**Министерство образования РФ**

**Нижегородский государственный университет имени Н. И. Лобачевского**

**Институт ИТММ**

**Отчет по лабораторным работам**

Выполнил: студент группы 381703-2

Долгополов Д. А.

Проверил: доцент кафедры МОСТ

к.т.н. Сысоев А. В.

Н. Новгород, 2018

Оглавление

[Введение 3](#_Toc515520248)

[Блок 1 4](#_Toc515520249)

[Постановка задачи 4](#_Toc515520250)

[Руководство пользователя 4](#_Toc515520251)

[Руководство программиста 5](#_Toc515520252)

[Блок 2 6](#_Toc515520253)

[Постановка задачи 6](#_Toc515520254)

[Руководство пользователя 6](#_Toc515520255)

[Руководство программиста 8](#_Toc515520256)

[Блок 3 10](#_Toc515520257)

[Постановка задачи 10](#_Toc515520258)

[Руководство пользователя 10](#_Toc515520259)

[Руководство программиста 11](#_Toc515520260)

[Блок 4 13](#_Toc515520261)

[Постановка задачи 13](#_Toc515520262)

[Руководство пользователя 13](#_Toc515520263)

[Руководство программиста 14](#_Toc515520264)

[Блок 5 15](#_Toc515520265)

[Постановка задачи 15](#_Toc515520266)

[Руководство пользователя 15](#_Toc515520267)

[Руководство программиста 17](#_Toc515520268)

[Блок 6 18](#_Toc515520269)

[Постановка задачи 18](#_Toc515520270)

[Руководство пользователя 18](#_Toc515520271)

[Руководство программиста 19](#_Toc515520272)

[Приложение 20](#_Toc515520273)

# Введение

Мной было проделано 6 практических работ, целью которых было научиться создавать классы и их системы и работать с ними.

# Блок 1

# Постановка задачи

Разработать класс **Конвертер 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

При запуске программы пользователь должен будет ввести температуру в градусах Цельсия.

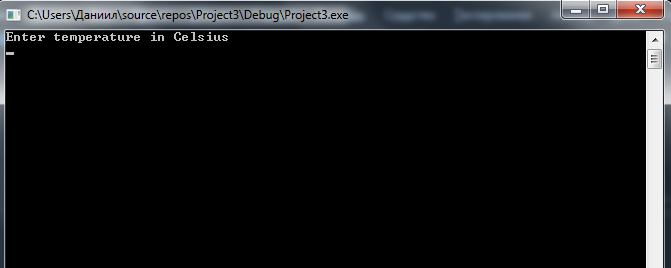


Рис. 1

После ввода программа конвертирует градусы Цельсия в другие единицы измерения

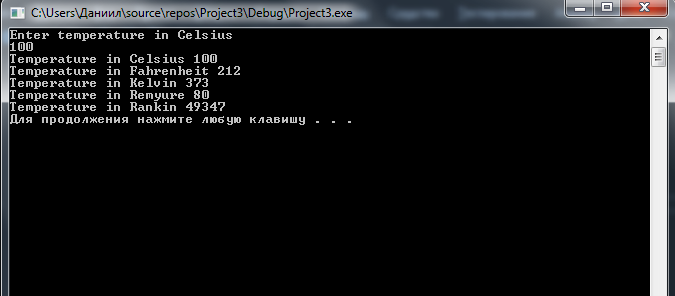


Рис. 2

# Руководство программиста

В классе присутствует поле tC, в котором хранится значение введенной температуры в Цельсиях. Методы get\_tC, get\_tF, get\_tK, get\_tRe, get\_tR соответственно возвращают преобразованные значения температуры в градусах Цельсия, Фаренгейта, Кельвина, Реймура и Ранкина. Метод see\_T выводит значения на консоль. Главная функция main представлена в виде последовательного выполнения методов класса.

# Блок 2

# Постановка задачи

Разработать класс **Календарь 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

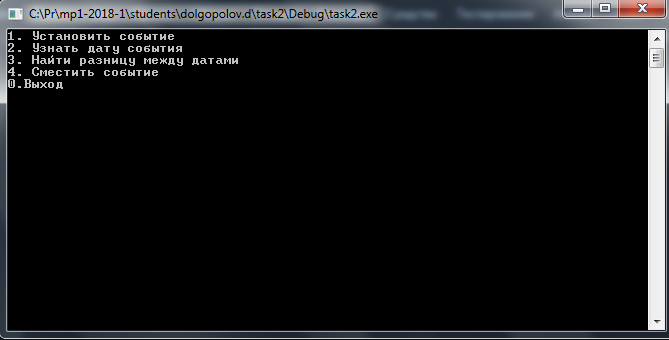
При запуске программы появится окно с меню, где для продолжения пользователь должен ввести нужную цифру. 

Рис. 3

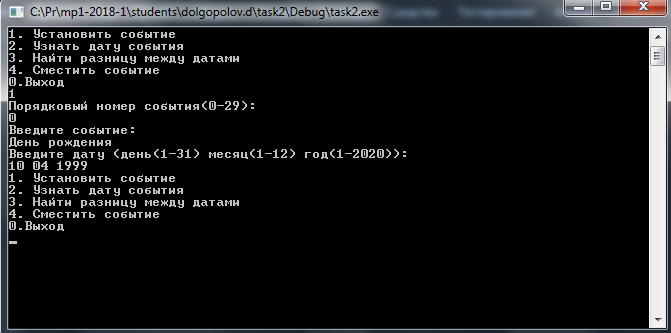
При вводе "1" пользователь должен ввести номер события, наименование события и его дату.

Рис. 4

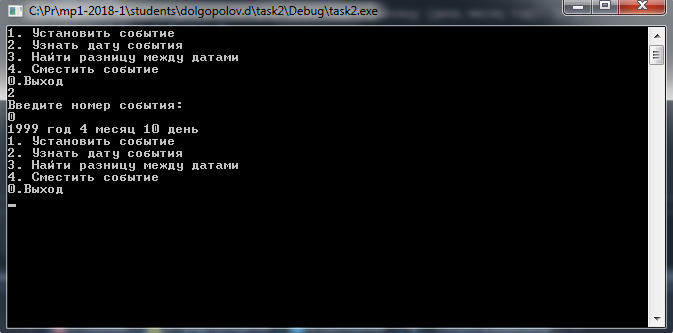
При вводе "2" пользователь должен ввести номер события, и на консоль выведется соответствующая дата

Рис. 5

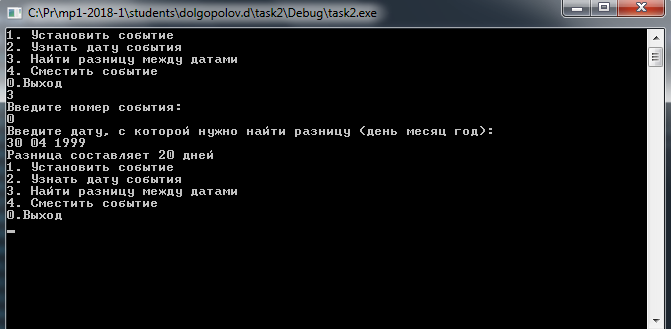
При вводе "3" пользователь должен ввести номер события и дату, с которой хочет найти разницу. 

Рис. 6

При вводе "4" пользователь должен ввести номер события, направление смещения даты и числа дней, месяцев и лет, на сколько сместится событие.

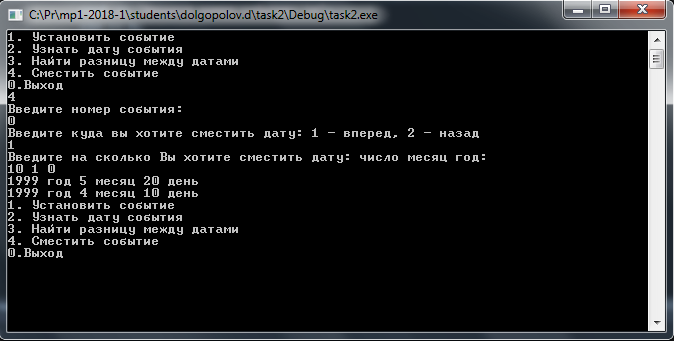
В консоли будут написаны даты после и до смещения.

Рис. 7

# Руководство программиста

В классе содержатся поля d, m, y, в которых хранится соответствующая дата событий, и поле event - названия событий. Метод setEvent выполняет функцию установки даты, getEvent - выводит информацию о событии на консоль. Метод Difference подсчитывает разницу между выбранным пользователем событием и введенной датой и выводит ее на консоль. Метод moveEvent сдвигает дату события вперед или назад на заданную дату. Методы getDay, getMonth, getYear возвращают значение даты, месяца и года выбранного события. Главная функция main выполнена в виде меню, с помощью которого пользователь вызывает нужные методы.

# Блок 3

# Постановка задачи

Разработать класс **Табулятор функции**.

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

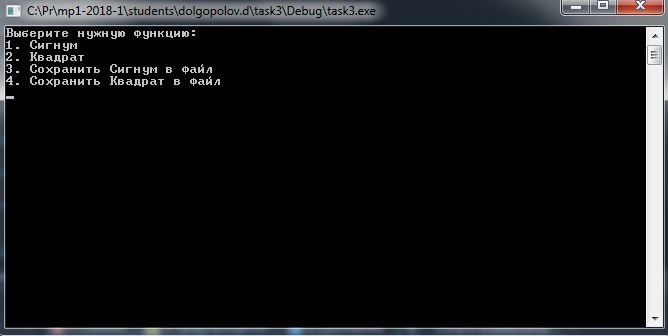
При запуске программы появится консольное окно с меню, где для продолжения пользователь должен ввести нужную цифру.

Рис. 8

При вводе "1" или "2" пользователь выбирает нужную функцию, табуляция которой будет выведена на экран. После выбора нужно ввести число точек табуляции и интервалы.

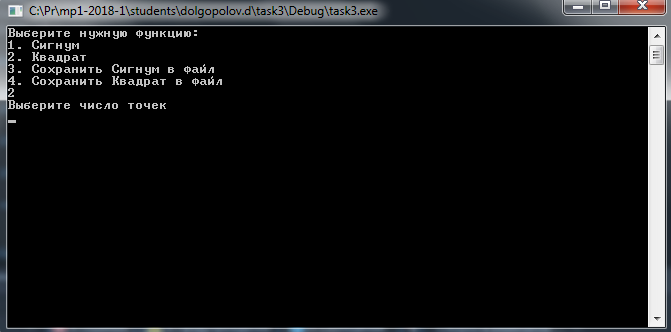


Рис. 9

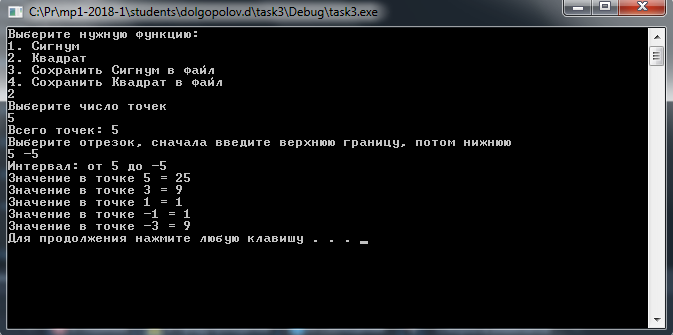


Рис. 10

Когда введены все данные выполняется табуляция функции и программа завершается.

При вводе "3" или "4" и необходимых данных результат табуляции сохранится в текстовый файл "out.txt" в папке с программой.

# Руководство программиста

В программе кроме класса присутствуют две вспомогательные функции - sign и quad, которые возвращают свое значение в заданной точке. В классе находятся поля: a,b, которые хранят значение заданных границ, num - номер выбранной функции, dot - количество точек табуляции, dotvalue и funcvalue - связанные массивы, хранящие значение точки и функции. Также, класс содержит методы: setFunction и getFunc - задание и возвращение номера выбранной функции, setDots и getDots - задание и возвращение количества точек табуляции, setInter, getInter1 и getInter2 - задание и возвращение интервалов, doSign и doQuad - выполнение соответствующих функций и записи результатов в массивы, getDot\_value и getFunc\_value - возвращение значений точек и функций. Главная функция main выполнена в виде меню, с помощью которого пользователь вызывает нужные методы.

# Блок 4

# Постановка задачи

Разработать класс **Шагомер**.

Класс должен хранить историю подсчета шагов владельца. Каждый подсчет описывается датой (день, месяц, год) и интервалом времени (час, минута начала движения, час, минута окончания движения). Подсчет ведется в шагах с точностью до единицы.

Класс должен содержать необходимые служебные методы.

Класс должен предоставлять следующие операции: 1) установить дату начала подсчетов, 2) узнать дату начала подсчетов, 3) задать подсчет, 4) получить информацию о выбранном подсчете, 5) найти среднее число шагов в выбранном месяце или за всю историю наблюдений, 6) найти среднее число шагов в выбранный день недели за всю историю наблюдений, 7) найти максимальное число шагов в день в выбранном месяце или за всю историю наблюдений и дату, когда оно было достигнуто, 8) сохранить историю подсчетов в файл, 9) считать историю подсчетов из файла.

# Руководство пользователя

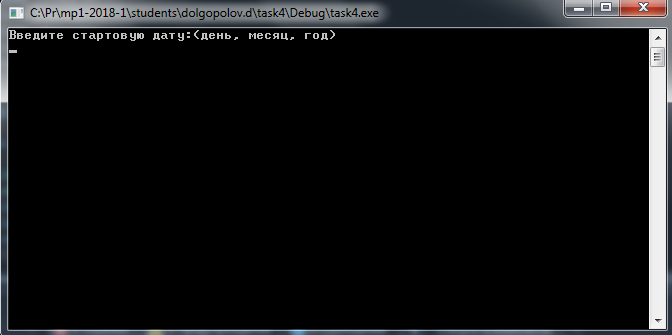
При запуске программы появляется окно, где нужно ввести стартовую дату подсчетов.

Рис. 11

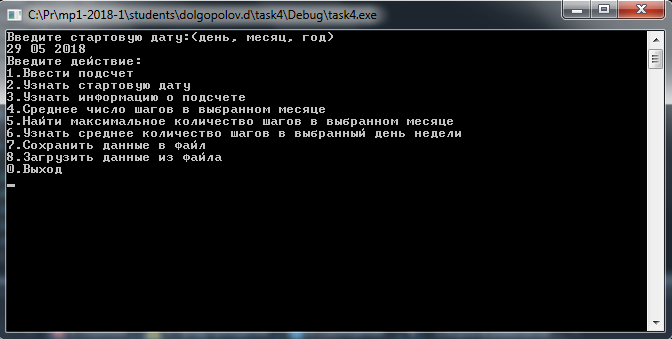
После ввода появляется меню выбора нужной функции.

Рис. 12

При вводе "1" необходимо ввести номер, количество шагов, дату и время подсчета. При вводе "2" будет показана дата начала подсчетов. При вводе "3" необходимо ввести номер нужного подсчета и в появившемся меню выбрать нужную информацию - количество шагов, дату, время начала и конца подсчета. При вводе "4" необходимо ввести нужный месяц, и будет показано среднее количество шагов среди всех подсчетов в выбранном месяце. При вводе "5" необходимо ввести нужный месяц, и будет показано максимальное количество шагов в выбранном месяце и дата, когда оно было достигнуто. При выборе "6" нужно выбрать нужный день недели, и будет показано максимальное количество шагов среди всех подсчетов в выбранном дне. При выборе "7" или "8" данные о подсчетах будут сохранены/загружены в/из файл(-а) "out.txt", находящемуся по пути "C:\Test\".

# Руководство программиста

В классе находится поле Steps, которое хранит количество шагов, структуры: Date - хранит даты подсчетов, Time - хранит время начала и конца подсчетов и StDate - хранит дату начала подсчетов, методы: SetStDate и GetStDate - задание и возвращение стартовой даты, SetCountInfo и GetCountInfo - задание и возвращение информации о подсчетах, AverageStepsMonth - выполнение подсчета среднего количества шагов в определенный месяц и возвращение результата, GetDateFromDate - вспомогательный метод для определения дня недели исходя из его даты, AverageStepsDay - выполнение подсчета среднего количества шагов в определенный день недели и возвращение результата, MaxSteps - определение максимального количества шагов в определенном месяце и возвращение результаты и даты, SaveToFile и GetFromFile - запись данных в файл и чтение данных из файла. Главная функция main выполнена в виде меню, с помощью которого пользователь вызывает нужные методы.

# Блок 5

# Постановка задачи

Разработать классы Железнодорожная касса и Горьковская железная дорога.

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

При запуске программы появляется окно, где нужно ввести информацию о поездке: дату, направление движения (в Москву или Нижний Новгород), имя, фамилия и отчество заказчика, тип поезда - если выбран Фирменный или Скорый, то необходимо выбрать тип вагона (плацкарт, купе или СВ). После ввода появляется меню выбора нужной функции

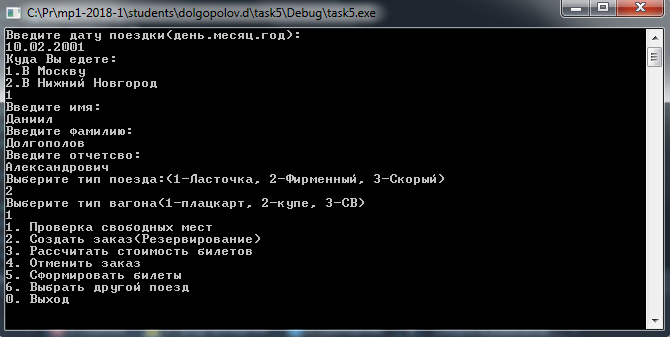


Рис. 13

При вводе "1" будет выведены номера места и их статус - занято или свободно. При вводе "2" пользователю необходимо ввести номер места, которое он хочет зарезервировать. При вводе "3" будет выведена стоимость заказанных билетов в различных поездах и общая стоимость. При вводе "4" пользователю необходимо ввести номер места в заказе, чтобы отменить его резервацию. При вводе "5" будут сформированы билеты вида:

Дата:

\_\_\_\_\_

Номер поезда:

\_\_\_\_\_

Номер вагона:

\_\_\_\_\_

Номер места:

\_\_\_\_\_

Пассажир:

ФИО пассажира

Станция отправки:

\_\_\_\_\_

Станция прибытия:

\_\_\_\_\_

При вводе "6" пользователь должен выбрать другой поезд, в котором будут выбраны места.

# Руководство программиста

В программе находятся две структуры: ticket - хранит информацию о билете и passengers\_data - хранит информацию о данных пассажира; классы: GorRailWay - хранит информацию о местах поездов, в нем находятся поля trainLastoch\_place, trainFirm\_place, trainScor\_place - хранят статус места в виде строки (занято или свободно), occupied\_trainLastoch\_place, occupied\_trainFirm\_place, occupied\_trainScor\_place - хранят статус места в виде true или false (занято или свободно), is\_in\_order\_trainLastoch\_place, is\_in\_order\_trainFirm\_place, is\_in\_order\_trainScor\_place - хранят статус нахождения места в заказе в виде true или false (находится в заказе или не находится), RailWayStation - содержит методы для выполнения поставленных задач; методы класса RailWayStation: SetPassengersData - задание данных пассажира, SetTicketsData - задание данных билета, GetPlaceStatus и GetPlaceStatus1 - возвращение статуса места в виде строки или true или false (занято или свободно), IsInOrder - возвращение статуса нахождения места в заказе в виде true или false (находится в заказе или не находится), ReservePlace - задание месту статус "Занято" и помещение его в текущий заказ, Places - задание и возвращение номеров мест, CancelOrder - задание месту статус "Свободно" и исключение его из текущего заказа. Главная функция main выполнена в виде меню, с помощью которого пользователь вызывает нужные методы.

# Блок 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.

# Руководство пользователя

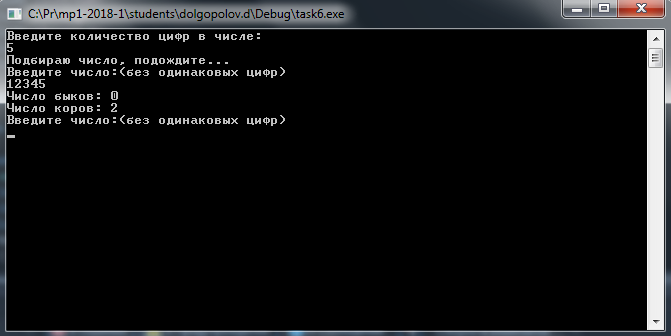
При запуске программы появляется окно, где пользователь должен ввести количество цифр в числе, которое будет загадано. После этого компьютер подбирает число без повторяющихся цифр, и пользователь может его угадывать. 

Рис. 14

# Руководство программиста

В программе находится класс Bulls\_n\_Cows, с помощью которого осуществляется игра. Он содержит поля: numbers - содержит загаданное число, bulls и cows - количество быков и коров; методы: GetNumbers - возвращает загаданное число, Amount\_Of\_Bulls\_n\_Cows - выполняет подсчет количества быков и коров, GetCows\_n\_Bulls - возвращает количество быков и коров. Главная функция main выполнена в виде повторения вводов до тех пор, пока загаданное число не будет отгадано.

# Приложение

**Блок 6**

#include <vector>

#include <iostream>

#include <string>

#include <locale.h>

#include <time.h>

#include <cmath>

using namespace std;

class Bulls\_n\_Cows

{

private:

vector<int> numbers;

int bulls = 0;

int cows = 0;

public:

Bulls\_n\_Cows(int amount)

{

int st = pow(10, amount);

srand(time(0));

int temp = (rand() \* 99) % st;

do

{

numbers.push\_back(temp % 10);

temp = temp / 10;

} while (temp != 0);

}

vector<int> GetNumbers()

{

return numbers;

}

void Amount\_Of\_Bulls\_n\_Cows(vector<int> number)

{

cows = 0;

bulls = 0;

for (int i = number.size() - 1; i >= 0; i--)

if (numbers[i] == number[i])

bulls++;

else

{

for (int k = number.size() - 1; k >= 0; k--)

if ((i != k) && (numbers[i] == number[k]))

cows++;

}

}

int GetCows\_n\_Bulls(int i)

{

switch (i)

{

case 1:

return bulls;

case 2:

return cows;

}

}

};

void main()

{

int amount;

int number;

int temp;

vector<int> digits;

setlocale(LC\_ALL, "Russian");

l:cout << "Введите количество цифр в числе:\n";

cin >> amount;

cout << "Подбираю число, подождите... \n";

k:if (amount <= 0)

{

cout << "Неверный ввод\n";

goto l;

}

Bulls\_n\_Cows ex(amount);

if (ex.GetNumbers().size() != amount)

goto k;

for (int i = ex.GetNumbers().size() - 1; i >= 0; i--)

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

if (ex.GetNumbers()[i] == ex.GetNumbers()[j])

goto k;

start:cout << "Введите число:(без одинаковых цифр)\n";

cin >> number;

temp = number;

digits.clear();

do

{

digits.push\_back(temp % 10);

temp = temp / 10;

} while (temp != 0);

if (digits.size() != amount)

{

cout << "Неверный ввод\n";

goto start;

}

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

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

if (digits[i] == digits[j])

{

cout << "Неверный ввод\n";

goto start;

}

ex.Amount\_Of\_Bulls\_n\_Cows(digits);

cout << "Число быков: " << ex.GetCows\_n\_Bulls(1) << endl << "Число коров: " << ex.GetCows\_n\_Bulls(2) << endl;

if (ex.GetCows\_n\_Bulls(1) != amount)

goto start;

system("pause");

}