



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ОТЧЕТ ОБ УЧЕБНОЙ ПРАКТИКЕ

Тип практики Проектно-технологическая практика

Название
предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Студент группы ИУ6-24Б

ИУ 23.05.2023
(Подпись, дата)

А. И. Мокшина

(И.О. Фамилия)

Руководитель практики

23.05.2023
(Подпись, дата)

Мокшина А.И.
(И.О. Фамилия)

Оценка отлично

2023 г.

З А Д А Н И Е

на учебную практику

по теме Проектирование и реализация программного обеспечения с использованием
структурного и объектного
ПОДХОДОВ

Студент группы ИУ6-24Б

Мокшина Анастасия Игоревна
(Фамилия, имя, отчество)

Тип практики Проектно-технологическая практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание:

Задание 1. Создание программной системы на Object Pascal

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База демографических данных (файл) содержит сведения о населении городов: название города, численность населения, год проведения переписи. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать все демографические сведения, полученные в указанном году.
2. Показать список городов-миллионеров (по данным последней переписи).
3. Определить, сколько раз проводилась перепись населения в указанном городе.
4. Построить график изменения численности населения заданного города по годам.

Задание 2. Создание программной системы с элементарным интерфейсом консольного режима на C++

Выполнить структурную декомпозицию, разработать структурную схему, содержащую не менее 3 подпрограмм, и алгоритмы этих подпрограмм. Реализовать на C++ в консольном режиме. Предусмотреть примитивный интерфейс типа меню, позволяющий выбирать нужную подпрограмму.

Написать программу нахождения корней функции $y=F(x)$ на заданном отрезке $[a,b]$ методом хорд и методом половинного деления отрезка с заданной точностью ξ . Интервал пользователь должен иметь возможность задавать по запросу, функцию для расчета

выбирать из списка функций, а метод поиска корня – из списка методов. Для реализации возможности выбора функции использовать указатель на функцию. В список функций включить не менее 5 функций по своему выбору (лучше такие, которые легко проверять).

Задание 3. Создание программной системы с Qt интерфейсом на C++

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

База демографических данных содержит сведения о населении городов: название города, численность населения, год проведения переписи. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

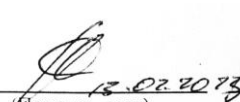
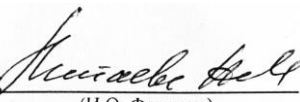
1. Показать все демографические сведения, полученные в указанном году.
2. Показать список городов-миллионеров (по данным последней переписи).
3. Определить, сколько раз проводилась перепись населения в указанном городе.
4. Построить график изменения численности населения заданного города по годам.

Оформление отчета по практике:


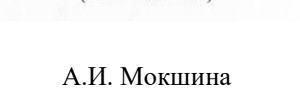
Отчет на 25-35 листах формата А4 должен включать титульный лист, задание (печатать с двух сторон), оглавление, введение, три главы, заключение и список использованных источников. Отдельная глава по каждому заданию должна содержать анализ задания, требуемые чертежи, текст программы, результаты тестирования и выводы.

Дата выдачи задания « 07 » февраля 2023 г.

Руководитель практики

 13.02.2023 
(Подпись, дата) (И.О. Фамилия)

Студент

 13.02.2023 
(Подпись, дата) А.И. Мокшина
(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Задание 1. Создание программной системы на Object Pascal.

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База демографических данных (файл) содержит сведения о населении городов: название города, численность населения, год проведения переписи. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать все демографические сведения, полученные в указанном году.
2. Показать список городов-миллионеров (по данным последней переписи).
3. Определить, сколько раз проводилась перепись населения в указанном городе.
4. Построить график изменения численности населения заданного города по годам.

Цель работы: создание программной системы на Object Pascal с использованием возможностей VCL Lazarus.

Для выполнения работы были разработаны формы приложения:

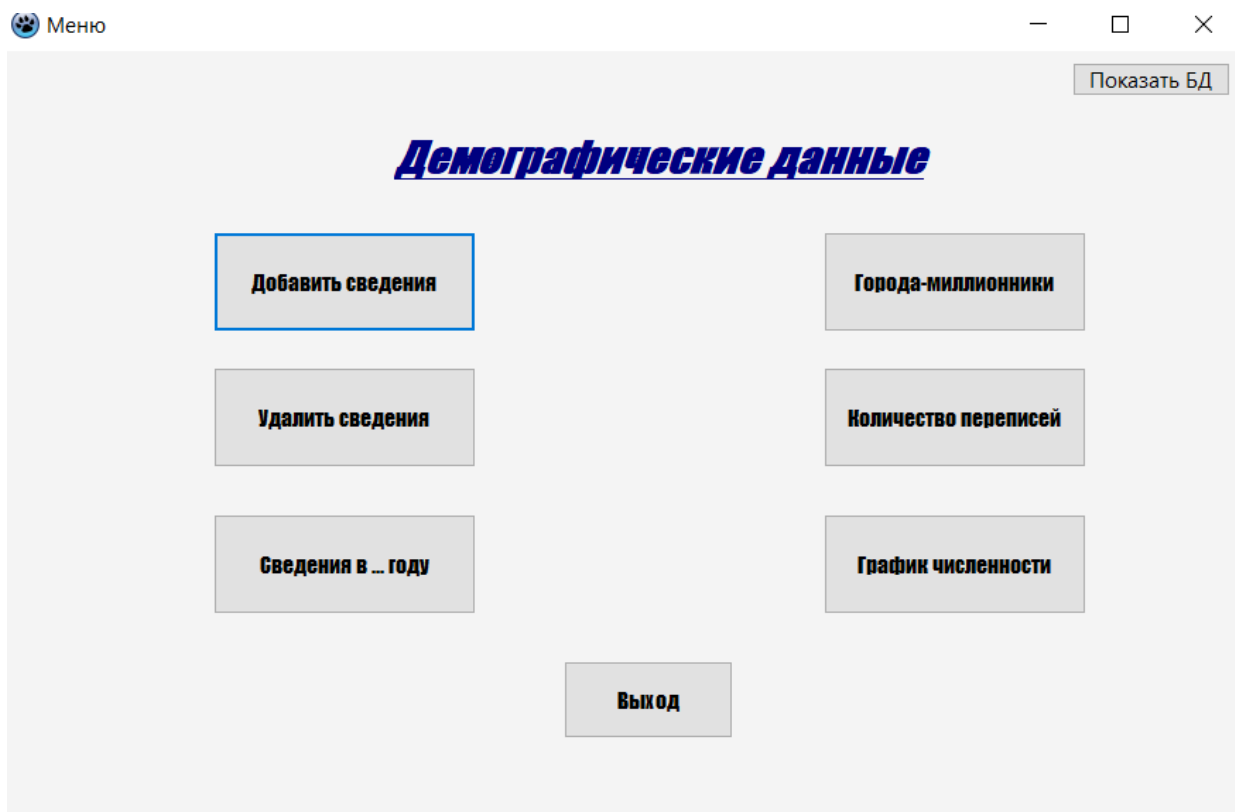


Рис. 1 – Форма главного окна

Добавление

Добавить сведения

Город

Численность

Год

Записать

Конец

Рис. 2 – Форма добавления сведений

Удаление сведений

Город

Численность

Год

Записать**Конец**

Рис. 3 – Форма удаления сведений

Демографические сведения, полученные в указанном году

Введите год

Краснодар численность 8428482 год переписи 2015

Искать**Конец**

Рис. 4 – Форма получения сведений

Города-миллионеры

Список городов-миллионеров (по данным последней переписи)

Краснодар численность 123456789
Симферополь численность 34853893
Москва численность 1383965137
Челябинск численность 1200580

Конец

Рис. 5 – Форма получения списка городов-миллионеров

Количество переписей

Количество переписей в указанном городе

Введите название города

численность 24802840 год переписи 2005
численность 123456789 год переписи 2022
численность 8428482 год переписи 2015
численность 492030 год переписи 2008
Количество переписей 4

Искать

Конец

Рис. 6 – Форма количества переписей



График изменения численности населения заданного города по годам

Построить

Конец

Рис. 7 – Форма рисования графика

База данных

Москва численность 47284944 год переписи 2018
Москва численность 120000 год переписи 2009
Краснодар численность 24802840 год переписи 2005
Москвариум численность 848338 год переписи 2005
Edit1 численность 1 год переписи 2
Краснодар численность 123456789 год переписи 2022
Краснодар численность 8428482 год переписи 2015
Якутск численность 28482942 год переписи 2016
Симферополь численность 34853893 год переписи 2022
Краснодар численность 492030 год переписи 2008
Симферополь численность -1813958877 год переписи 2019
Москва численность 1383965137 год переписи 2022
Челябинск численность 1200580 год переписи 2022
Москва численность 120000 год переписи 2009
Москва численность 47284944 год переписи 2018

Конец

Рис. 8 – База данных

Были разработаны диаграммы состояний интерфейса

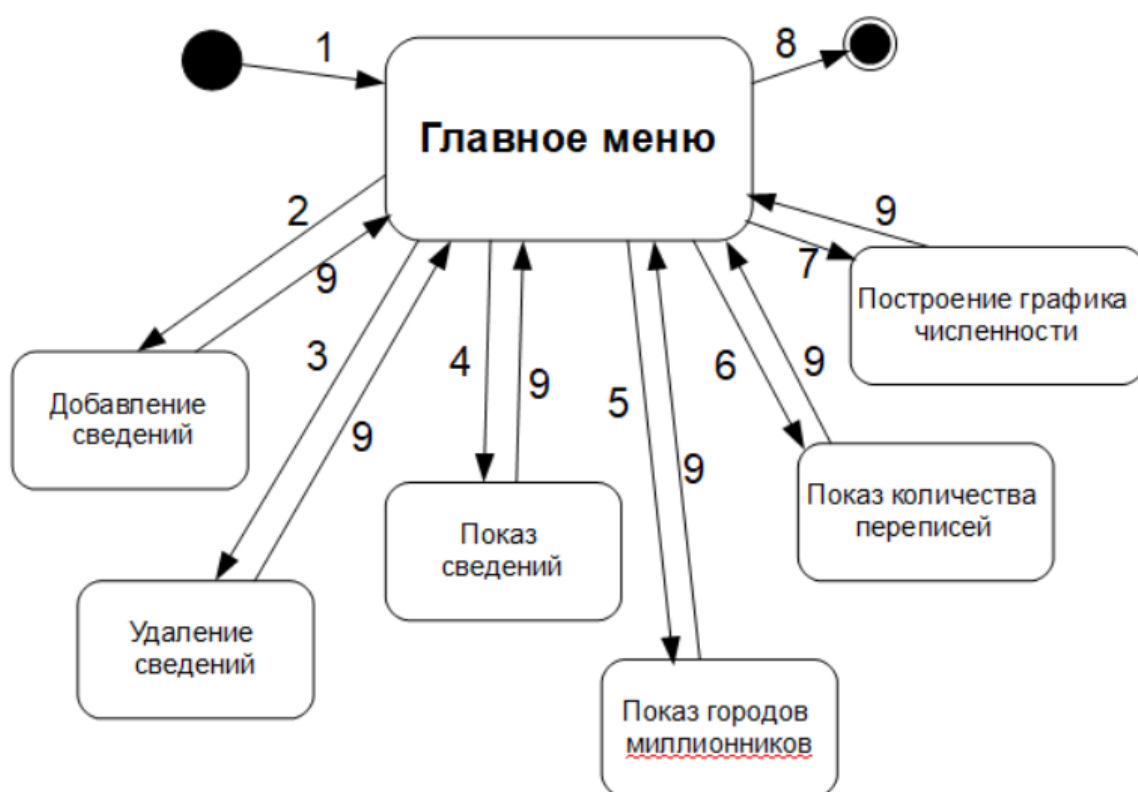


Рис. 9 - Диаграмма состояний интерфейса главной формы



Рис. 10 - Диаграмма состояний интерфейса формы «Количество переписей»

Была разработана диаграмма объектов

Была разработана диаграмма последовательностей для операции добавления записей



Рис. 13 - Диаграмма последовательностей для операции добавления записей

Текст программы

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
  Unit2, Unit3, Unit4, Unit5, Unit6, Unit7;
type
  { TForm1 }
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.lfm}
{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2.Show;
  Form2.edit1.setfocus;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Form3.Show;
  Form3.edit1.setfocus;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Form4.Show;
  Form4.edit1.setfocus;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Form5.Show;
  Form5.Button1.setfocus;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  Form6.Show;
  Form6.edit1.setfocus;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
  Form7.Show;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  close;
end;

end.
```

Рис. 14 - Текст формы главного окна

```

unit Unit2; // добавить
($mode ObjFPC){$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
  { TForm2 }
  TForm2 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;

var
  Form2: TForm2;

implementation
  {$R *.lfm}
  { TForm2 }
  procedure TForm2.Button1Click(Sender: TObject);
  begin
    z.name:=edit1.text;
    z.popul:=strtoint(edit2.text);
    z.year:=strtoint(edit3.text);
    edit1.clear;
    edit2.clear;
    edit3.clear;
    write(f,z);
    edit1.setfocus;
  end;

  procedure TForm2.Button2Click(Sender: TObject);
  begin
    closefile(f);
    self.hide;
  end;

  procedure TForm2.FormActivate(Sender: TObject);
  var size: integer;
  begin
    AssignFile(f, 'data.dat');
    {$I-} Reset(F); {$I+}
    if ioresult=0 then
    begin
      size := FileSize(f);
      seek(f, size);
    end
    else
      rewrite(f);
  end;

end.

```

Рис. 15 - Текст формы добавления

```

unit Unit3; // удалить
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
  { TForm3 }
  TForm3 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;

var
  Form3: TForm3;

implementation
  {$R *.lfm}
  { TForm3 }

  procedure TForm3.FormActivate(Sender: TObject);
  var size: integer;
  begin
    AssignFile(f, 'data.dat');
    {$I-} Reset(f); {$I+}
  end;

  procedure TForm3.Button2Click(Sender: TObject);
  begin
    closefile(f);
    self.hide;
  end;

  procedure TForm3.Button1Click(Sender: TObject);
  var k, i, counter: integer; vs: zap;
  begin
    z.name:=edit1.text;
    z.popul:=strtoint(edit2.text);
    z.year:=strtoint(edit3.text);
    edit1.clear;
    edit2.clear;
    edit3.clear;
    //удаление
    i := 0; counter := 0;
    while not EOF(f) do
    begin
      i := i+1;
      read (f, vs);
      if (vs.name = z.name) and (vs.popul = z.popul) and
        (vs.year = z.year) then
        begin
          k := FileSize(f) - counter;
          seek(f, k-1);
          read(f, vs);
          seek(f, i-1);
          write(f, vs);
          counter := counter + 1;
        end;
      end;
      edit1.setfocus;
    end;
  end.

```

Рис. 16 - Текст формы удаления

```

unit Unit4; // сведения в .. году
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
  { TForm4 }
  TForm4 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    ListBox1: TListBox;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;
var
  Form4: TForm4;

implementation

{$R *.lfm}

{ TForm4 }

procedure TForm4.Button2Click(Sender: TObject);
begin
  closefile(f);
  self.hide;
end;

procedure TForm4.FormActivate(Sender: TObject);
begin
  AssignFile(f, 'data.dat');
  {$I-} Reset(f); {$I+}
end;

procedure TForm4.Button1Click(Sender: TObject);
var vs: zap; str: string;
begin
  z.year:=strtoint(edit1.text);
  edit1.clear;
  while not EOF(f) do
    begin
      read (f, vs);
      if (vs.year = z.year) then
        begin
          str := vs.name + ' численность ' +
            IntToStr(vs.popul) + ' год переписи' + IntToStr(vs.year);
          ListBox1.Items.Add(str);
        end;
      end;
    edit1.setfocus;
  end;
end.

```

Рис. 17 - Текст формы со сведениями в указанном году

```

unit Unit5; // города миллионеры
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
  { TForm5 }
  TForm5 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    ListBox1: TListBox;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;
var
  Form5: TForm5;

implementation
{$R *.lfm}
{ TForm5 }

procedure TForm5.Button1Click(Sender: TObject);
begin
  closefile(f);
  self.hide;
end;

procedure TForm5.FormActivate(Sender: TObject);
var str: string;
const god = 2022;
begin
  AssignFile(f, 'data.dat');
  {$I-} Reset(f); {$I+}
  while not EOF(f) do
  begin
    read (f, z);
    if (z.year = god) and (z.popul >= 1000000) then
    begin
      str := z.name + ' численность ' + IntToStr(z.popul);
      ListBox1.Items.Add(str);
    end;
  end;
end;

end.

```

Рис. 18 - Текст формы с поиском городов-миллионеров


```

unit Unit6; // кол-во переписей
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
] { TForm6 }
]
  TForm6 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    ListBox1: TListBox;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;

var
  Form6: TForm6;

implementation
{$R *.lfm}
] { TForm6 }

] procedure TForm6.FormActivate(Sender: TObject);
] begin
    AssignFile(f, 'data.dat');
    {$I-} Reset(f); {$I+}
  end;

] procedure TForm6.Button1Click(Sender: TObject);
var vs: zap; str: string; counter: integer;
] begin
  z.name:=edit1.text;
  edit1.clear;
  counter := 0;
  while not EOF(f) do
  ]   begin
        read (f, vs);
        if (vs.name = z.name) then
          begin
            counter := counter + 1;
            str := ' численность ' + IntToStr(vs.popul) +
              ' год переписи ' + IntToStr(vs.year);
            ListBox1.Items.Add(str);
          end;
        end;
      str := 'Количество переписей ' + IntToStr(counter);
      ListBox1.Items.Add(str);
      edit1.setfocus;
    end;

] procedure TForm6.Button2Click(Sender: TObject);
] begin
  closefile(f);
  self.hide;
end;

end.

```

Рис. 19 - Текст формы нахождения количества переписей в указанном году

```

unit Unit7;    //график
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,
  Buttons, Grids, DBGrids, TAGraph, TASources, TAIIntervalSources, TASTyles,
  TASeries, TARadialSeries, HelpUnit;
type
  { TForm7 }
  TForm7 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Chart1: TChart;
    Chart1LineSeries1: TLineSeries;
    Edit1: TEdit;
    Image1: TImage;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;
var
  Form7: TForm7;

implementation
  {$R *.lfm}
  { TForm7 }

procedure TForm7.Button2Click(Sender: TObject);
begin
  closefile(f);
  self.hide;
end;

procedure TForm7.FormActivate(Sender: TObject);
begin
  AssignFile(f, 'data.dat');
  {$I-} Reset(f); {$I+}
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(Rect(0,0,Width,Height));
  Image1.Canvas.Brush.Color:=clBlack;
end;

procedure TForm7.Button1Click(Sender: TObject);
var counter, i, j: integer; vs: zap;
    a, b: array [1..22,1..2] of integer;    //[year, population]
begin
  Chart1LineSeries1 := TLineSeries.Create(Chart1);
  Chart1LineSeries1.SeriesColor:= clRed;
  Chart1LineSeries1.ShowPoints:= true;
  Chart1.AddSeries(Chart1LineSeries1);
  Chart1.Title.Visible:= true;
  Chart1.Title.Text.Text:= 'График изменения численности населения в г.' + edit1.text;
  z.name:=edit1.text;
  edit1.clear;
  counter := 0;
  for i:=1 to 22 do begin
    for j:=1 to 2 do
      a[i,j]:=0;
    end;
  end;

  while not EOF(f) do
  begin
    read (f, vs);
    if (vs.name = z.name) then
    begin
      counter := counter + 1;
      for i:=1 to 22 do
        if (vs.year mod 100 = i) then
        begin
          a[i,1]:=vs.year;
          a[i,2]:=vs.popul;
        end;
      end;
    end;
  end;
  for i:=1 to 22 do begin
    if a[i, 1] > 2000 then
      Chart1LineSeries1.AddXY(a[i, 1],a[i, 2]);
    end;
  end;
end;

```

Рис. 20 - Текст формы рисования графика изменения численности

```

unit Unit8;
{$mode ObjFPC}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, HelpUnit;
type
  { TForm8 }
  TForm8 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    ListBox1: TListBox;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;

var
  Form8: TForm8;

implementation
{$R *.lfm}
{ TForm8 }

procedure TForm8.FormActivate(Sender: TObject);
var str: string;
begin
  ListBox1.Clear;
  AssignFile(f, 'data.dat');
  {$I-} Reset(f); {$I+}
  while not EOF(f) do
  begin
    read (f, z);
    str := z.name + ' численность ' + IntToStr(z.popul) + ' год переписи ' + IntToStr(z.year);
    ListBox1.Items.Add(str);
  end;
end;

procedure TForm8.Button1Click(Sender: TObject);
begin
  closefile(f);
  self.hide;
end;

end.

```

Рис. 21 – Текст формы показа базы данных

```

unit HelpUnit;
{$mode ObjFPC}{$H+}
interface
type zap = record
  name: string[22];
  popul: integer;
  year: word;
end;

var f:file of zap;
z:zap;

implementation

end.

```

Рис. 22 - Текст вспомогательного модуля

```

program project1;|
{$mode objfpc}{$H+}
uses
  {$IFDEF UNIX}
  cthreads,
  {$ENDIF}
  {$IFDEF HASAMIGA}
  athreads,
  {$ENDIF}
  Interfaces,
  Forms, tachartlazaruspkg, HelpUnit, Unit1,
  Unit2, Unit3, Unit4, Unit5, Unit6,Unit7;
{$R *.res}
begin
  RequireDerivedFormResource:=True;
  Application.Scaled:=True;
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.CreateForm(TForm5, Form5);
  Application.CreateForm(TForm6, Form6);
  Application.CreateForm(TForm7, Form7);
  Application.Run;
end.

```

Рис. 23 - Текст основной программы

Вывод: было разработано, протестировано и отлажено приложение, работающее с базой данных (файлом) внешнеэкономического ведомства. Программа в интерактивном режиме формирует файл, добавляет и удаляет данные, а также воспринимает каждый из перечисленных запросов и дает на него ответ.

Задание 2. Создание программной системы с элементарным интерфейсом консольного режима на C++

Выполнить структурную декомпозицию, разработать структурную схему, содержащую не менее 3 подпрограмм, и алгоритмы этих подпрограмм. Реализовать на C++ в консольном режиме. Предусмотреть примитивный интерфейс типа меню, позволяющий выбирать нужную подпрограмму.

Написать программу нахождения корней функции $y=F(x)$ на заданном отрезке $[a,b]$ методом хорд и методом половинного деления отрезка с заданной точностью ξ . Интервал пользователь должен иметь возможность задавать по

запросу, функцию для расчета выбирать из списка функций, а метод поиска корня – из списка методов. Для реализации возможности выбора функции использовать указатель на функцию. В список функций включить не менее 5 функций по своему выбору.

Цель работы: создание программной системы в консольном режиме на языке C++.

Для выполнения работы были разработаны структурные схемы:

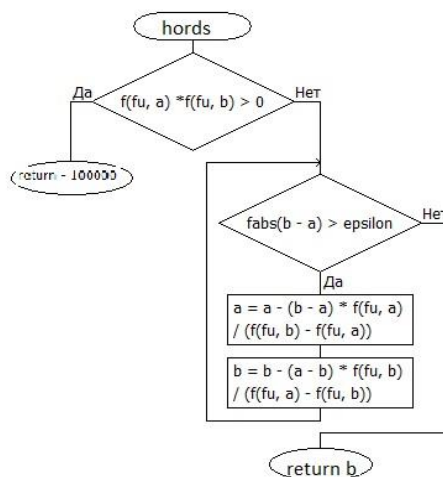


Рис. 24 – Структурная схема функции hords

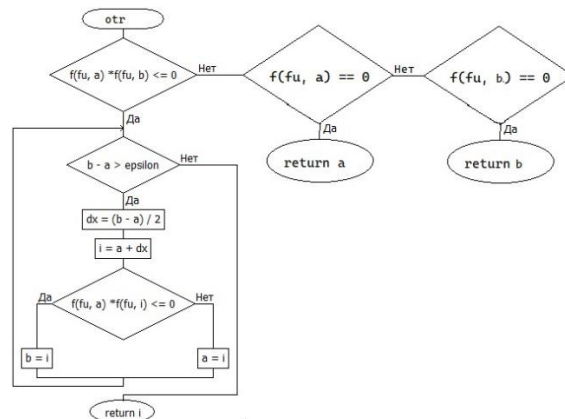


Рис. 25 - Структурная схема функции otr

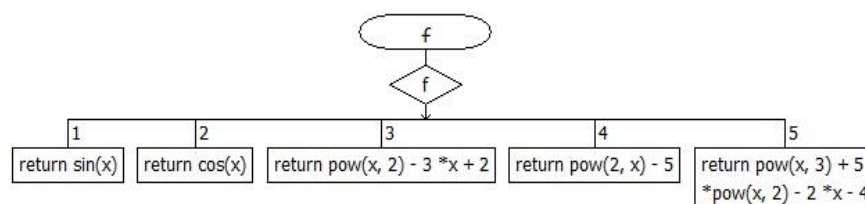


Рис. 26 - Структурная схема функции f

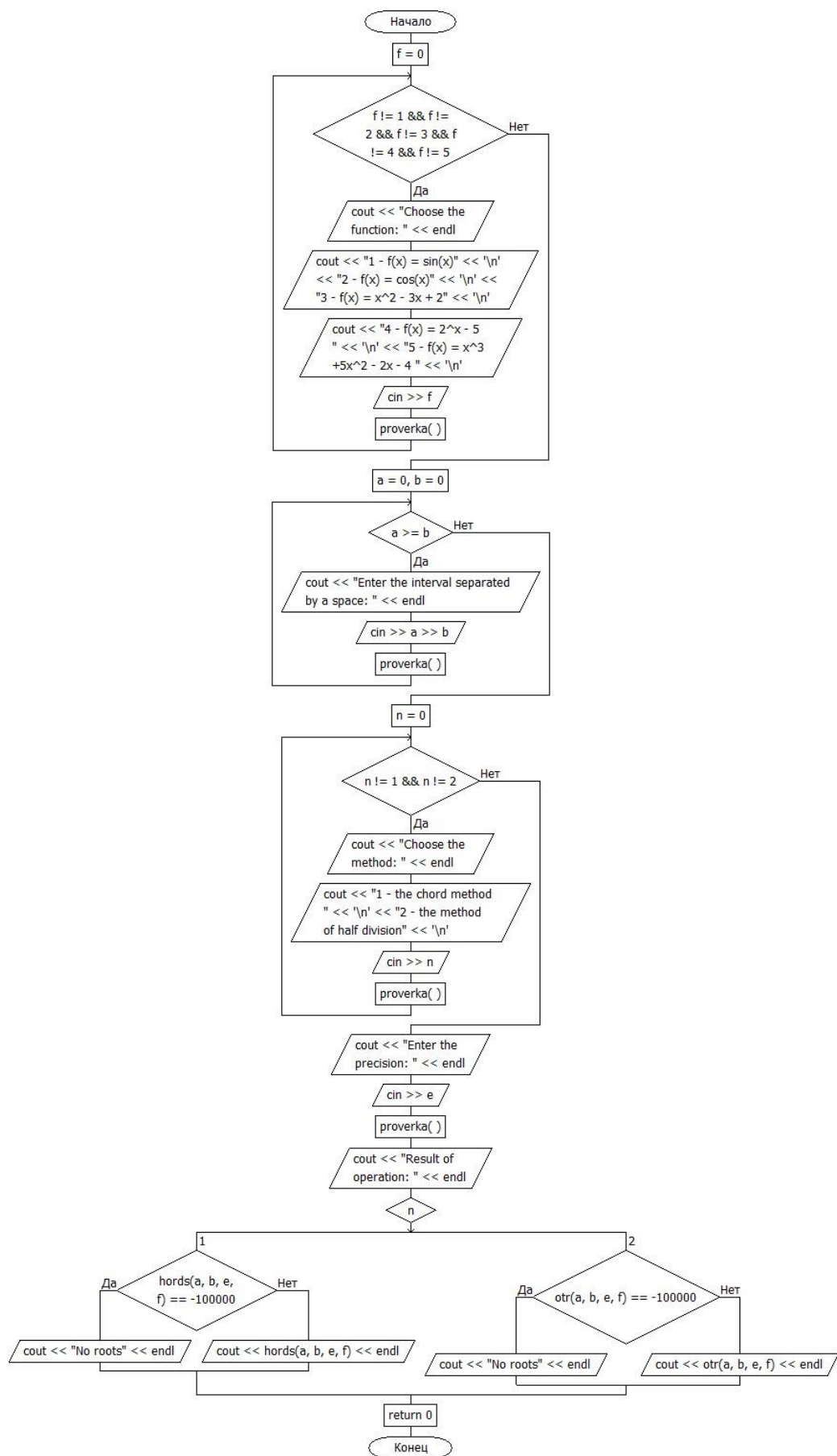


Рис. 27 - Структурная схема основной программы

Текст программы:

```
#include <iostream>
#include <cmath>
using namespace std;

double f(int f, double x) {
    switch (f)
    {
        case 1: return sin(x);
        case 2: return cos(x);
        case 3: return pow(x,2) - 3*x + 2;
        case 4: return pow(2, x) - 5;
        case 5: return pow(x,3) + 5* pow(x,2) - 2*x - 4;
    }
}

double otr(double a, double b, double epsilon, int fu){
    double i, dx;
    if (f(fu, a) == 0) return a;
    if (f(fu, b) == 0) return b;
    if (f(fu, a) * f(fu, b) <= 0) {
        while (b - a > epsilon) {
            dx = (b - a) / 2;
            i = a + dx;
            if (f(fu, a) * f(fu, i) <= 0) b = i;
            else a = i;
        }
        return i;
    }
    else return -100000;
}

double hords(double a, double b, double epsilon, int fu) {
    if (f(fu, a) * f(fu, b) > 0) return -100000;
    while (fabs(b - a) > epsilon) {
        a = a - (b - a) * f(fu, a) / (f(fu, b) - f(fu, a));
        b = b - (a - b) * f(fu, b) / (f(fu, a) - f(fu, b));
    }
    return b;
}

inline int proverka() {
    if (cin.fail()) {
        cout << "Invalid input! Please enter a number." << endl;
        cin.clear();
        cin.ignore(1000, '\n');
        return 1;
    }
}

int main()
{
    int f = 0;
    while (f != 1 && f != 2 && f != 3 && f != 4 && f != 5) {
        cout << "Choose the function: " << endl;
        cout << "1 - f(x) = sin(x)" << '\n' << "2 - f(x) = cos(x)" << '\n' << "3 - f(x) = x^2 - 3x + 2" << '\n';
        cout << "4 - f(x) = 2^x - 5 " << '\n' << "5 - f(x) = x^3 +5x^2 - 2x - 4 " << '\n';
        cin >> f;
        proverka();
    }

    double a = 0, b = 0;
    while (a >= b) {
```

```

        cout << "Enter the interval separated by a space: " << endl;
        cin >> a >> b;
        proverka();
    }

    int n = 0;
    while (n != 1 && n != 2) {
        cout << "Choose the method: " << endl;
        cout << "1 - the chord method " << '\n' << "2 - the method of half division"
<< '\n';
        cin >> n;
        proverka();
    }

    cout << "Enter the precision: " << endl;
    double e;
    cin >> e;
    proverka();

    cout << "Result of operation: " << endl;
    switch (n)
    {
    case 1:
        if (hords(a, b, e, f) == -100000) cout << "No roots" << endl;
        else cout << hords(a, b, e, f) << endl;
        break;
    case 2:
        if (otr(a, b, e, f) == -100000) cout << "No roots" << endl;
        else cout << otr(a, b, e, f) << endl;
        break;
    }

    return 0;
}

```

```

Choose the function:
1 - f(x) = sin(x)
2 - f(x) = cos(x)
3 - f(x) = x^2 - 3x + 2
4 - f(x) = 2^x - 5
5 - f(x) = x^3 + 5x^2 - 2x - 4
5
Enter the interval separated by a space:
-6 -3
Choose the method:
1 - the chord method
2 - the method of half division
1
Enter the precision:
0.0001
Result of operation:
-5.23607

```

Рис. 28 - Демонстрация работы программы


```
Choose the function:
1 - f(x) = sin(x)
2 - f(x) = cos(x)
3 - f(x) = x^2 - 3x + 2
4 - f(x) = 2^x - 5
5 - f(x) = x^3 + 5x^2 - 2x - 4
1
Enter the interval separated by a space:
1 4
Choose the method:
1 - the chord method
2 - the method of half division
2
Enter the precision:
0.0001
Result of operation:
3.14151
```

Рис. 29 – Демонстрация работы программы

Вывод: была создана программная система в консольном режиме на языке C++ с примитивным интерфейсом типа меню, позволяющим выбрать нужную подпрограмму.

Задание 3. Создание программной системы с Qt интерфейсом на C++

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

База демографических данных (файл) содержит сведения о населении городов: название города, численность населения, год проведения переписи. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать все демографические сведения, полученные в указанном году.

2. Показать список городов-миллионеров (по данным последней переписи).

3. Определить, сколько раз проводилась перепись населения в указанном городе.

4. Построить график изменения численности населения заданного города по годам.

Цель работы: создание программной системы на C++ с использованием возможностей QtCreator.

Для выполнения работы были разработаны формы приложения:

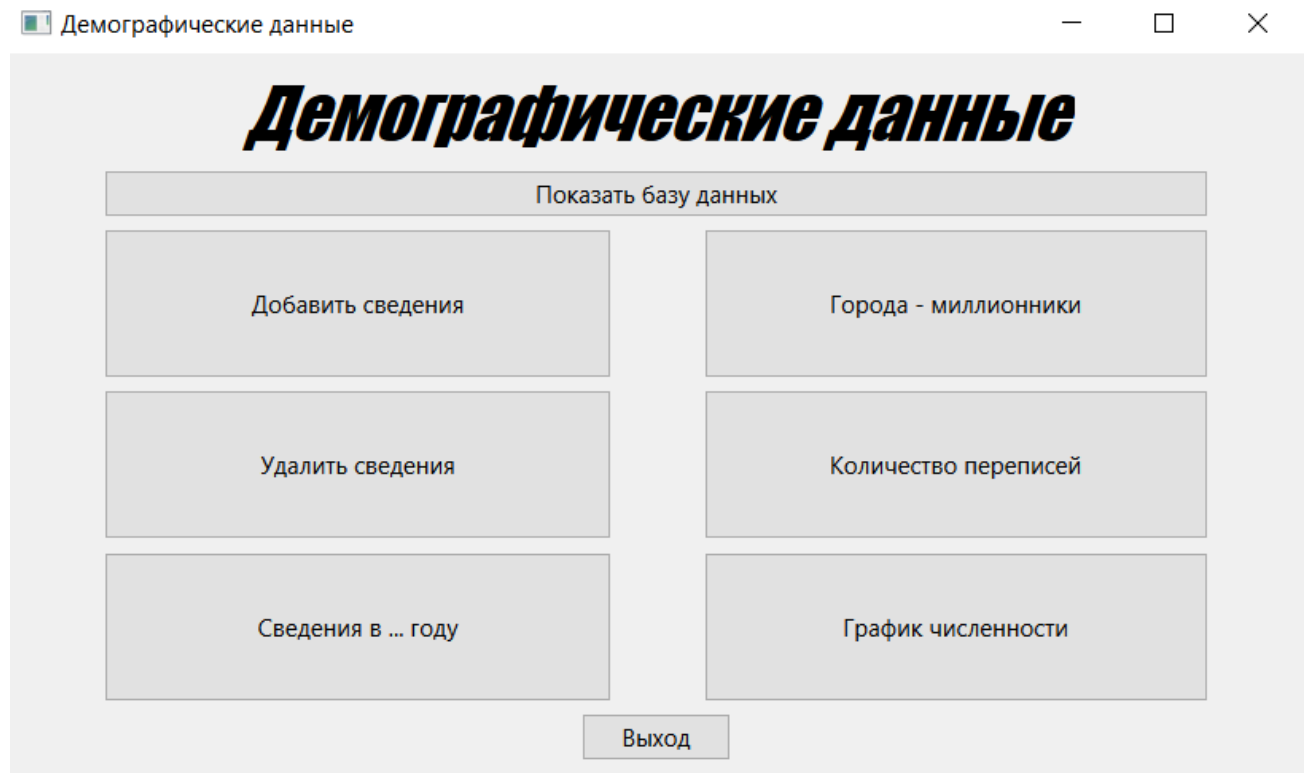


Рис. 30 – Форма главного окна

База данных

	Город	Численность	Год
1	Москва	13010112	2022
2	Москва	12678079	2020
3	Москва	12108257	2014
4	Москва	9783242	1999
5	Москва	9932932	2000
6	Краснодар	1121291	2022
7	Краснодар	744900	2011
8	Мурманск	267422	2022
9	Мурманск	311209	2009
10	Мурманск	307310	2011
11	Краснодар	709000	2007
12	Краснодар	899542	2018

Назад

Рис. 31 – Форма показа базы данных

Добавление данных

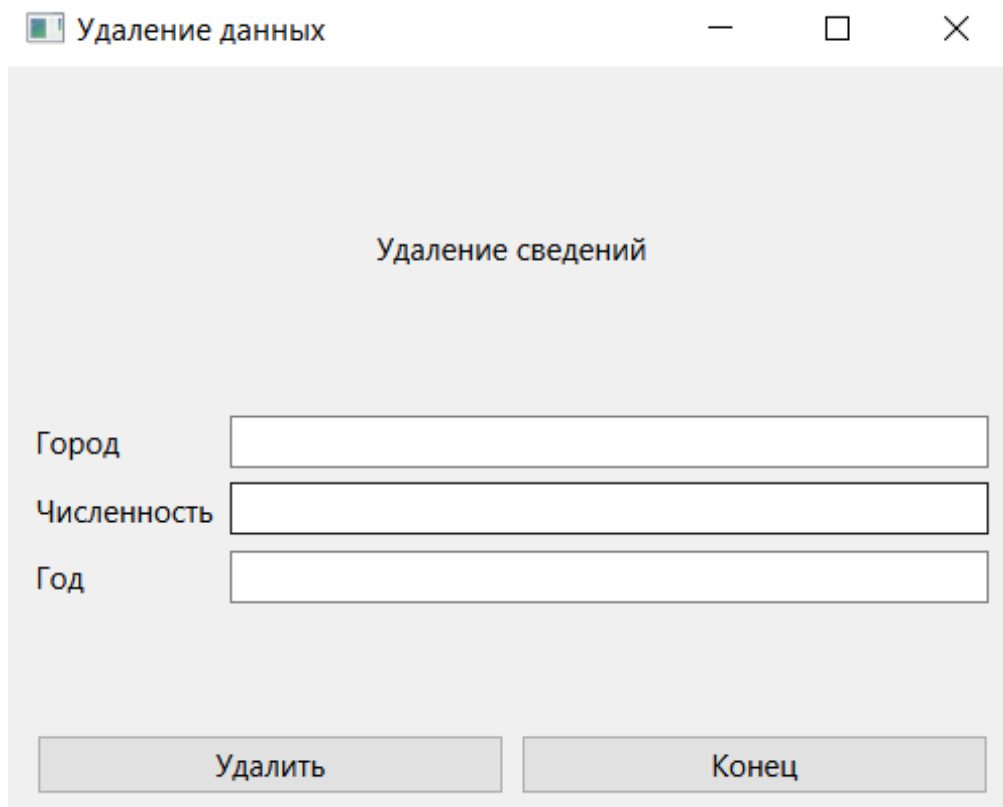
Добавление сведений

Город

Численность

Год

Рис. 32 – Форма добавления сведений



Удаление данных

Удаление сведений

Город

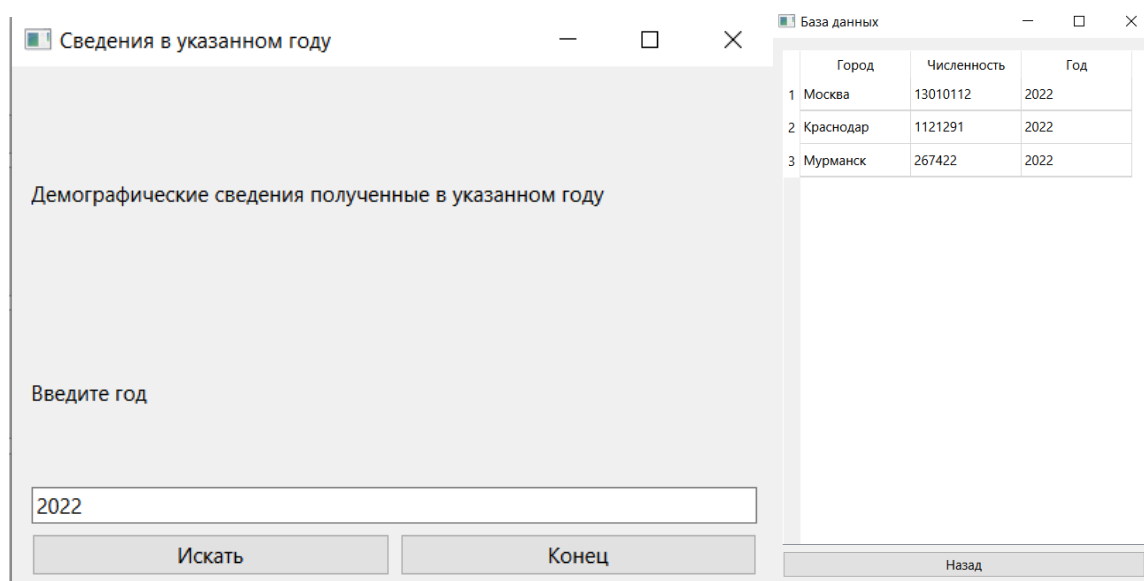
Численность

Год

Удалить

Конец

Рис. 33 – Форма удаления сведений



Сведения в указанном году

Демографические сведения полученные в указанном году

Введите год

2022

Искать

Конец

База данных

	Город	Численность	Год
1	Москва	13010112	2022
2	Краснодар	1121291	2022
3	Мурманск	267422	2022

Назад

Рис. 34 – Форма получения сведений

База данных

	Город	Численность	Год
1	Москва	13010112	2022
2	Краснодар	1121291	2022

Назад

Рис. 35 – Форма получения списка городов-миллионеров

База данных

	Город	Численность	Год
1	Москва	13010112	2022
2	Москва	12678079	2020
3	Москва	12108257	2014
4	Москва	9783242	1999
5	Москва	9932932	2000

Назад

Перепись

Количество переписей в указанном городе

Введите город

Москва

Искать Конец

Рис. 36 – Форма количества переписей

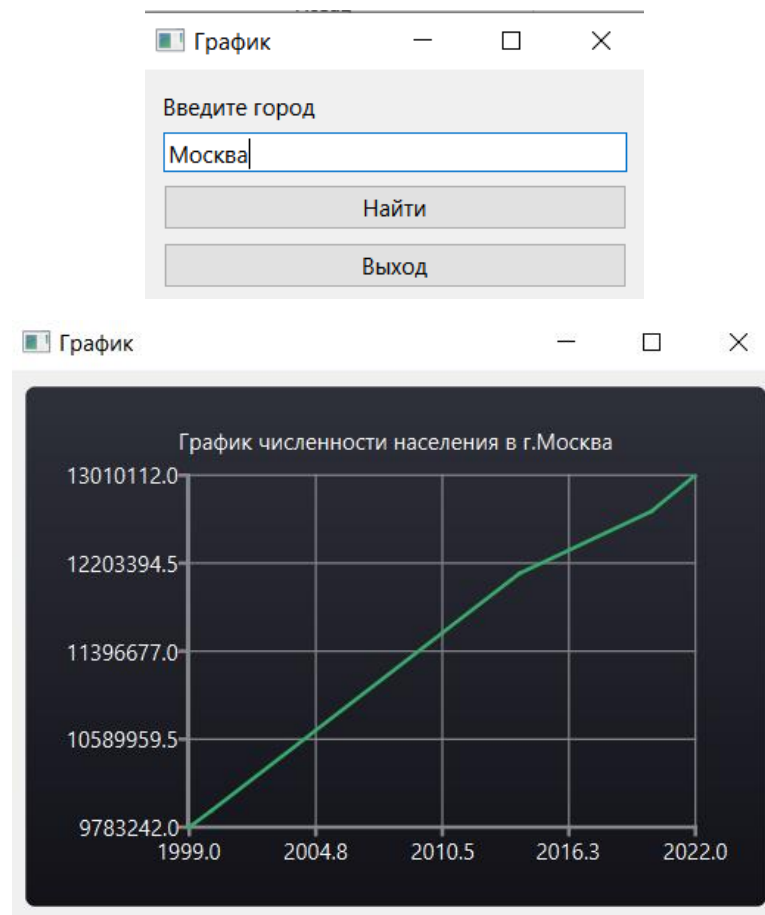


Рис. 37 – Форма рисования графика

Были разработаны диаграммы состояний интерфейса

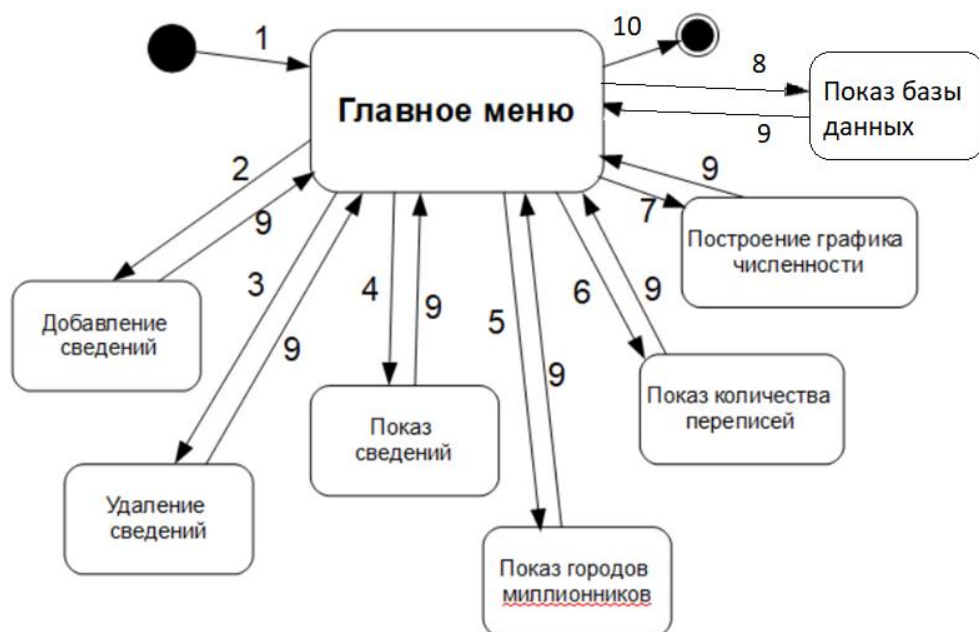


Рис. 38 - Диаграмма состояний интерфейса главной формы

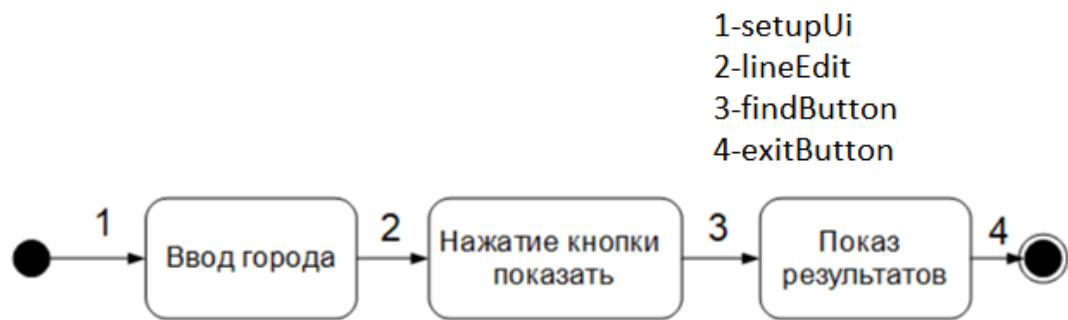


Рис. 39 - Диаграмма состояний интерфейса формы «Количество переписей»

Была разработана диаграмма объектов



Рис. 40 - Диаграмма объектов

Была разработана диаграмма классов

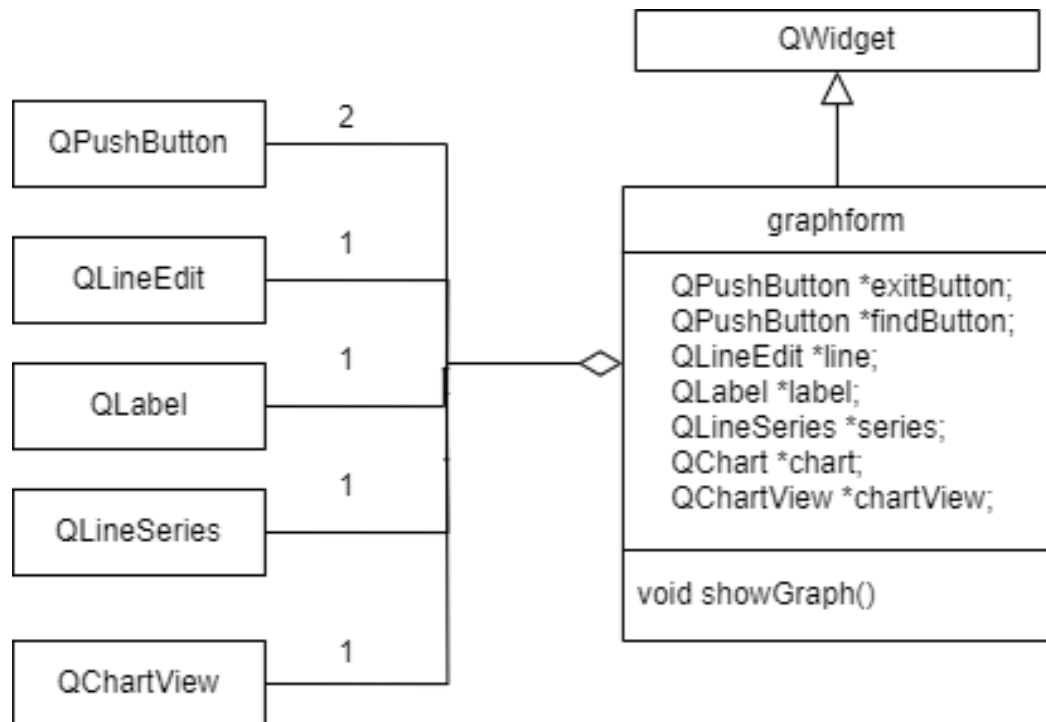


Рис.41 - Диаграмма классов формы рисования графика

Была разработана диаграмма последовательностей для операции добавления записей

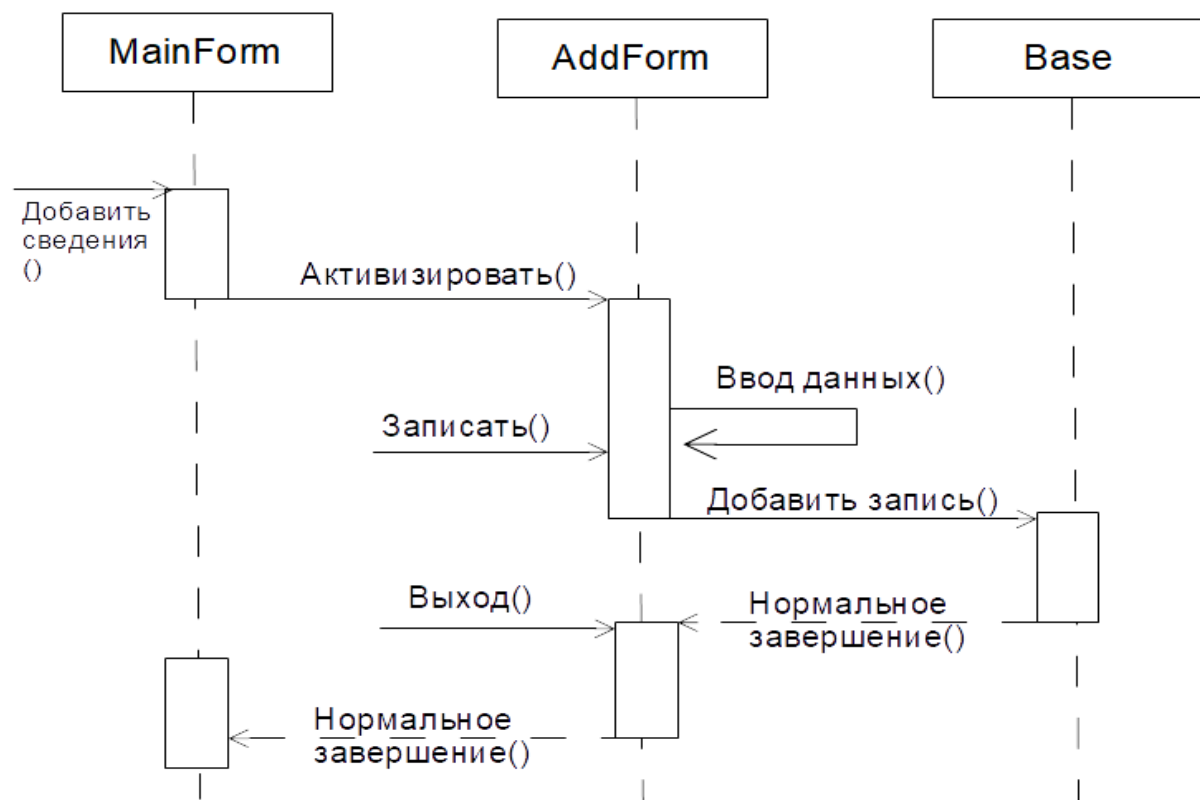


Рис. 42 - Диаграмма последовательностей для операции добавления записей

Текст программы:

Main.cpp:

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char* argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

#include "addform.h"
#include "delform.h"
#include "graphform.h"
#include "millionform.h"
#include "perepisform.h"
#include "showallform.h"
#include "yearform.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget* parent = nullptr);
    ~MainWindow();

private:
    Ui::MainWindow* ui;
    addForm winAdd;
    delForm winDel;
    showAllForm winShowAll;
    yearForm winYear;
    perepisForm winPerepis;
    graphform winGraph;

public slots:
    void showAdd();
    void showPrint(); // показать форму Отображения всех
    void showDel();
    void showYear();
    void showMillion();
    void showPerepis();
    void showGraph();
}
```

```
};
```

```
#endif // MAINWINDOW_H
```

MainWindow.cpp:

```
#include "mainwindow.h"  
#include "ui_mainwindow.h"
```

```
MainWindow::MainWindow(QWidget* parent) : QMainWindow(parent), ui(new  
Ui::MainWindow)  
{  
    ui->setupUi(this);  
    this->setWindowTitle("Демографические данные");  
  
    connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));  
    connect(ui->addButton, SIGNAL(clicked(bool)), this, SLOT(showAdd()));  
    connect(ui->delButton, SIGNAL(clicked(bool)), this, SLOT(showDel()));  
    connect(ui->showDataButton, SIGNAL(clicked(bool)), this, SLOT(showPrint())); //  
    connect(ui->yearButton, SIGNAL(clicked(bool)), this, SLOT(showYear()));  
    connect(ui->millionButton, SIGNAL(clicked(bool)), this, SLOT(showMillion()));  
    connect(ui->perepisButton, SIGNAL(clicked(bool)), this, SLOT(showPerepis()));  
    connect(ui->grButton, SIGNAL(clicked(bool)), this, SLOT(showGraph()));  
}  
  
MainWindow::~MainWindow()  
{  
    delete ui;  
}  
  
void MainWindow::showAdd() {  
    winAdd.show();  
}  
  
void MainWindow::showDel() {  
    winDel.show();  
}  
  
void MainWindow::showPrint() {  
    winShowAll.showAll();  
}  
  
void MainWindow::showYear() {  
    winYear.show();  
}  
  
void MainWindow::showMillion() {  
    winShowAll.showMil();  
}  
  
void MainWindow::showPerepis() {  
    winPerepis.show();  
}  
  
void MainWindow::showGraph() {  
    winGraph.show();  
}
```

data.h:

```
#ifndef DATA_H  
#define DATA_H
```

```

#include <QFile>
#include <QDataStream>

struct recType {
    QString city, population, year;
};

class TData
{
    bool k1, k2, k3;
public:
    QFile* f, * f1;
    recType r;
    TData();
    ~TData();
    bool addRec(recType r);
    bool readRec();
    bool delRec(const recType r1);
    int count();
};

#endif // DATA_H

```

data.cpp:

```

#include "data.h"
#include "mainwindow.h"
#include <QMessageBox>

TData::TData()
{
    f = new QFile("book.txt");
    if (!f->exists())
    {
        QMessageBox msg(QMessageBox::Critical, "Файл не найден", "Файл book.txt
создан", QMessageBox::Ok, 0);
        msg.exec();
    }
    f->open(QFile::ReadWrite);
}

TData::~TData()
{
    f->close();
    delete f;
}

bool TData::addRec(recType r)
{
    f->seek(f->size());
    QDataStream out(f);
    out << r.city << r.population << r.year;
    return true;
}

bool TData::readRec()
{
    QDataStream in(f);
    if (in.atEnd()) return false;
    else
    {
        in >> r.city >> r.population >> r.year;
        return true;
    }
}

```

```

}

bool TData::delRec(const recType r1)
{
    bool fff = readRec();

    f1 = new QFile("book2.txt");
    f1->open(QFile::ReadWrite);
    QDataStream out(f1);

    while (fff)
    {
        k1 = (r1.city == r.city);
        k2 = (r1.population == r.population);
        k3 = (r1.year == r.year);

        if (!(k1 && k2 && k3))
        {
            f1->seek(f1->size());
            out << r.city << r.population << r.year;
        }
        fff = readRec();
    }
    f->close();
    f1->close();
    remove("book.txt");
    f1->rename("book.txt");
    return fff;
}

int TData::count()
{
    int c = 0;

    TData d;
    d.f->reset();
    while (d.readRec()) c++;

    return c;
}

```

addForm.h:

```

#ifndef ADDFORM_H
#define ADDFORM_H

#include <QWidget>

namespace Ui {
    class addForm;
}

class addForm : public QWidget
{
    Q_OBJECT

public:
    explicit addForm(QWidget* parent = nullptr);
    ~addForm();

public slots:
    void addRecord();

private:

```

```

        Ui::addForm* ui;
};

#endif // ADDFORM_H

```

addForm.cpp:

```

#include "addForm.h"
#include "ui_addForm.h"
#include "data.h"
#include "mainwindow.h"

addForm::addForm(QWidget* parent) :
    QWidget(parent),
    ui(new Ui::addForm)
{
    ui->setupUi(this);
    this->setWindowTitle("Добавление данных");
    connect(ui->addButton, SIGNAL(clicked(bool)), this, SLOT(addRecord()));
    connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
}

void addForm::addRecord()
{
    TData d;
    recType r;
    r.city = ui->line1->text();
    r.population = ui->line2->text();
    r.year = ui->line3->text();
    ui->line1->clear();
    ui->line2->clear();
    ui->line3->clear();
    d.addRec(r);
}

addForm::~addForm()
{
    delete ui;
}

```

delForm.h:

```

#ifndef DELFORM_H
#define DELFORM_H

#include "data.h"
#include <QWidget>

namespace Ui {
    class delForm;
}

class delForm : public QWidget
{
    Q_OBJECT

public:
    explicit delForm(QWidget* parent = nullptr);
    ~delForm();
public slots:
    void deleteRec();

private:
    Ui::delForm* ui;
}

```

```
};

#endif // DELFORM_H
```

delForm.cpp:

```
#include "delform.h"
#include "ui_delform.h"

delForm::delForm(QWidget* parent) :
    QWidget(parent),
    ui(new Ui::delForm)
{
    ui->setupUi(this);
    this->setWindowTitle("Удаление данных");

    connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
    connect(ui->delButton, SIGNAL(clicked(bool)), this, SLOT(deleteRec()));
}

void delForm::deleteRec()
{
    recType r;

    r.city = ui->city->text();
    r.population = ui->popul->text();
    r.year = ui->year->text();

    TData book;
    book.delRec(r);

    ui->city->clear();
    ui->popul->clear();
    ui->year->clear();
}

delForm::~delForm()
{
    delete ui;
}
```

graphForm.h:

```
#ifndef GRAPHFORM_H
#define GRAPHFORM_H

#include <QWidget>

#include <QtCharts/QChartView>
#include <QtCharts/QLineSeries>
#include <QtCharts/QValueAxis>
#include <QtWidgets/QApplication>
#include <QtWidgets/QMainWindow>

#include <QPushButton>
#include <QLabel>
#include <QLineEdit>

#include "data.h"

class graphform : public QWidget
{
```

```

    Q_OBJECT
    QPushButton* exitButton;
    QPushButton* findButton;
    QLineEdit* line;
    QLabel* label;
    QLineSeries* series;
    QChart* chart;
    QChartView* chartView;
public:
    explicit graphform(QWidget* parent = nullptr);

public slots:
    void showGraph();

};

#endif // GRAPHFORM_H

```

graphForm.cpp:

```

#include "graphform.h"
#include <QBoxLayout>
#include <QMessageBox>
#include <QtDataVisualization>

graphform::graphform(QWidget* parent) : QWidget{ parent }
{
    this->setWindowTitle("График");

    exitButton = new QPushButton("Выход", this);
    findButton = new QPushButton("Найти", this);

    line = new QLineEdit("", this);
    label = new QLabel("Введите город", this);

    QVBoxLayout* layout1 = new QVBoxLayout(this);
    layout1->addWidget(label);
    layout1->addWidget(line);
    layout1->addWidget(findButton);
    layout1->addWidget(exitButton);

    connect(exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
    connect(findButton, SIGNAL(clicked(bool)), this, SLOT(showGraph()));
}

void graphform::showGraph()
{
    chart = new QChart();

    chart->setTheme(QChart::ChartThemeDark);

    chartView = new QChartView(chart);

    QVector<QPointF> points;

    QString city = line->text();

    TData d;
    d.f->reset();
    if (d.readRec())
    {
        if (d.r.city == city) {
            QPointF point(d.r.year.toInt(), d.r.population.toInt());
            points.push_back(point);
        }
    }
}

```

```

    }
    while (d.readRec()) {
        if (d.r.city == city) {
            QPointF point(d.r.year.toInt(), d.r.population.toInt());
            points.push_back(point);
        }
    }
}
else
{
    QMessageBox msg(QMessageBox::Critical, "Нет данных", "Данные не найдены",
QMessageBox::Ok, 0);
    msg.exec();
}

std::sort(points.begin(), points.end(), [](const QPointF& p1, const QPointF& p2)
{return p1.x() < p2.x(); });
QLineSeries* series = new QLineSeries();
series->append(points);

chart->legend()->hide();
chart->addSeries(series);
chart->createDefaultAxes();
chart->setTitle("График численности населения в г." + city);

chartView->setRenderHint(QPainter::Antialiasing);
chartView->resize(420, 300);
chartView->setWindowTitle("График");
chartView->show();
}

```

perepisForm.h:

```

#ifndef PEREPISFORM_H
#define PEREPISFORM_H

#include <QWidget>
#include "showallform.h"

namespace Ui {
    class perepisForm;
}

class perepisForm : public QWidget
{
    Q_OBJECT

public:
    explicit perepisForm(QWidget* parent = nullptr);
    ~perepisForm();

public slots:
    void showData();

private:
    showAllForm winShowAll;
    Ui::perepisForm* ui;
};

#endif // PEREPISFORM_H

```


perepisForm.cpp:

```
#include "perepisform.h"
#include "ui_perepisform.h"

perepisForm::perepisForm(QWidget* parent) :
    QWidget(parent),
    ui(new Ui::perepisForm)
{
    ui->setupUi(this);
    this->setWindowTitle("Перепись");

    connect(ui->findButton, SIGNAL(clicked(bool)), this, SLOT(showData()));
    connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
}

void perepisForm::showData() {
    QString c = ui->lineEdit->text();
    winShowAll.showPer(c);
}

perepisForm::~perepisForm()
{
    delete ui;
}
```

showAllForm.h:

```
#ifndef SHOWALLFORM_H
#define SHOWALLFORM_H

#include <QWidget>
#include <QtGui>
#include <QTableWidget>
#include <QPushButton>
#include "data.h"

class showAllForm : public QWidget
{
    Q_OBJECT
    QTableWidget* table;
    QPushButton* btnExit;
    void showRow(int i, recType r);
public:
    showAllForm(QWidget* parent = nullptr);
    ~showAllForm() {}
    void showAll(); // показать все записи
    void showYear(int y);
    void showMil();
    void showPer(QString c);
};

#endif // SHOWALLFORM_H
```

showAllForm.cpp:

```
#include "showallform.h"
#include "mainwindow.h"
#include <QMessageBox>
#include <QTableWidget>
#include <QBoxLayout>
#include <iostream>
```

```

showAllForm::showAllForm(QWidget* parent) :QWidget(parent)
{
    this->setWindowTitle("База данных");

    QStringList strlist;
    strlist << "Город" << "Численность" << "Год";
    table = new QTableWidgetItem(1, 3, this);
    table->setHorizontalHeaderLabels(strlist);

    QHBoxLayout* layoutG2 = new QHBoxLayout();
    btnExit = new QPushButton("Назад", this);
    layoutG2->addWidget(btnExit);
    QVBoxLayout* layout = new QVBoxLayout(this);
    layout->addWidget(table);
    layout->addLayout(layoutG2);

    connect(btnExit, SIGNAL(clicked(bool)),
            this, SLOT(close()));
}

void showAllForm::showRow(int i, recType r)
{
    QTableWidgetItem* item;

    item = new QTableWidgetItem();
    item->setFlags(Qt::NoItemFlags);
    item->setText(r.city);
    table->setItem(i, 0, item);

    item = new QTableWidgetItem();
    item->setFlags(Qt::NoItemFlags);
    item->setText(r.population);
    table->setItem(i, 1, item);

    item = new QTableWidgetItem();
    item->setFlags(Qt::NoItemFlags);
    item->setText(r.year);
    table->setItem(i, 2, item);
}

void showAllForm::showAll()
{
    TData d;
    d.f->reset();
    if (!d.readRec())
    {
        QMessageBox msg(QMessageBox::Critical, "Нет данных", "База пуста",
        QMessageBox::Ok, 0);
        msg.exec();
    }
    else
    {
        showRow(0, d.r);
        int i = 0;
        table->setRowCount(d.count());

        while (d.readRec()) showRow(++i, d.r);

        resize(350, 500);
        show();
    }
}

void showAllForm::showMil()

```

```

{
    TData d;
    d.f->reset();
    if (!d.readRec())
    {
        QMessageBox msg(QMessageBox::Critical, "Нет данных", "Данные не найдены",
        QMessageBox::Ok, 0);
        msg.exec();
    }
    else
    {
        int i = -1;
        if (d.r.year.toInt() == 2022 && d.r.population.toInt() >= 1000000)
        showRow(++i, d.r);
        while (d.readRec())
        {
            if (d.r.year.toInt() == 2022 && d.r.population.toInt() >= 1000000)
            showRow(++i, d.r);
        }
        table->setRowCount(i + 1);
        resize(350, 500);
        show();
    }
}

void showAllForm::showPer(QString c) {
    TData d;
    d.f->reset();
    if (!d.readRec())
    {
        QMessageBox msg(QMessageBox::Critical, "Нет данных", "Данные не найдены",
        QMessageBox::Ok, 0);
        msg.exec();
    }
    else
    {
        int i = -1;
        if (d.r.city == c) showRow(++i, d.r);
        while (d.readRec()) {
            if (d.r.city == c) showRow(++i, d.r);
        }
        table->setRowCount(i + 1);
        resize(350, 500);
        show();
    }
}

void showAllForm::showYear(int y) {
    TData d;
    d.f->reset();
    if (!d.readRec())
    {
        QMessageBox msg(QMessageBox::Critical, "Нет данных", "Данные не найдены",
        QMessageBox::Ok, 0);
        msg.exec();
    }
    else
    {
        int i = -1;
        if (d.r.year.toInt() == y) showRow(++i, d.r);
        while (d.readRec()) {
            if (d.r.year.toInt() == y) showRow(++i, d.r);
        }
        table->setRowCount(i + 1);
    }
}

```

```

        resize(350, 500);
        show();
    }
}

```

yearForm.h:

```

#ifndef YEARFORM_H
#define YEARFORM_H

#include <QWidget>
#include "showallform.h"

namespace Ui {
    class yearForm;
}

class yearForm : public QWidget
{
    Q_OBJECT

public:
    explicit yearForm(QWidget* parent = nullptr);
    ~yearForm();

public slots:
    void showData();

private:
    showAllForm winShowAll;
    Ui::yearForm* ui;
};

#endif // YEARFORM_H

```

yearForm.cpp:

```

#include "yearform.h"
#include "ui_yearform.h"

yearForm::yearForm(QWidget* parent) :
    QWidget(parent),
    ui(new Ui::yearForm)
{
    ui->setupUi(this);
    this->setWindowTitle("Сведения в указанном году");

    connect(ui->findButton, SIGNAL(clicked(bool)), this, SLOT(showData()));
    connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
}

void yearForm::showData() {
    int y = ui->lineEdit->text().toInt();
    winShowAll.showYear(y);
}

yearForm::~yearForm()
{
    delete ui;
}

```

Вывод: было разработано, протестировано и отлажено приложение, работающее с базой данных (файлом) внешнеэкономического ведомства. Программа в интерактивном режиме формирует файл, добавляет и удаляет данные, а также воспринимает каждый из перечисленных запросов и дает на него ответ.