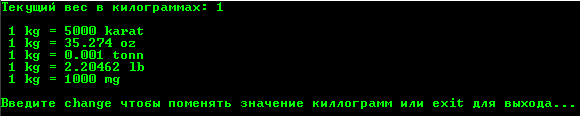
**Приложение37**

**Программа №1**

* 1. **Постановка задачи**
* Разработать класс **Конвертер весов**.
* Класс должен хранить вес в килограммах и предоставлять методы по его преобразованию в другие единицы измерения (например, аптечный фунт, тройская унция, пуд, …).
* Класс должен предоставлять операции: 1) установить текущий вес в килограммах, 2) узнать текущий вес в килограммах, 3) узнать текущий вес в выбранной единице измерения (из списка поддерживаемых).
* Класс должен содержать все необходимые конструкторы, оператор присваивания, а также «уметь» выводить себя на консоль.

**1.2 руководство пользователя**

**1.2.1** Начальный экран

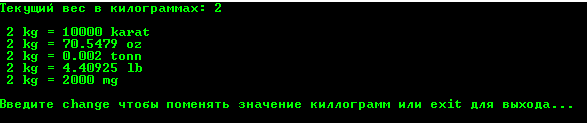
****

**1.2.2** Введите “change” или “exit”

**1.2.3** Если ты ввел “change”, то ты молодец, и можешь поменять значение килограмм



**1.2.4** Наслаждайся полученной информацией



* 1. **Руководство программиста**

Класс **KON** содержит методы , которые реализуют перевод величин веса из килограмм в караты, миллиграммы, тонны, фунты, унции и всё.

Методы возвращающие значение в соответствующих величинах:

**double Karat()**

**double mg()**

**double Tonn()**

**double lb()**

**double oz()**

**double GetKG()**

Метод изменяющий значение :

**double SetKG(double kg)**

Конструктор универсальный:

KON (double kg = 1.0)

{

this -> kg = kg;

}

**1.4 Эксперимент**

В результате эксперимента, показанного на рис.1 и рис. 2, мы видим, что в итоге выводится вес преобразованный из килограммов в другие единицы измерения. Программа функционирует и работает правильно.



Рис. 1

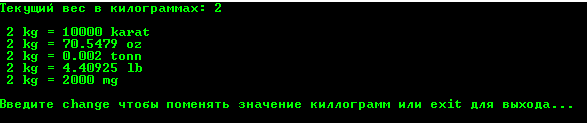


Рис. 2

**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

**Программа №2**

**2.1 Постановка задачи**

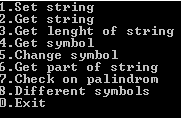
* Разработать класс **Строка**.
* Класс должен хранить строку символов произвольной длины (от 0 до 40).
* Класс должен предоставлять следующие операции: 1) задать строку, 2) узнать длину строки, 3) получить символ строки по его индексу, 4) изменить символ строки по заданному индексу, 5) выделить подстроку из строки, 6) проверить, является ли строка палиндромом, 7) найти, сколько разных символов латинского алфавита содержится в строке.
* Класс должен содержать все необходимые конструкторы, деструктор, оператор присваивания, а также «уметь» выводить себя на консоль.

**2.2 руководство пользователя**

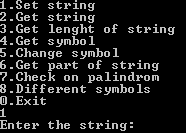
**2.2.1** При запуске введите вашу строчку



**2.2.2** Выбирайте пункт меню



**2.2.2.1** Изменение строки



**2.2.2.2** Получение строки

**2.2.2.3** Получение количества элемента

**2.2.2.4** Получение определенного символа

**2.2.2.5** Изменение символа

**2.2.2.6** Получение подстроки

**2.2.2.7** Проверка на палиндром

**2.2.2.8** Определение сколько разных символов латинского алфавита содержится в строке

**2.2.2.0** Выход из программы

**2.3 руководство программиста**

Class Line

int length – длина строки

char \*string – указатель на строку

Содержит 2 конструктора:

Line() – выделяет память под строку

Line(const Line &obj2) – конструктор копирования

Деструктор:

~Line() – освобождает память

И такие метода как:

**void SetString(char\* word)**

**char\* GetString()**

**int GetStringLength()**

**bool isPalindrom()**

**char GetSymbol(int index)**

**void ChangeSymbolFromString(int index2, char change)**

**char GetPartOfString(int k)**

**int HowManyDifferentLatinSymbolsInString()**

Так же есть оператор присваивания

**Line& operator=(Line &obj1)**

**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

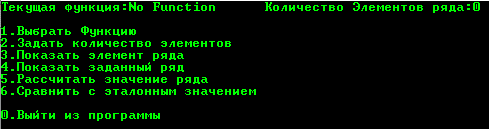
**Программа №3**

**3.1 Постановка задачи**

* Разработать класс **Ряд Тейлора**.
* Класс должен формировать ряд Тейлора для выбранной функции из списка поддерживаемых с заданным числом членов ряда. Минимальный список функций: sin(x), cos(x), exp(x).
* Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
* Класс должен предоставлять следующие операции: 1) задать текущую функцию, 2) узнать текущую функцию, 3) задать текущее число членов ряда, 4) узнать текущее число членов ряда, 5) выдать формулу ряда для выбранной функции, 6) выдать значение заданного члена ряда, 7) рассчитать значение ряда в выбранной точке x, 8) вывести отклонение значения ряда в выбранной точке от эталонного значения текущей функции в данной точке (эталонное значение вычисляется, используя соответствующую функцию из стандартной библиотеки C++).

**3.2 руководство пользователя**

**3.2.1** При открытии программы вы попадаете в главное меню

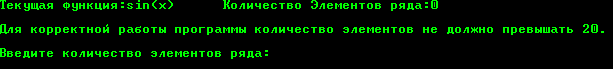


**3.2.2** Выбирайте пункты меню

**3.2.2.1** Задайте функцию из списка



**3.2.2.2** Задайте количество элементов



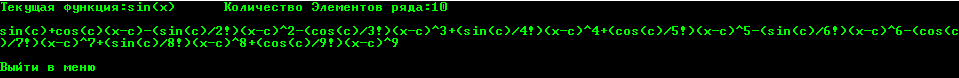
**3.2.2.3.1** Введите номер требуемого элемента ряда Тейлора



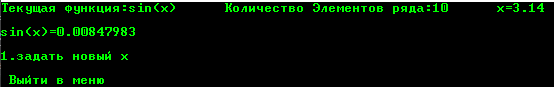
**3.2.2.3.2** Получите результат



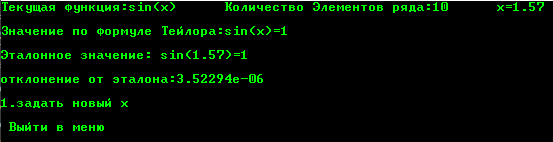
**3.2.2.4** Получение всего ряда Тейлора



**3.2.2.5** Получить результат ряда Мак Лорена



**3.2.2.6** Сравнить с эталонным значением



**3.2.2.0** Выйти из программы

**3.3 руководство программиста**

Класс **Taylor** хранит в себе такие данные как:

**int Function;//текущая функция(1-sin(x) 2-cos(x) 3-exp(x))**

**int number=0;//количество членов ряда**

**double x=0.0;//точка х в формуле Маклорена**

**double const c = 0.0;//точка для формулы Тейлора**

**//bool vr;//вывод(0) или расчет функции(1)**

**long double ans;//результат подсчета**

Cодержит в себе три основных метода:

**void SinTaylor(bool vr = 0, int number = 0, int a = -1)**

**void CosTaylor(bool vr = 0, int number = 0, int a = -1)**

**void ExpTaylor(bool vr = 0, int number = 0, int a = -1)**

которые выполняют основную работу класса – выводят и считают ряд Тейлора.

Данные в классе задаются с помощью методов:

**void SetFunction(int Function)**

**void SetNumber(int number)**

**void SetPoint(double x)**

Данные в классе выводятся с помощью методов:

**long double GetAns()**

**void GetFunction()**

**int GetFunctionint()**

**int GetNumber()**

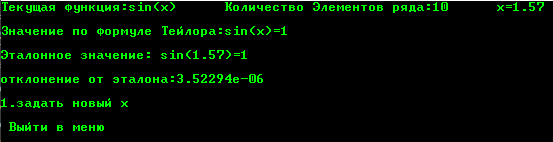
**void GetElemen ( int number=0/\*номер элемента\*/)**

**void GetSequence()**

**double GetPoint()**

**3.4 Эксперимент**

В результате эксперимента, показанного на последующих рисунках, мы видим, что словарь и методы, требуемые в постановке задачи, работают исправно и без тормозов. 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
* http://cppstudio.com/post/446/
* http://purecodecpp.com/archives/2751

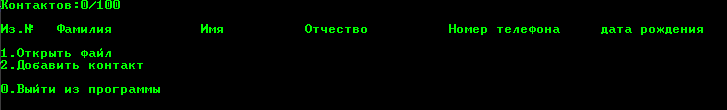
**Программа №4**

**4.1 Постановка задачи**

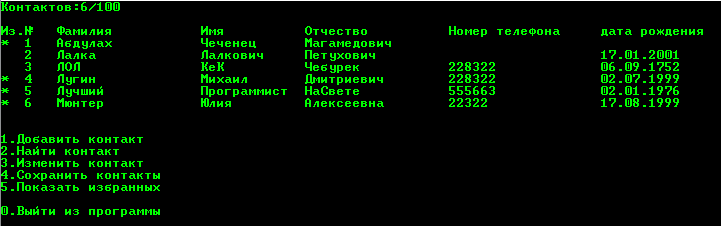
* Разработать класс **Контакты**.
* Класс должен хранить информацию о контактах владельца. Каждый контакт содержит следующие данные: фамилия; имя; отчество; телефон; день рождения (день, месяц, год); признак, относится ли контакт к избранным. Контакты хранятся упорядоченно по фамилии, имени, отчеству. Фамилия, имя, отчество (ФИО) являются обязательными полями. Данные вводятся на русском языке.
* Класс должен содержать необходимые служебные методы.
* Класс должен предоставлять следующие операции: 1) создать новый контакт, 2) изменить выбранный контакт, 3) найти контакт по ФИО, 4) найти контакт по телефону, 5) выдать все контакты на заданную букву, 6) узнать текущее число контактов, 7) внести контакт в список избранных, 8) удалить контакт из списка избранных, 9) выдать все избранные контакты, 10) удалить контакт, 11) сохранить контакты в файл, 12) считать контакты из файла.

**4.2 Руководство пользователя**

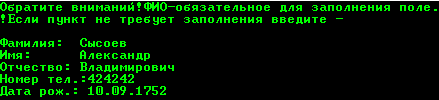
**4.2.1** После открытия программы выбирайте добавить новый контакт или открыть файл



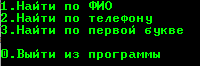
**4.2.2** Далеебудет доступны следующие пункты

****

**4.2.2.1** Добавление контакта происходит пошагово



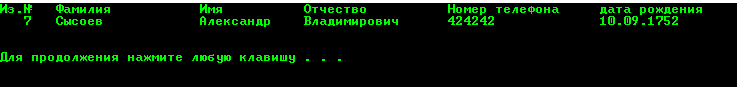
**4.2.2.2** Найти контакты можно тремя способами



В поиске по ФИО вводим любую часть имени



Если такие контакты нашлись то они выводятся на экран

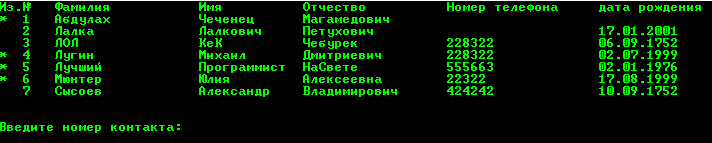


Если не нашлись, то ничего не выведется

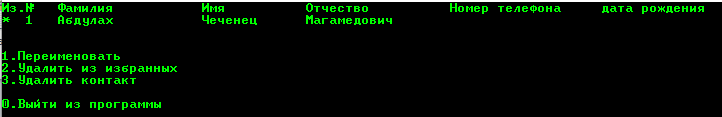




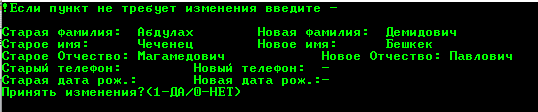
**4.2.3.1** Для изменения контакта введите его уникальный номер

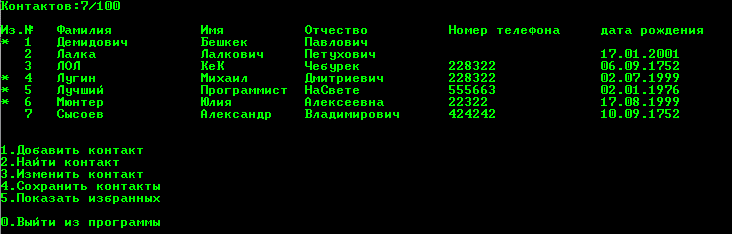
****

**4.2.3.2** Далее откроется меню изменений



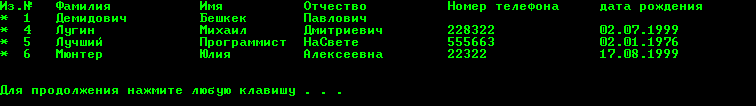
Выбираете и контакт изменен





**4.2.4** При сохранении контактов, они появятся в следующий раз при открытии программы !Внимание! если контакты не были сохранены то они будут утеряны после закрытия программы.

**4.2.5** Показывает избранные контакты



**4.2.0** Выход из программы

**4.3 Руководство программиста**

Класс **Contact** хранит в себе такие данные :

**int count=0;**

**string Surname[Max];**

**string Name[Max];**

**string SecondName[Max];**

**string PhoneNumber[Max];**

**string BD[Max];//день рождения**

**bool FAV[Max];//избранный**

**string ALFAVIT = RUString();**

**string FILE = "Contacts.txt";**

**(MAX = 100), но его можно легко поменять в коде программы.**

Данные задаются с помощью методов:

**void SetSurname(string Surname, int i)**

**void SetName(string Name, int i)**

**void SetSecondName(string SecondName, int i)**

**void SetPhoneNumber(string PhoneNumber, int i)**

**void SetBD(string BD, int i)**

**void SetFAV(bool FAV, int i)**

Данные выводятся с помощью методов:

**void ShowContacts()**

**void ShowContact(int i)**

**void ShowFAV()**

**int GetCount()**

**string GetSurname(int i)**

**string GetName(int i)**

**string GetSecondName(int i)**

**string GetPhoneNumber(int i)**

**string GetBD(int i)**

**bool GetFAV( int i)**

**void UpOut()**

Вся логика находится в методах:

**void AddContact(string Surname,string Name,string SecondName,string PhoneNumber,string BD)**

**void DeleteContact(int i)**

**void FAVplus(int i)**

**void FAVminus(int i)**

**void SetContact(int i ,string a[5])**

**void Sort()**

**void Search(string a)**

**void SearchChar(char a)**

**void SearchPhone(string a)**

Данные методы сохраняют и выводят данные :

**void FILEOpen()**

**void FILESave()**

**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/>

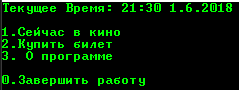
**Программа №5**

**5.1 Постановка задачи**

* Разработать классы **Билетная касса** и **Кинотеатр**.
* Класс **Билетная касса** должен имитировать работу кассы по продаже билетов на киносеансы в многозальном кинотеатре. Считать, что продажа билетов проводится на сеансы в пределах трех дней от текущей даты. Каждый сеанс описывается датой, временем начала сеанса, названием фильма, номером зала, стоимостью билета в зависимости от зоны (VIP и обычная). Для упрощения считать, что покупатель указывает тип зоны и требуемое число билетов, а места выделяются кассой автоматически (при наличии свободных). Зрительные места в каждом зале описываются номером ряда и номером в ряду. Для упрощения считать, что число мест во всех рядах в одном зале одинаково. Продажа билетов на сеанс прекращается через 10 минут после начала сеанса.
* Информация о всех сеансах на ближайшие 30 дней проката хранится в классе **Кинотеатр**. Для каждого зала установлена базовая стоимость билетов (на дневные сеансы – от 12.00 до 18.00). Стоимость билетов на утренние сеансы (до 12.00) составляет 75% от базовой, стоимость билетов на вечерние сеансы (после 18.00) – 150% от базовой. Информация о зрительных местах (свободно/занято) в каждом зале на каждом сеансе также хранится в классе **Кинотеатр**.
* Класс **Билетная касса** должен предоставлять следующие операции: 1) принять данные покупателя: дату, время сеанса, название фильма, номер зала, тип зоны, число мест, 2) проверить наличие требуемого количества свободных мест в требуемой зоне, 3) зарезервировать требуемое количество мест, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, время сеанса, название фильма, номер зала, номер ряда, номер места в ряду).
* Класс **Кинотеатр** должен использоваться для поддержки работы класса **Билетная касса** и может быть разработан в минимально-необходимом варианте.

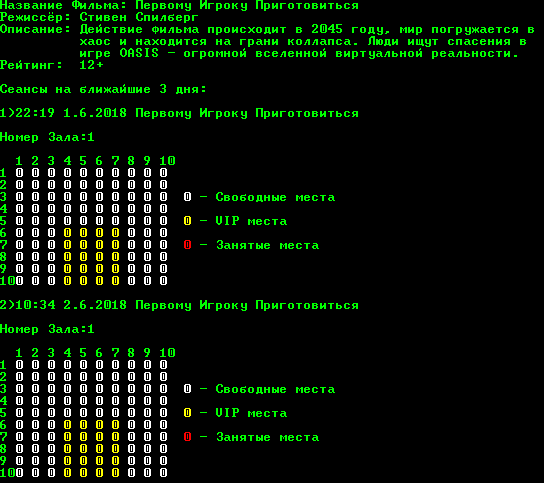
**5.2 Руководство пользователя**

**5.2.1** После открытия программы вы попадаете в главное меню

****

**5.2.2.1** Можно выбрать определённый фильм

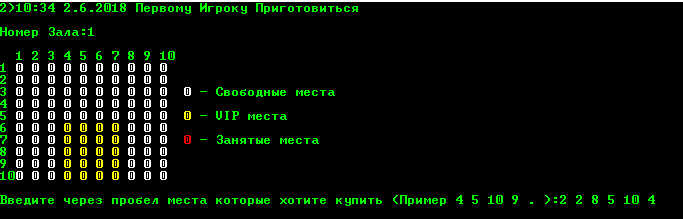


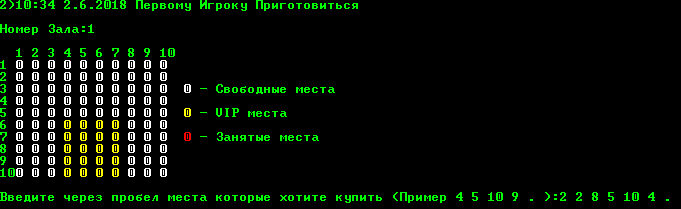
**5.2.2.2** Откроется меню, где можно посмотреть информацию о фильме и о ближайших сеансах

**5.2.2.3** Введите номер сеанса для заказа билетов

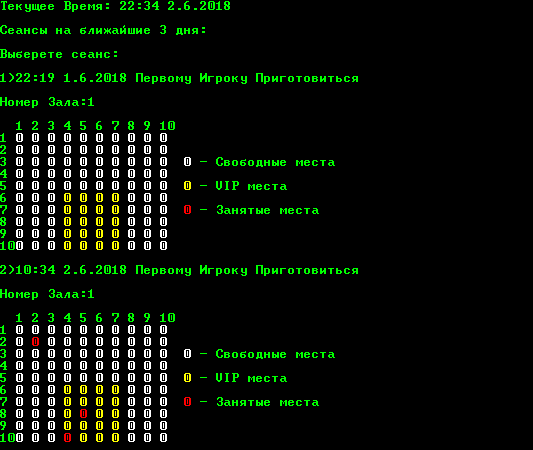


**5.2.2.4** Введите номер места которое хотите купить(сначала ряд, потом место). В конце ведите ‘.’.





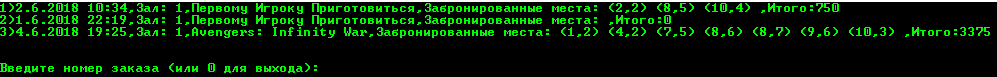
**5.5.3** При выборе пункта 2 в главном меню откроются все сеансы на ближайшее время.



**5.5.4** При выборе пункта 3 будет выведена информация о том как можно получить возможность работать с заказами.



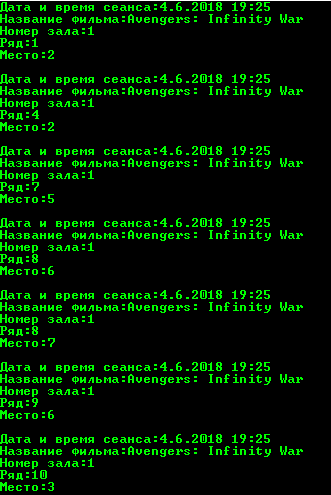
**5.5.5.1** В меню заказов введите номер ордера.



**5.5.5.2** Выберите из двух пунктов



**5.5.5.3.1** Выводит все билеты



**5.5.5.3.1** Отменяет заказ



* В программе есть пасхаЛка))))))))

**5.3 Руководство программиста**

В программе есть два класса

1. Cinema
2. Cash – наследник

Класс **Cinema** хранит информацию о всех сеансах , фильмах и ордерах.

Вся информация хранится в виде вектора определенных структур:

vector<SEANCE> Seance

vector<FILM> Biblioteka

vector<ORDER> Order

Структуры:

struct FILM - хранит информацию о фильме(название, режиссёр и т.д.)

struct DAT – хранит информацию о дате сеанса

struct SEANCE – совокупность всей информации о сеансе(номер зала, дата, фильм, цена билета, массив информации о каждом месте, количество VIP мест)

struct ORDER – хранит информацию о заказе (что было до и после изменений, номер заказа, номер сеанса при заказе, общая стоймость)

Так же этот класс имеет метод void AddFilm(string Name, string Director, string Description, int Rat), который добавляет в библиотеку новый фильм.

Класс **Cash** представляет собой сборник методов.

Методы для создания новых элементов векторов и их удаления:

**void AddSeance(int Month, int Day, int Hour, int Minute)**

**void AddOrder(int Number, vector<int> PlaceH, vector<int> PlaceL,int NumberTicket)**

**void CancerOrder(int Number)**

**void DeleteSeance()**

Методы, участвующие в предыдущих методах:

**bool CheckVIP(int PlaceH, int PlaceL)**

**int FreePlaceNumberOfSeance(int N)**

**int FreeVIPPlaceNumberOfSeance(int N)**

**void SetDate(int n, int Month, int Day, int Hour, int Minute)**

**void SetNumberOfRoom(int n, int Num)**

**bool FreePlace(int n, int PlaceH, int PlaceL)**

**int NumberOfFreePlace(int n)**

Методы, которые выводят информацию:

**int GetNumberFilms()**

**string GetNameFilm(int Number)**

**string GetDirectorFilm(int Number)**

**string GetDescriptionFilm(int Number)**

**int GetRatingFilm(int Number)**

**SEANCE GetSeance(int Number)**

**int GetNumberSeance()**

**int GetNumberOrders()**

**ORDER GetOrder(int Number)**

Методы, выполняющие работу с файлами:

**bool Save()**

**bool Open()**

**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/>

**Программа №6**

**Постановка задачи**

Разработать систему классов и реализовать с ее помощью игру **Быки и коровы**.

Требования (правила).

* 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.

Пример.

* Пусть n = 4.
* Пусть задумано тайное число «3219».
* Игрок ввел число «2310».
* Результат: две «коровы» (две цифры: «2» и «3» — угаданы на неверных позициях) и один «бык» (одна цифра «1» угадана вплоть до позиции).

**6.2 Руководство пользователя**

**6.2.1** Введите количество цифр в числе

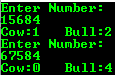




**6.2.2** Введите число заданной длины



**6.2.3** 1 корова и 2 быка означает, что 1 цифра стоит не на своем месте, а 2 на своем, делая выводы из полученной информации, вводите следующее число



**6.2.4** Вы выиграете, если число быков будет равно длине числа



**6.3 Руководство программиста**

В программе есть 1 класс **Game**, в котором содержится:

* Метод **void SetNumberS(int N)** задаетколичество цифр в получаемом числе
* Метод **void AddNumber()** создает рандомное число с неповторяющимися цифрами
* Методы возвращающие информацию **long int GetNumber()**,**int GetNumberS()**.
* Методы выполняющие проверку числа **int CheckBull(long int a)**, **int CheckCow(long int a)**, **bool CheckUnic(long int a)**.

**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/>

**Приложение**

#include<iostream>

#include<string>

#include<ctime>

#include<vector>

#include<math.h>

#include<conio.h>

# include<windows.h>

//#include<>

using namespace std;

class Game

{

int NumberS = 5;

int Difficult = 0;

long int Number = 0;

public:

Game()

{

srand(time(NULL));

}

void SetNumberS(int N)

{

NumberS = N;

}

void SetDifficult(int H)

{

Difficult = H;

}

void AddNumber()

{

int Mass[10];

int n = NumberS;

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

Mass[i] = i;

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

{

int Num = rand() % (10 - i);

if (i == 0) while (!Num)Num = rand() % (10 - i);

Number += Mass[Num] \* pow(10, n - 1);

n--;

Mass[Num] = Mass[9 - i];

}

}

long int GetNumber()

{

return Number;

}

int GetNumberS()

{

return NumberS;

}

int CheckBull(long int a)

{

int Bull = 0;

long int b = Number;

for (int i = 1; i < NumberS + 1; i++)

{

if (a % 10 == b % 10) Bull++;

a = a / 10;

b = b / 10;

}

return Bull;

}

int CheckCow(long int a)

{

int Cow = 0;

long int b = Number;

for (int i = 1; i < NumberS + 1; i++)

for (int j = 1; j < NumberS + 1; j++)

{

int b10 = pow(10, i);

int a10 = pow(10, j);

int b1 = pow(10, i - 1);

int a1 = pow(10, j - 1);

if (b % b10 / b1 == a % a10 / a1) Cow++;

}

return Cow - CheckBull(a);

}

void YouWin()

{

do

{

system("color 04");

cout << "You win!";

system("color 06");

cout << "You win!";

system("color 02");

cout << "You win!";

system("color 01");

cout << "You win!";

system("color 05");

cout << "You win!";

if (\_kbhit()) break;

} while (1);

system("color 0A");

Number = 0;

}

int GetDifficult()

{

return Difficult;

}

bool CheckUnic(long int a)

{

int Mass[10];

long int n = a;

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

Mass[i] = i;

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

{

int j = 0;

for (;j<10-i;j++)

if (Mass[j] == n % 10)

{

Mass[j] = Mass[9 - i];

n /= 10;

j = 0;

break;

}

if (j == 10 - i) return 0;

}

return 1;

}

};

int main()

{

char menu;

Game a;

bool GameOver = 0;

system("color 0A");

Ula:cout << "HI!" << endl << endl;

while (!GameOver)

{

int i = 0;

while (i < 1 || i>10)

{

system("cls");

cout << "Enter number numeral(max 9):";

cin >> i;

}

a.SetNumberS(i);

a.AddNumber();

system("cls");

while (!GameOver)

{

cout << "Enter Number:" << endl;

long int Num = -1, Hum, Ran;

cin >> Num;

Hum = Num;

Ran = a.GetNumber();

if (i == 1) cout << "Cow:" << a.CheckCow(Num) << " " << "Bull:" << a.CheckBull(Num) << endl;

else

{

for (int k = 0; Hum != 0; k++)

{

Hum /= 10;

Ran /= 10;

}

Hum = Num;

for (int k = 0; k < a.GetNumberS(); k++)

{

Hum /= 10;

}

if (Ran > 0 || Hum) cout << "You didn't Enter " << a.GetNumberS() << " numeral!" << endl;

else

{

if (!a.CheckUnic(Num))cout << "You Entered repetitive numeral!" << endl;

else

cout << "Cow:" << a.CheckCow(Num) << " " << "Bull:" << a.CheckBull(Num) << endl;

}

}

if (a.CheckBull(Num) == a.GetNumberS())

{

a.YouWin();

GameOver = 1;

}

}

}

system("cls");

cout << "Restart?:" << endl

<< "1.Yes" << endl

<< "2.No" << endl;

cin >> menu;

switch (menu)

{

case '1': GameOver = 0;

goto Ula;

break;

case '2': exit(0);

}

return 1;

}