МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Факультет Инфокоммуникационных сетей и систем
Кафедра Защищенных систем связи
Дисциплина: «Методы и технологии программирования»

Отчет по лабораторной работе №1:

«Введение в объектно-ориентированное программирование C++»

Выполнил студент гр. ИБС-01:

Гребенников Тимофей

Проверил:

Ерофеев Сергей Анатольевич

доцент кафедры, кандидат физико-математических наук

Санкт-Петербург

Оглавление

Постановка задачи

Требуется разработать базу данных по странам с отражением таких свойств, как название страны, название столицы, численность населения и государственный строй, включающий информацию о форме правления, административно-территориальном устройстве, а также политическом режиме. База данных предусматривает создание класса на языке C++ в среде Microsoft Visual Studio Community 2022 версии 17.1.0.

Техническое задание

- **Список входных данных:**

Название переменной	Назначение	Тип данных
name	название страны	string
capital	название столицы	string
population	численность населения	string
form	форма правления	string
adm	административно-территориальное устройство	string
regime	политический режим	string

- **Список выходных данных:**

Название переменной	Назначение	Тип данных
name	название страны	string
capital	название столицы	string
population	численность населения (млн)	int
politics	государственный строй (форма правления, административно-территориальное устройство, политический режим)	vector<string>

- **Список промежуточных данных:**

Название переменной	Назначение	Тип данных

id	идентификатор экземпляра	int
counter	счётчик инициализации экземпляров	static int

- **Контроль входных данных:**

Для ввода пользователем названия страны и названия столицы контроль входных данных не применяется. Причём присутствует поддержка двух языков: английского и русского. Использование функции *getline()* позволяет осуществлять ввод пробела пользователем для добавления в базу данных стран, имеющих в названии более одного слова (например, «Российская Федерация»).

Ввод пользователем численности населения производится в переменную *population* типа (класса) *string*. Контроль входных данных осуществляется с помощью конструкции *try-catch* на этапе инициализации свойств очередного экземпляра класса. При этом переменной конструктора класса, отвечающей за хранение информации о численности населения, передаётся значение типа *int*. Таким образом, при вводе пользователем корректного значения (целого числа) исполняется ветвь *try* и в свойство *population* класса *country* записывается целочисленное значение. В случае же ввода некорректного значения исполнение программы может пойти по двум сценариям: запись переданного значения в свойство *population* через блок *try* с частичной потерей данных (например, при вводе пользователем числа с плавающей точкой) или запись в свойство *population* значения по умолчанию (0) при возникновении какой-либо ошибки, вызывающей исполнение блока *catch*.

- **Примеры исполнения программы при различных сценариях ввода численности населения:**

ввод пользователя	реакция программы	реализация проверки ввода
150	population = 150	*try{* *obj.Set_All(name, capital, stoi(population), { form, adm, regime });* *}* *catch(const exception& ex){* *obj.Set_All(name, capital, 0, { form, adm, regime });* *}*
99999999999999	population = 0	
150.0	population = 150	
150.9	population = 150	
-150	population = -150	
word	population = 0	

- **Список методов класса:**

метод	описание
string& Get_Name();	Возвращает значение свойства name экземпляра класса country.
void Set_Name(const string& name);	Позволяет изменить свойство name экземпляра класса country.
string& Get_Capital();	Возвращает значение свойства capital экземпляра класса country.
void Set_Capital(const string& capital);	Позволяет изменить свойство capital экземпляра класса country.
long Get_Population();	Возвращает значение свойства population экземпляра класса country.
void Set_Population(const int population);	Позволяет изменить свойство population экземпляра класса country.

vector<string> Get_Politics();	Возвращает значение свойства politics экземпляра класса country.
void Set_Politics(vector<string> arr);	Позволяет изменить свойство politics экземпляра класса country.
void Set_All(const string& name, const string& capital, const int population, const vector<string> politics);	Позволяет изменить все свойства экземпляра класса country кроме служебных.
int GetId();	Возвращает идентификатор экземпляра класса country.
void DecreaseId();	Позволяет понизить на единицу значение идентификатора экземпляра класса country.
static int GetCount();	Возвращает значение счётчика инициализации экземпляров класса country.
static void IncreaseCount();	Позволяет увеличить на единицу значение счётчика инициализации экземпляров класса country.
static void DecreaseCount();	Позволяет уменьшить на единицу значение счётчика инициализации экземпляров класса country.
static void ResetCount();	Позволяет обнулить значение счётчика инициализации экземпляров класса country.
void Print();	Выводит в консоль информацию об экземпляре класса country (список значений свойств с их описанием).

Структурное описание

База данных по умолчанию при запуске программы:

1) Порядковый номер в базе данных: 1

   Название страны: Российская Федерация

   Столица: Москва

   Численность населения: 144 млн. человек

   Государственный строй:

   - Форма правления – республика
   - Административно-территориальное устройство – федеративное
   - Политический режим – демократический

2) Порядковый номер в базе данных: 2

   Название страны: Украина

   Столица: Киев

   Численность населения: 44 млн. человек

   Государственный строй:

   - Форма правления – республика
   - Административно-территориальное устройство – унитарное
   - Политический режим - демократический

3) Порядковый номер в базе данных: 3

   Название страны: Беларусь

   Столица: Минск

   Численность населения: 9 млн. человек

   Государственный строй:

   - Форма правления – республика
   - Административно-территориальное устройство – унитарное
   - Политический режим - демократический

Листинг программы

1) Файлы с исходным кодом:

- **main.cpp:**

```
#include <iostream>
#include <vector>
#include <Windows.h>

#include "country.h"
#include "main_menu.h"
#include "data.h"

//#define DEBUG

using namespace std;


int main()  //организация логики переходов между разделами
{
   SetConsoleCP(1251);
   setlocale(LC_ALL, "ru");

   bool worked = true;

#ifdef DEBUG
   vector<country> database;
   for (int i = 0; i < 5; i++)
   {
      country x("x", "x", 12, { "f", "s", "f" });
      database.push_back(x);
   }

   for (int i = 0; i < 5; i++)
   {
      database[i].Print();
   }
#endif // DEBUG создание + вывод экземпляров

   Set_Default_Data();

   while (worked)
   {
      //функция start() возвращает значение, соответвующее
выбранному пользователем разделу:
```

```
switch (Start())
{
   //переход в раздел очистки БД:
   case 0:
   {
      if (ClearWarning()) Clear_Data();
      break;
   }
   //переход в раздел просмотра БД:
   case 1:
   {
      Print_Data();
      break;
   }
   //переход в раздел добавления экземпляра в БД:
   case 2:
   {
      Add_Data_Member();
      AddCompletedBanner();
      break;
   }
   //переход в раздел удаления экземпляра из БД:
   case 3:
   {
      int id;
      extern vector<country> database;

      if (!database.empty())
      {
         id = DeleteSelector();
         if (id == 0)
         {
            DeleteError();
            break;
         }
         else
         {
            if (DeleteWarning(id)) Delete_Data(id);
            break;
         }
      }
      else
      {
         EmptyError();
```

```
                break;
            }
        }
        //переход в раздел изменения экземпляра в БД:
        case 4:
        {
            int id;
            extern vector<country> database;

            if (!database.empty())
            {
                id = ChangeSelector();
                if (id == 0)
                {
                    ChangeError();
                    break;
                }
                else
                {
                    Change_Data(id);
                    break;
                }
            }
            else
            {
                EmptyError();
                break;
            }
        }
        //выход из программы:
        case 27: worked = false;
    }
  }

  return 0;
}
```

- **data.cpp:**

```
#include "data.h"
#include "country.h"

#include <vector>
#include <string>
#include <iostream>
#include <conio.h>
#include <Windows.h>

using namespace std;

//создание массива объектов класса:
vector<country> database;

void Set_Default_Data()
{
   //предварительная очистка массива для избежания багов:
   if (!database.empty()) database.clear();

   //массивы данных для инициализации свойств дефолтных
объектов
   vector<string> names = {"Российская Федерация", "Украина",
"Беларусь"};
   vector<string> capitals = { "Москва", "Киев", "Минск" };
   vector<int> populations = { 144, 44, 9 };
   vector<string> forms = { "республика", "республика",
"республика" };
   vector<string> administrative = { "федеративное", "унитарное",
"унитарное" };
   vector<string> regime = { "демократический",
"демократический", "демократический" };

   //инициализация свойств дефолтных объектов:
   for (int i = 0; i < 3; i++)
   {
      country obj;
      obj.Set_All(names[i], capitals[i], populations[i], {forms[i],
administrative[i], regime[i]});
      database.push_back(obj);
   }
}

bool Print_Data()
```

```
{
   system("cls");

   //вывод БД в случае наличия объектов класса:
   if (!database.empty())
   {
      for (int i = 0; i < database.size(); i++) database[i].Print();
   }
   //вывод БД в случае отсутствия объектов класса:
   else
   {
      cout << endl << "В базе данных нет сведений." << endl <<
endl;
   }

   cout << endl << "Для возврата в главное меню нажмите
Escape.";

   //ожидание нажатия на Escape:
   while (true)
   {
      if (_kbhit())
      {

         switch (_getch())
         {
         case 27: return true;  //Escape
         }
      }
   }
}

void Clear_Data()
{
   bool worked = true;

   system("cls");

   //очистка массива объектов класса и обнуление счётчика:
   database.clear();
   country::ResetCount();

   cout << endl;
   cout << "База данных успешно очищена!" << endl << endl;
```

```
    cout << "Для возврата в главное меню нажмите Escape." <<
endl;

    //ожидание нажатия на клавишу Escape:
    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case 27: worked = false; break;
            }
        }
    }
}

void Add_Data_Member()
{
    system("cls");

    string name;
    string capital;
    string population;
    string form;
    string adm;
    string regime;

    bool worked = true;

    //сохранение полученных от пользователя свойств объекта:
    cout << endl << "Введите название страны: ";
    getline(cin, name);
    cout << endl;

    cout << "Введите название столицы: ";
    getline(cin, capital);
    cout << endl;

    cout << "Введите численность населения (млн): ";
    getline(cin, population);
    cout << endl;

    cout << "Выберите форму правления в стране:" << endl;
```

```
    cout << "1) Республика" << endl;
    cout << "2) Монархия" << endl;
    cout << endl;

    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case '1': form = "республика"; worked = false; break;
            case '2': form = "монархия"; worked = false; break;
            }
        }
    }

    worked = true;

    cout << "Выберите административно-территориальное
устройство страны:" << endl;
    cout << "1) Унитарное" << endl;
    cout << "2) Федеративное" << endl;
    cout << "3) Конфедеративное" << endl;
    cout << endl;

    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case '1': adm = "унитарное"; worked = false; break;
            case '2': adm = "федеративное"; worked = false; break;
            case '3': adm = "конфедеративное"; worked = false; break;
            }
        }
    }

    worked = true;

    cout << "Выберите политический режим в стране:" << endl;
    cout << "1) Демократический" << endl;
```

```
    cout << "2) Авторитарный" << endl;
    cout << "3) Тоталитарный" << endl;
    cout << endl;

    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case '1': regime = "демократический"; worked = false;
break;
            case '2': regime = "авторитарный"; worked = false; break;
            case '3': regime = "тоталитарный"; worked = false; break;
            }
        }
    }

    country obj;

    //попытка инициализации полученных от пользователя
свойств (обработка исключения связанного с некорректным
вводом численности населения):
    try
    {
        obj.Set_All(name, capital, stoi(population), { form, adm, regime
});
    }
    catch(const exception& ex)
    {
        obj.Set_All(name, capital, 0, { form, adm, regime });
    }

    database.push_back(obj);
}

void Change_Data(int id)
{
    id--;
    bool worked = true;

    //метка старт используется для возврата пользователя в
меню изменения:
```

```
start:

system("cls");

cout << endl;
cout << "Выберите поле, которое хотите изменить:" << endl;
cout << endl;
cout << "1) Название страны" << endl;
cout << "2) Название столицы" << endl;
cout << "3) Численность населения" << endl;
cout << "4) Государственный строй" << endl << endl;
cout << "После завершения изменений нажмите Escape для возврата в главное меню." << endl;

//ожидание нажатия на клавишу (выбор пункта из меню изменения):
while (worked)
{
    if (_kbhit())
    {

        switch (_getch())
        {
            case '1':
            {
                string value;
                bool worked_inside = true;

                system("cls");

                cout << endl;
                cout << "Введите новое название страны и нажмите Enter: ";

                getline(cin, value);

                database[id].Set_Name(value);

                cout << endl;
                cout << "Название страны успешно изменено!" << endl << endl;
                cout << "Для возврата в меню изменения нажмите Escape.";
```

```
            while (worked_inside)
            {
                if (_kbhit())
                {

                    switch (_getch())
                    {
                    case 27: worked_inside = false; break;
                    }
                }
            }
            goto start;
        }

        case '2':
        {
            string value;
            bool worked_inside = true;

            system("cls");

            cout << endl;
            cout << "Введите новое название столицы и
нажмите Enter: ";

            getline(cin, value);

            database[id].Set_Capital(value);

            cout << endl;
            cout << "Название столицы успешно изменено!" <<
endl << endl;
            cout << "Для возврата в меню изменения нажмите
Escape.";

            while (worked_inside)
            {
                if (_kbhit())
                {

                    switch (_getch())
                    {
                    case 27: worked_inside = false; break;
```

```
                }
            }
        }
        goto start;
    }

    case '3':
    {
        string value;
        bool worked_inside = true;

        system("cls");

        cout << endl;
        cout << "Введите новую численность населения
страны и нажмите Enter: ";

        getline(cin, value);

        try
        {
            database[id].Set_Population(stoi(value));
        }
        catch (const exception& ex)
        {
            cout << endl;
            cout << "Введено некорректное значение!" << endl
<< endl;
            cout << "Для возврата в меню изменения нажмите
Escape.";
            goto exit_1;
        }

        cout << endl;
        cout << "Численность населения страны успешно
изменена!" << endl << endl;
        cout << "Для возврата в меню изменения нажмите
Escape.";

        exit_1:
        while (worked_inside)
        {
            if (_kbhit())
            {
```

```
                switch (_getch())
                {
                case 27: worked_inside = false; break;
                }
            }
        }
        goto start;
    }

    case '4':
    {
        bool worked_inside = true;
        string form, adm, regime;

        system("cls");

        cout << endl;
        cout << "Выберите новую форму правления в
стране:" << endl;
        cout << "1) Республика" << endl;
        cout << "2) Монархия" << endl;
        cout << endl;

        while (worked_inside)
        {
            if (_kbhit())
            {

                switch (_getch())
                {
                case '1': form = "республика"; worked_inside =
false; break;
                case '2': form = "монархия"; worked_inside =
false; break;
                }
            }
        }

        worked_inside = true;

        cout << "Выберите новое административно-
территориальное устройство страны:" << endl;
        cout << "1) Унитарное" << endl;
```

```
            cout << "2) Федеративное" << endl;
            cout << "3) Конфедеративное" << endl;
            cout << endl;

            while (worked_inside)
            {
               if (_kbhit())
               {

                  switch (_getch())
                  {
                  case '1': adm = "унитарное"; worked_inside =
false; break;
                  case '2': adm = "федеративное"; worked_inside =
false; break;
                  case '3': adm = "конфедеративное";
worked_inside = false; break;
                  }
               }
            }

            worked_inside = true;

            cout << "Выберите новый политический режим в
стране:" << endl;
            cout << "1) Демократический" << endl;
            cout << "2) Авторитарный" << endl;
            cout << "3) Тоталитарный" << endl;
            cout << endl;

            while (worked_inside)
            {
               if (_kbhit())
               {

                  switch (_getch())
                  {
                  case '1': regime = "демократический";
worked_inside = false; break;
                  case '2': regime = "авторитарный";
worked_inside = false; break;
                  case '3': regime = "тоталитарный";
worked_inside = false; break;
                  }
```

```
                }
            }

            worked_inside = true;

            database[id].Set_Politics({ form, adm, regime });

            cout << endl;
            cout << "Параметры государственного строя
страны успешно изменены!" << endl << endl;
            cout << "Для возврата в меню изменения нажмите
Escape.";

            while (worked_inside)
            {
                if (_kbhit())
                {

                    switch (_getch())
                    {
                    case 27: worked_inside = false; break;
                    }
                }
            }
            goto start;
        }

        case 27:
        {
            worked = false;
            break;
        }
    }
  }
}

void Delete_Data(int id)
{
    bool worked = true;

    id--;

    system("cls");
```

```
    //удаление экземпляра в случае наличия единственного экземпляра (чтобы избежать багов out of range):
    if (database.size() == 1)
    {
        database.clear();
        country::ResetCount();
    }
    //удаление экземпляра в случае наличия нескольких экземпляров:
    else
    {
        for (int i = id + 1; i < database.size(); i++)
        {
            database[i].DecreaseId();
        }

        country::DecreaseCount();

        database.erase(database.begin() + id);
    }

    cout << endl;
    cout << "Страна успешно удалена из базы данных!" << endl << endl;
    cout << "Для возврата в главное меню нажмите Escape." << endl;

    //ожидание нажатия на клавишу Escape:
    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case 27: worked = false; break;
            }
        }
    }
}
```

- **country.cpp:**

```
#include "country.h"

#include <string>
#include <iostream>

//#define DEBUG

using namespace std;

//перегрузка конструктора для инициализации всех параметров:
country::country(const string& name, const string& capital, const int population, const vector<string> politics)
{
#ifdef DEBUG
	cout << "Вызвался конструктор" << endl;
#endif // DEBUG


	counter++;
	id = counter;

	this->name = name;
	this->capital = capital;
	this->population = population;

	for (int i = 0; i < 3; i++) this->politics.push_back(politics[i]);
}

//перегрузка конструктора без инициализации параметров:
country::country()
{
	counter++;
	id = counter;

	population = 0;
}

//геттеры и сеттеры для обычных свойств
string& country::Get_Name()
{
	return name;
}
```

```
void country::Set_Name(const string& name)
{
	this->name = name;
}

string& country::Get_Capital()
{
	return capital;
}

void country::Set_Capital(const string& capital)
{
	this->capital = capital;
}

long country::Get_Population()
{
	return population;
}

void country::Set_Population(const int population)
{
	this->population = population;
}

vector<string> country::Get_Politics()
{
	return politics;
}

void country::Set_Politics(vector<string> arr)
{
	for (int i = 0; i < 3; i++) politics[i] = arr[i];
}

void country::Set_All(const string& name, const string& capital,
const int population, const vector<string> politics)
{
	this->name = name;
	this->capital = capital;
	this->population = population;

	for (int i = 0; i < 3; i++) this->politics.push_back(politics[i]);
}
```

```
//метод вывода свойств в консоль:
void country::Print()
{
	cout <<
"**********************************************************
**********************************" << endl;
	cout << "Порядковый номер в базе данных: " << id << endl
<< endl;
	cout << "Название страны: " << name << "\t\t" <<
"Столица: " << capital << endl;
	cout << "Численность населения: " << population << " млн.
человек" << endl;
	cout << "Государственный строй:" << endl;
	cout << " -> Форма правления - " << politics[0] << endl;
	cout << " -> Административно-территориальное
устройство - " << politics[1] << endl;
	cout << " -> Политический режим - " << politics[2] <<
endl;
	cout <<
"**********************************************************
**********************************" << endl << endl;
}

//методы управления статическими и идентификационными
свойствами:
int country::GetId()
{
	return id;
}

void country::DecreaseId()
{
	id--;
}

int country::GetCount()
{
	return counter;
}

void country::IncreaseCount()
{
	counter++;
```

```
}

void country::DecreaseCount()
{
	counter--;
}

void country::ResetCount()
{
	counter = 0;
}

//тест деструктора для изменения static counter при удалении стран из БД
#ifdef DEBUG
country::~country()
{
	cout << "Вызвался деструктор" << endl;
	counter--;
}
#endif // DEBUG деструктор


int country::counter = 0;
```

- **main_menu.cpp:**

```
#include "main_menu.h"
#include "data.h"
#include "country.h"

#include <iostream>
#include <conio.h>
#include <Windows.h>
#include <vector>


//#define DEBUG

using namespace std;


int Start()
{
	system("cls");

	cout << "Меню работы с базой данных:" << endl << endl;
	cout << "-----------------------------------------------" << endl;
	cout << "1) Посмотреть список стран" << endl;
	cout << "2) Добавить страну в список" << endl;
	cout << "3) Удалить страну из списка" << endl;
	cout << "4) Изменить существующую в списке страну" <<
endl;
	cout << "0) Очистить базу данных" << endl;
	cout << "-----------------------------------------------" << endl
<< endl;
	cout << "Выберите действие и нажмите на
соответствующую кнопку на клавиатуре..." << endl << endl;
	cout << "Для выхода из программы нажмите Escape." <<
endl << endl;

   while (true)
   {
      //ожидание нажатия на клавишу:
      if (_kbhit()) // слушатель нажатия на клавишу
      {

         switch (_getch()) // ждёт нажатия на клавишу с
сохранением в буфер
         {
```

```
            case '0': return 0;
            case '1': return 1;
            case '2': return 2;
            case '3': return 3;
            case '4': return 4;
            case 27: return 27;    //Escape
            }
        }
    }
}


bool ClearWarning()
{
    system("cls");

    extern vector<country> database;
    if (!database.empty())
    {
        cout << endl << "Вы уверены, что хотите очистить базу 
данных?" << endl << endl << endl;
        cout << 
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~" << endl;
        cout << "Внимание! После выполнения очистки отменить 
действие будет невозможно." << endl;
        cout << 
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~" << endl << endl;

        cout << "Для подтверждения очистки нажмите Enter." << 
endl << endl;
        cout << "Для отмены и возврата в главное меню нажмите 
Escape.";

        //ожидание нажатия на клавишу:
        while (true)
        {
            if (_kbhit())
            {

                switch (_getch())
                {
                case 13: return true;   //Enter
```

```
            case 27: return false;  //Escape
            }
        }
    }
    else
    {
        cout << endl << "База данных уже очищена." << endl << endl;
        cout << "Для возврата в главное меню нажмите Escape.";

        //ожидание нажатия на клавишу:
        while (true)
        {
            if (_kbhit())
            {

                switch (_getch())
                {
                case 27: return false;  //Escape
                }
            }
        }
    }
}

void AddCompletedBanner()
{
    bool worked = true;

    cout << endl << "Данные успешно добавлены!" << endl << endl;
    cout << "Для возврата на главную страницу нажмите Escape.";

    //ожидание нажатия на клавишу:
    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case 27: worked = false; break;
```

```
            }
        }
    }
}

int DeleteSelector()
{
    extern vector<country> database;

    string IdNumber;

    system("cls");

    cout << endl;
    cout << "Выберите страну, которую хотите удалить из базы данных:" << endl << endl;

    //вывод списка существующих в БД стран
    for (int i = 0; i < country::GetCount(); i++)
    {
        cout << i + 1 << ") " << database[i].Get_Name() << endl;
    }

    cout << endl;
    cout << "Введите порядковый номер страны и нажмите Enter: ";
    getline(cin, IdNumber);

    //проверка на корректность ввода:
    try
    {
        if (stoi(IdNumber) <= country::GetCount() && stoi(IdNumber) > 0)
        {
            return stoi(IdNumber);
        }
        else return 0;
    }
    catch (const exception& ex)
    {
        return 0;
    }
}
```

```
void DeleteError()
{
   bool worked = true;

   system("cls");

   cout << endl;
   cout << "Выбран некорректный порядковый номер!" << endl
<< endl;
   cout << "Для возврата в главное меню нажмите Escape.";

   //ожидание нажатия на клавишу Escape
   while (worked)
   {
      if (_kbhit())
      {

         switch (_getch())
         {
         case 27: worked = false; break;
         }
      }
   }
}

int ChangeSelector()
{
   extern vector<country> database;

   string IdNumber;

   system("cls");

   cout << endl;
   cout << "Выберите страну, информацию о которой хотите
изменить:" << endl << endl;

   //вывод списка существующих в БД стран
   for (int i = 0; i < country::GetCount(); i++)
   {
      cout << i + 1 << ") " << database[i].Get_Name() << endl;
   }

   cout << endl;
```

```
    cout << "Введите порядковый номер страны и нажмите
Enter: ";
    getline(cin, IdNumber);

    //проверка на корректность ввода:
    try
    {
        if (stoi(IdNumber) - 1 < country::GetCount() &&
stoi(IdNumber) > 0)
        {
            return stoi(IdNumber);
        }
        else return 0;
    }
    catch(const exception& ex)
    {
        return 0;
    }
}

void ChangeError()
{
    bool worked = true;

    system("cls");

    cout << endl;
    cout << "Выбран некорректный порядковый номер!" << endl
<< endl;
    cout << "Для возврата в главное меню нажмите Escape.";

    //ожидание нажатия на клавишу Escape:
    while (worked)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case 27: worked = false; break;
            }
        }
    }
}
```

```
void EmptyError()
{
   bool worked = true;

   system("cls");

   cout << endl;
   cout << "Невозможно выполнить действие: база данных пуста!" << endl << endl;
   cout << "Для возврата в главное меню нажмите Escape." << endl;

   //ожидание нажатия на клавишу Escape:
   while (worked)
   {
      if (_kbhit())
      {

         switch (_getch())
         {
         case 27: worked = false; break;
         }
      }
   }
}

bool DeleteWarning(int id)
{
   id--;

   extern vector<country> database;

   system("cls");

   cout << endl << "Вы уверены, что хотите удалить страну <<" << database[id].Get_Name() << ">>?" << endl << endl << endl;

   cout << "Для подтверждения удаления нажмите Enter." << endl << endl;
   cout << "Для отмены и возврата в главное меню нажмите Escape.";
```

```
    //ожидание нажатия на клавишу:
    while (true)
    {
        if (_kbhit())
        {

            switch (_getch())
            {
            case 13: return true;   //Enter
            case 27: return false;  //Escape
            }
        }
    }
}
```

2) **Файлы заголовков:**

- **data.h:**

```
#pragma once

#include "country.h"

void Set_Default_Data();

bool Print_Data();

void Clear_Data();

void Add_Data_Member();

void Change_Data(int id);

void Delete_Data(int id);
```

- **country.h:**

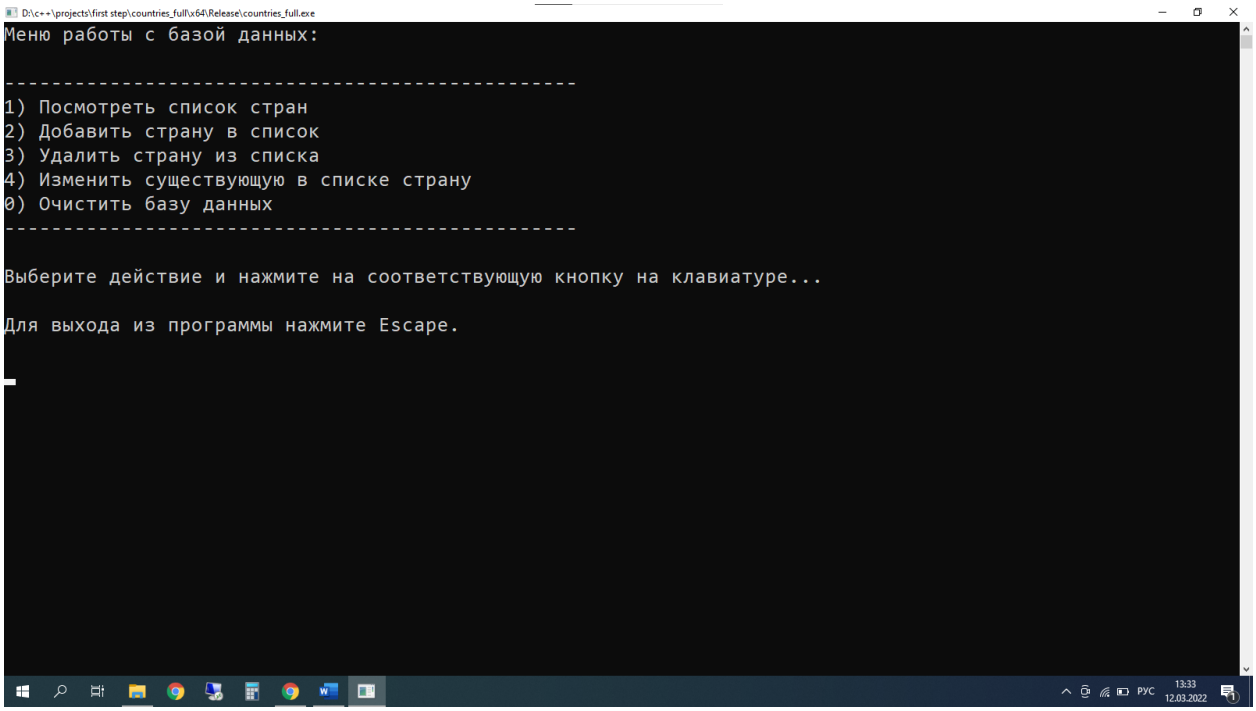
```
#pragma once
#include <string>
#include <vector>

//#define DEBUG

using namespace std;


class country
{
private:
	static int counter;
	int id;

	string name;
	string capital;
	int population;
	vector<string> politics;

public:
	country(const string& name, const string& capital, const int population, const vector<string> politics);

	country();

	string& Get_Name();
	void Set_Name(const string& name);

	string& Get_Capital();
	void Set_Capital(const string& capital);

	long Get_Population();
	void Set_Population(const int population);

	vector<string> Get_Politics();
	void Set_Politics(vector<string> arr);

	void Set_All(const string& name, const string& capital, const int population, const vector<string> politics);

	void Print();
```

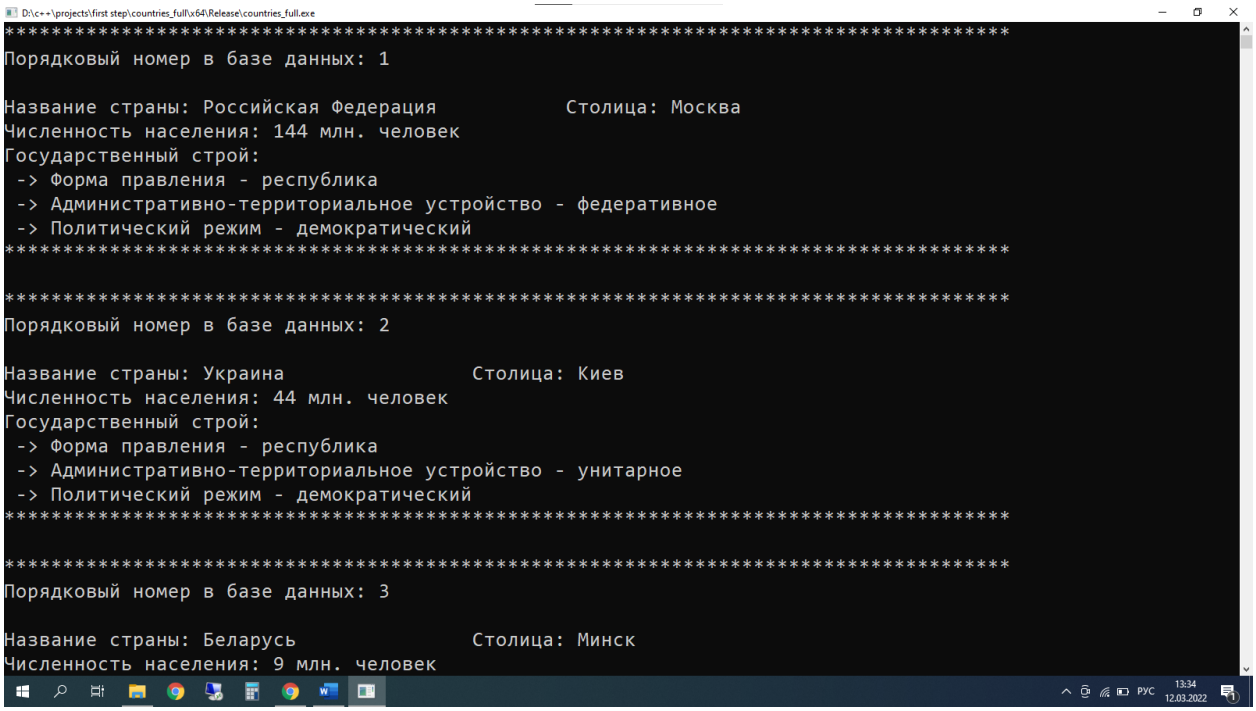
```
        int GetId();

        void DecreaseId();

        static int GetCount();

        static void IncreaseCount();

        static void DecreaseCount();

        static void ResetCount();

#ifdef DEBUG
        ~country();
#endif // DEBUG

};
```

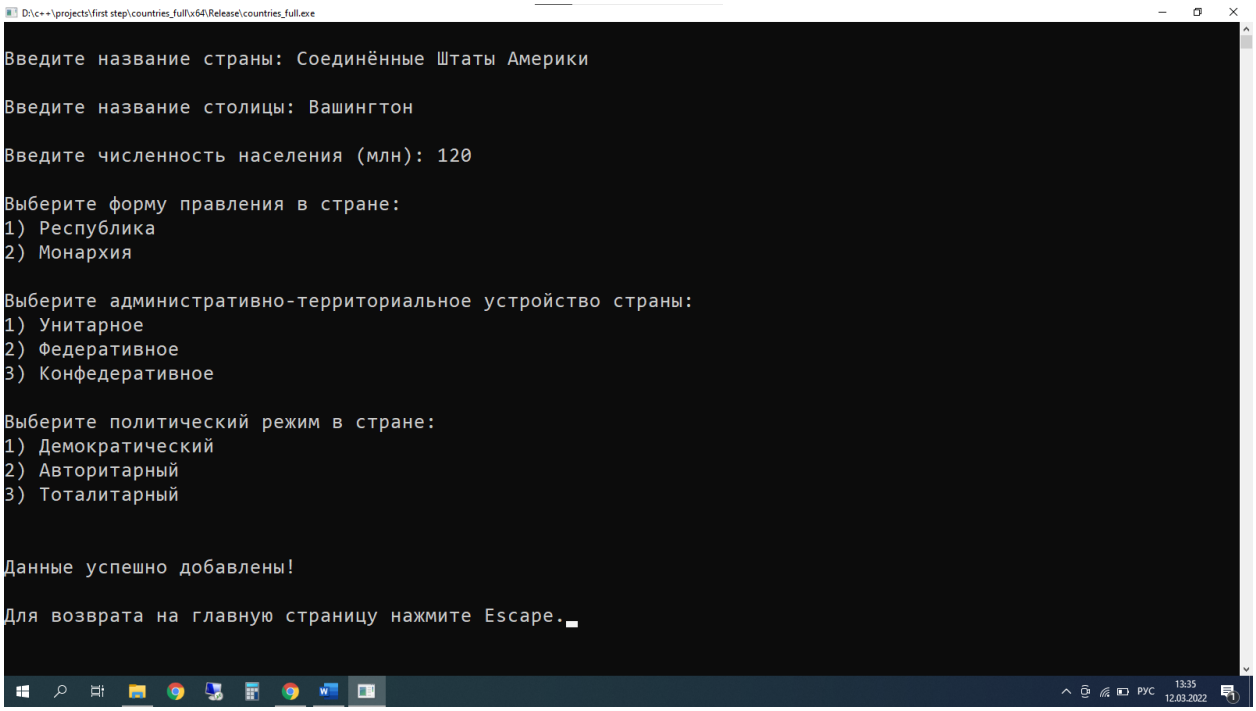
- **main_menu.h:**

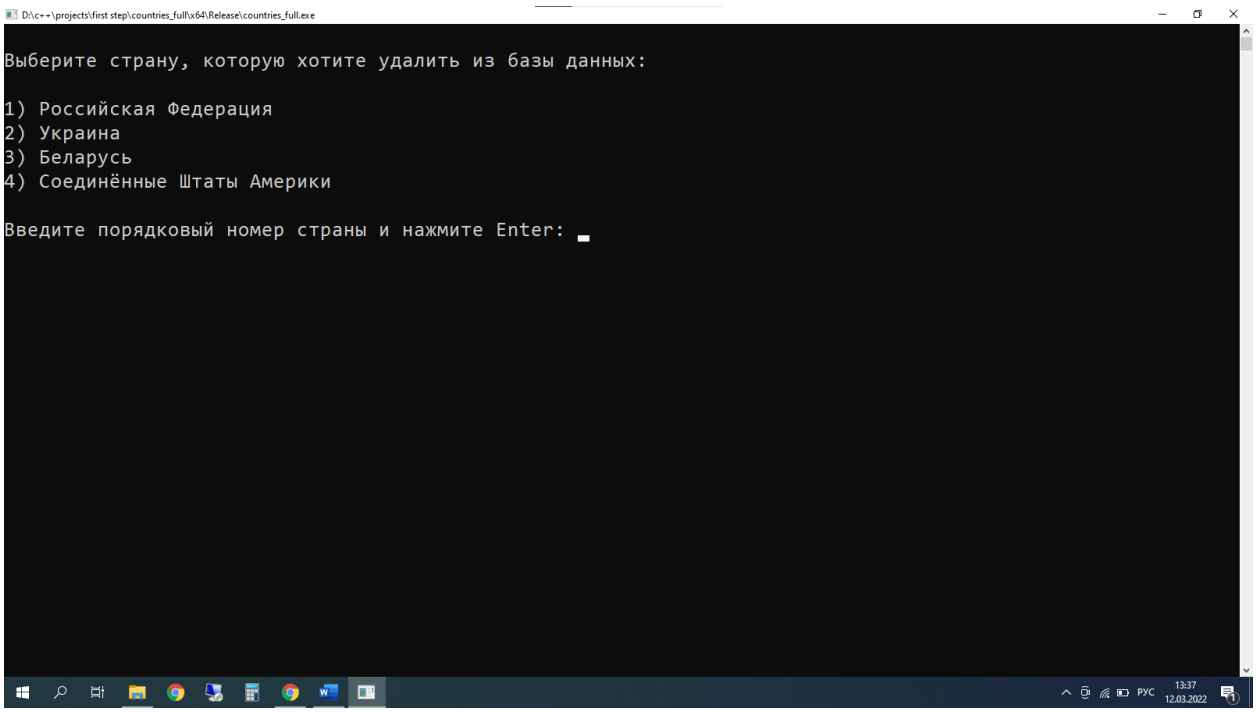
```
#pragma once

int Start();

bool ClearWarning();

void AddCompletedBanner();

int DeleteSelector();

void DeleteError();

int ChangeSelector();

void ChangeError();

void EmptyError();

bool DeleteWarning(int id);
```

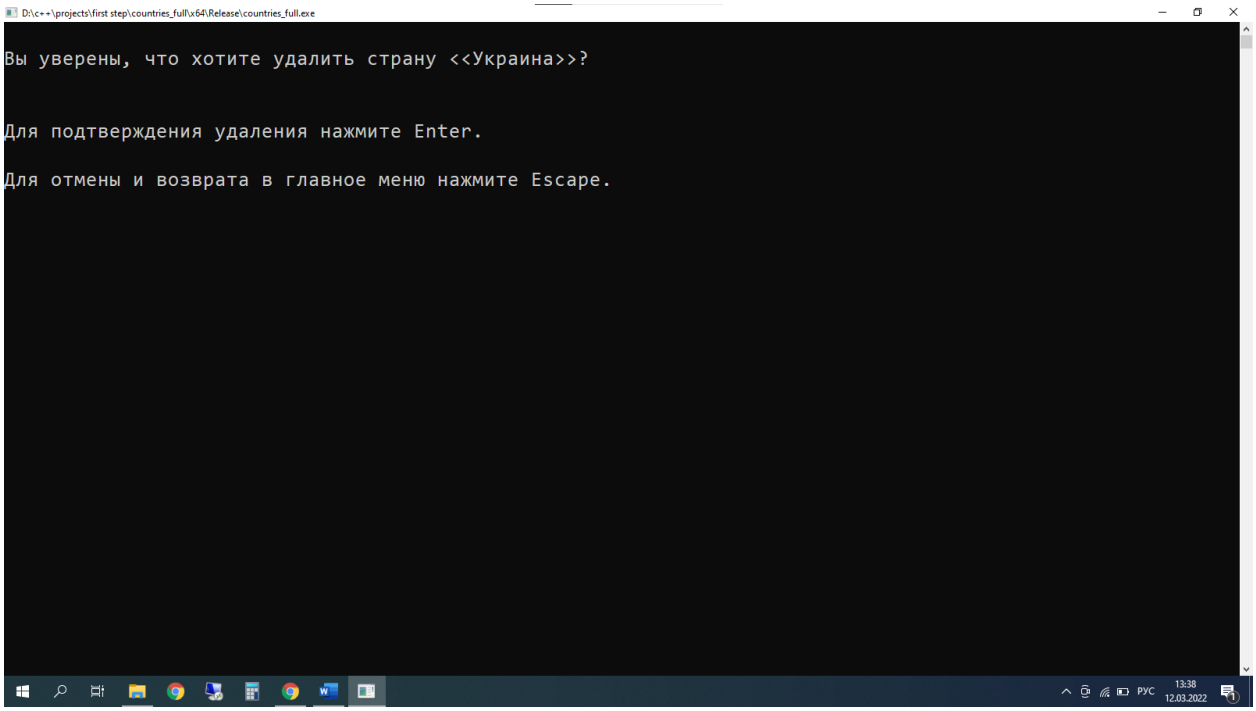
Тестирование

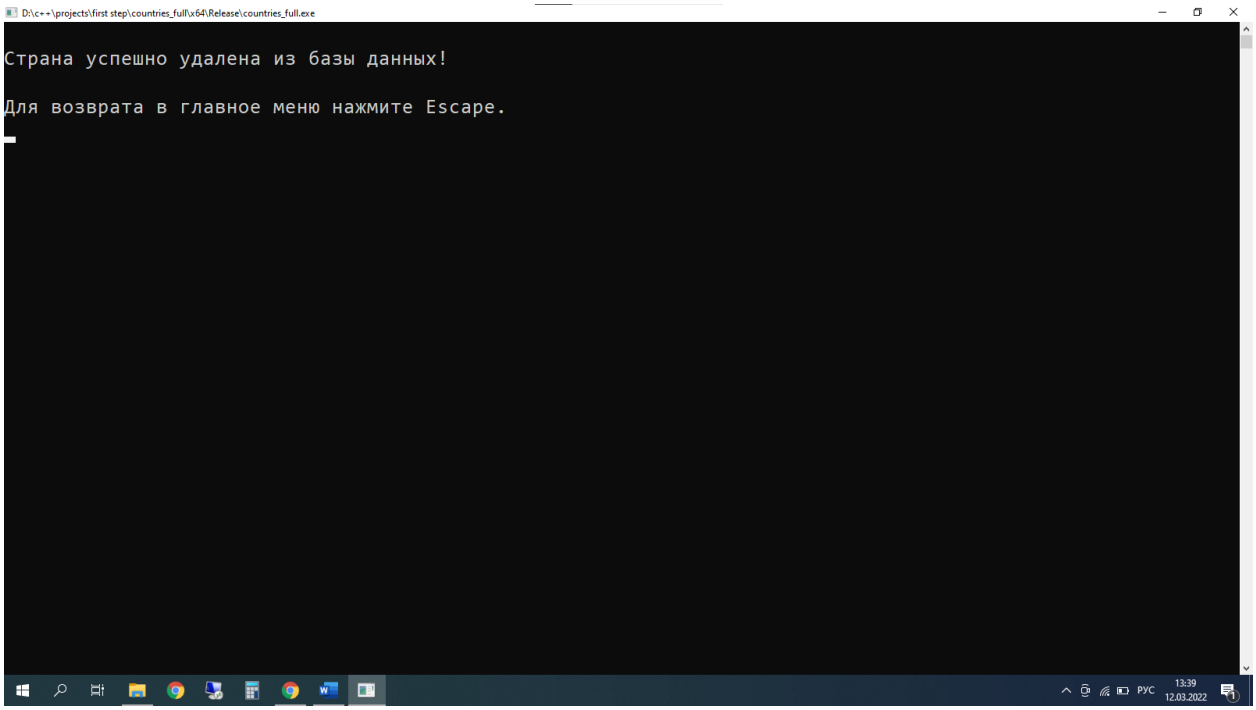


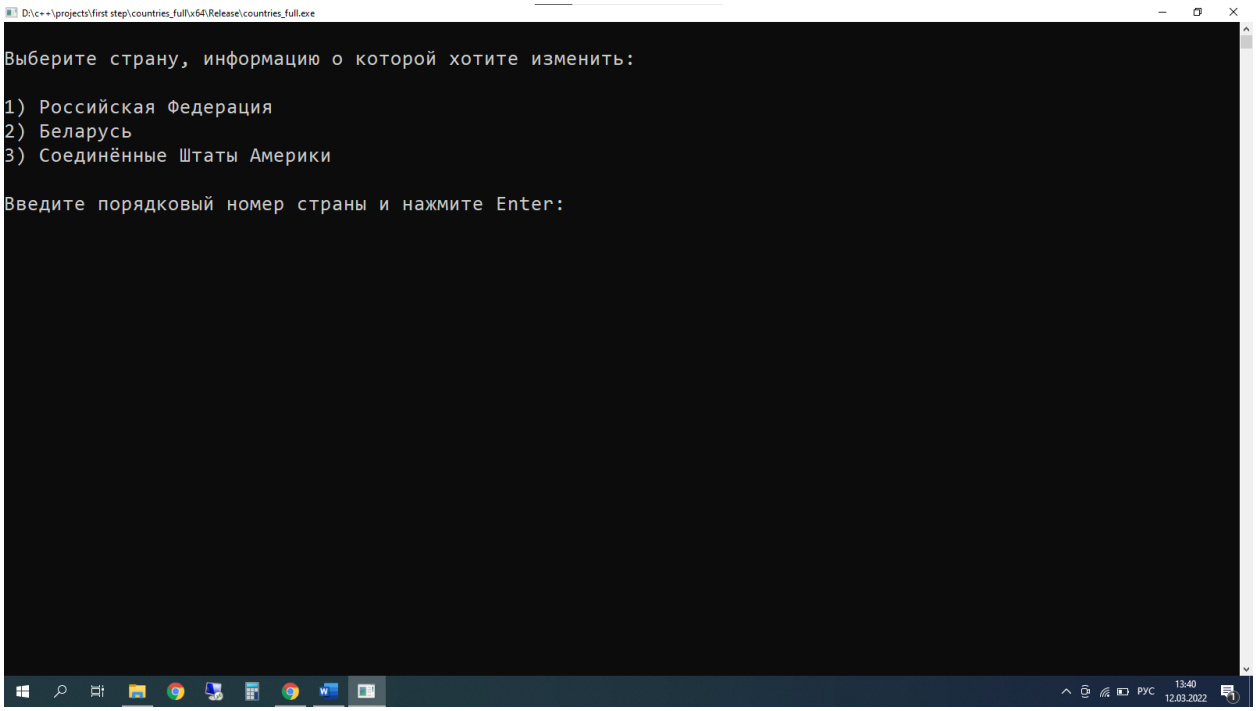


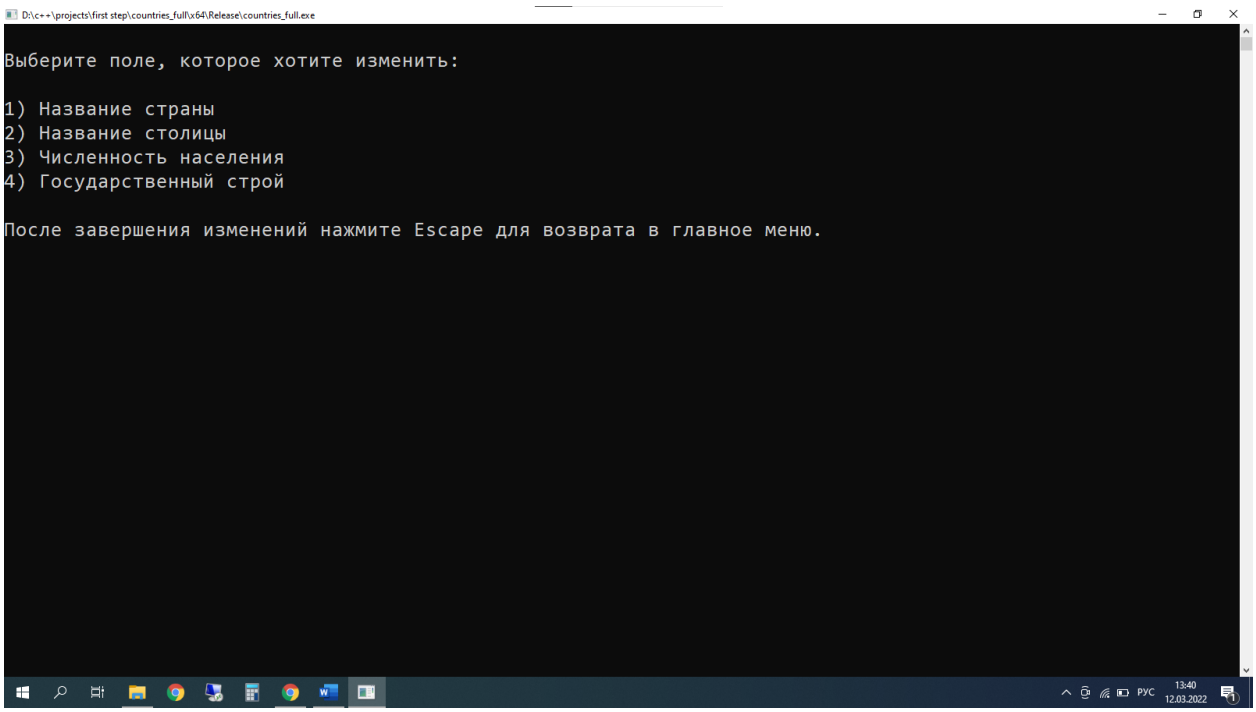

D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Введите название страны: Соединённые Штаты Америки
Введите название столицы: Вашингтон
Введите численность населения (млн): 120
Выберите форму правления в стране:
1) Республика
2) Монархия
Выберите административно-территориальное устройство страны:
1) Унитарное
2) Федеративное
3) Конфедеративное
Выберите политический режим в стране:
1) Демократический
2) Авторитарный
3) Тоталитарный
Данные успешно добавлены!
Для возврата на главную страницу нажмите Escape.
РУС
13:35
12.03.2022

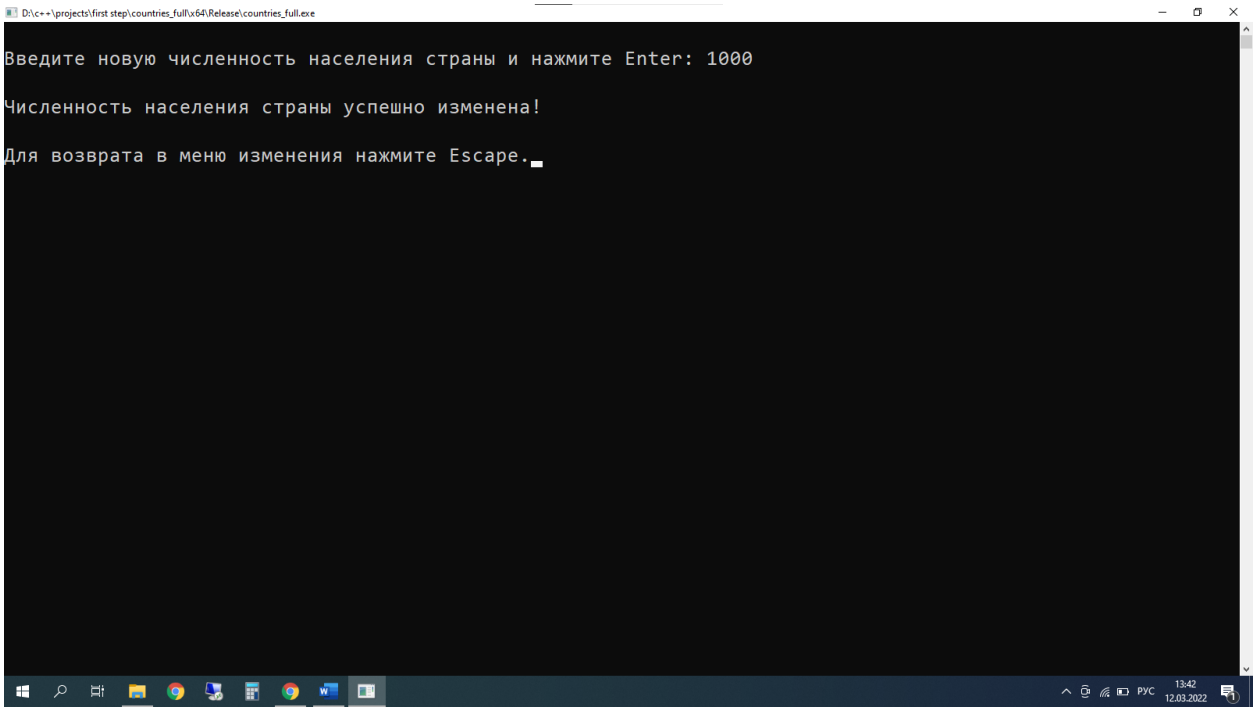

D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Выберите страну, которую хотите удалить из базы данных:
1) Российская Федерация
2) Украина
3) Беларусь
4) Соединённые Штаты Америки
Введите порядковый номер страны и нажмите Enter:
РУС
13:37
12.03.2022

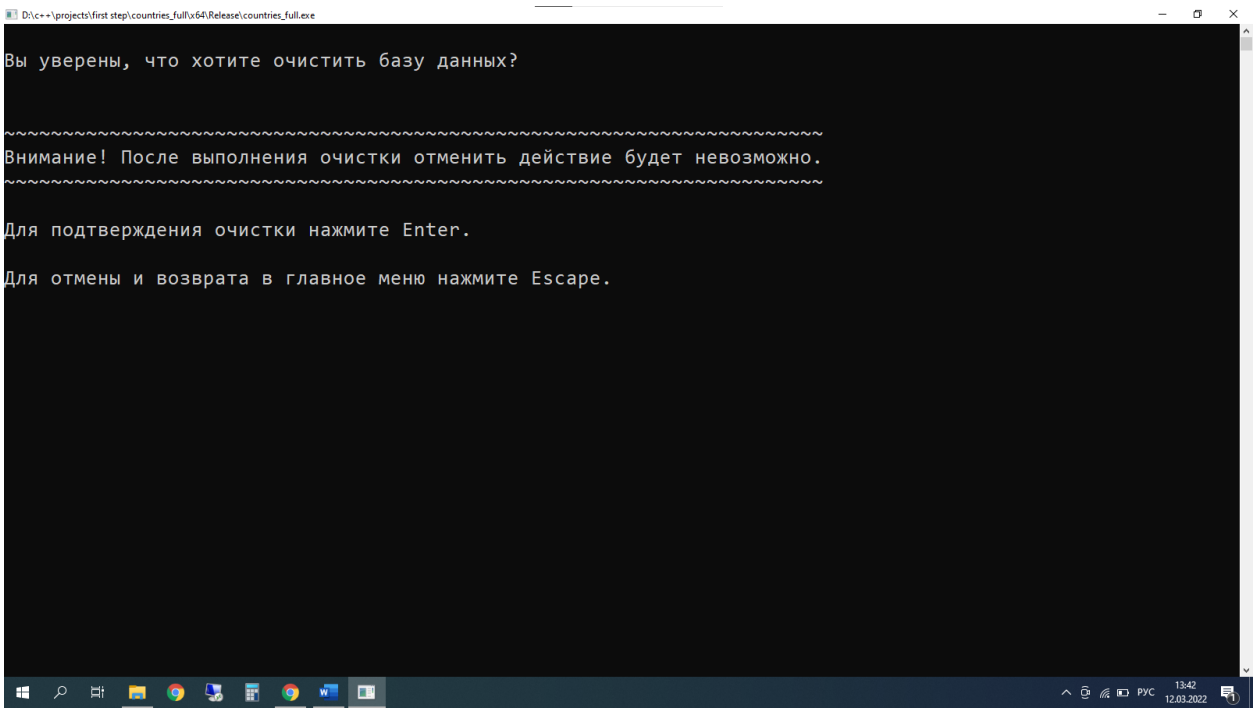

D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Вы уверены, что хотите удалить страну <<Украина>>?
Для подтверждения удаления нажмите Enter.
Для отмены и возврата в главное меню нажмите Escape.
РУС
13:38
12.03.2022

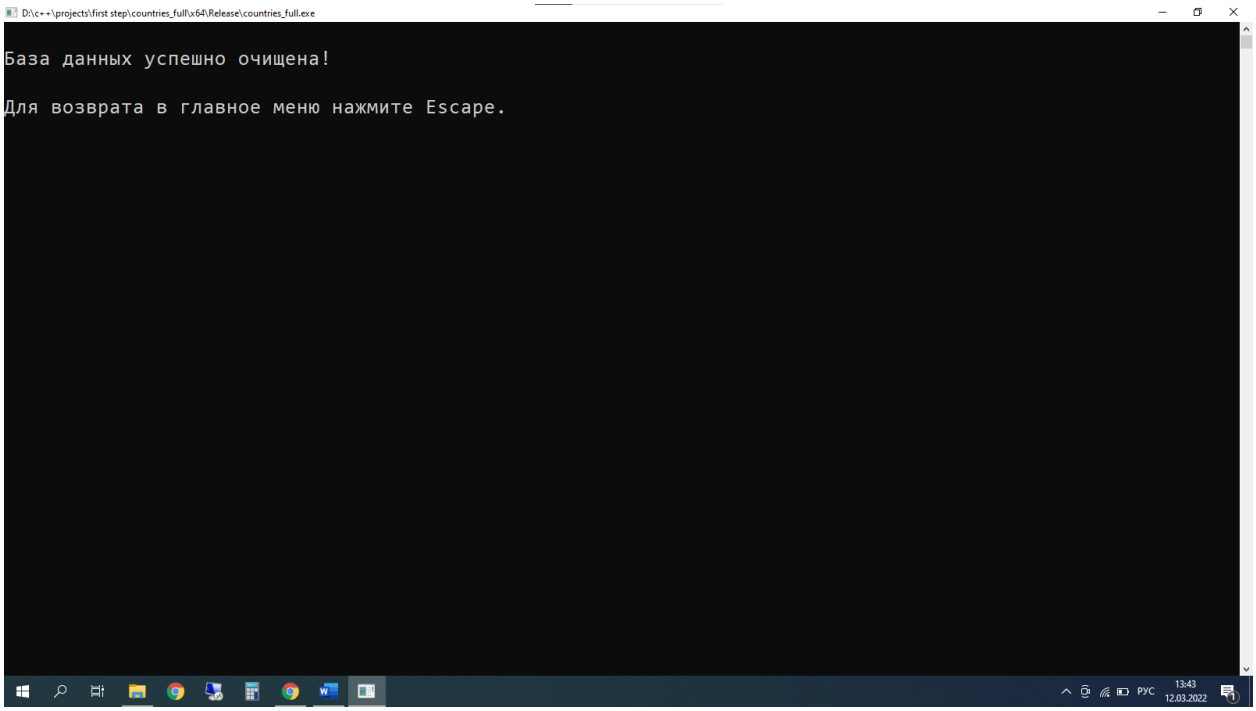

D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Страна успешно удалена из базы данных!
Для возврата в главное меню нажмите Escape.
РУС
13:39
12.03.2022


D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Выберите страну, информацию о которой хотите изменить:
1) Российская Федерация
2) Беларусь
3) Соединённые Штаты Америки
Введите порядковый номер страны и нажмите Enter:


D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Выберите поле, которое хотите изменить:
1) Название страны
2) Название столицы
3) Численность населения
4) Государственный строй
После завершения изменений нажмите Escape для возврата в главное меню.


D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Введите новую численность населения страны и нажмите Enter: 1000
Численность населения страны успешно изменена!
Для возврата в меню изменения нажмите Escape.


D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
Вы уверены, что хотите очистить базу данных?
Внимание! После выполнения очистки отменить действие будет невозможно.
Для подтверждения очистки нажмите Enter.
Для отмены и возврата в главное меню нажмите Escape.


D:\c++\projects\first step\countries_full\x64\Release\countries_full.exe
База данных успешно очищена!
Для возврата в главное меню нажмите Escape.
РУС
13:43
12.03.2022

Вывод

При получении задания по лабораторной работе №1 была поставлена задача написать базу данных, содержащую информацию о странах, с использованием класса на языке C++. Поставленная задача была успешно выполнена в среде Microsoft Visual Studio Community 2022 версии 17.1.0.

Выполнение лабораторной работы помогло ознакомиться с основными конструкциями языка, понять принципы построения классов в C++ и применения объектно-ориентированного программирования в целом.

Написанная в ходе выполнения лабораторной работы программа имеет поддержку пользовательского интерфейса для навигации по базе данных, а также обладает частичным контролем входных данных, что позволяет избежать ошибок при некорректном использовании. Весь функционал протестирован при различных сценариях, программа работает корректно.