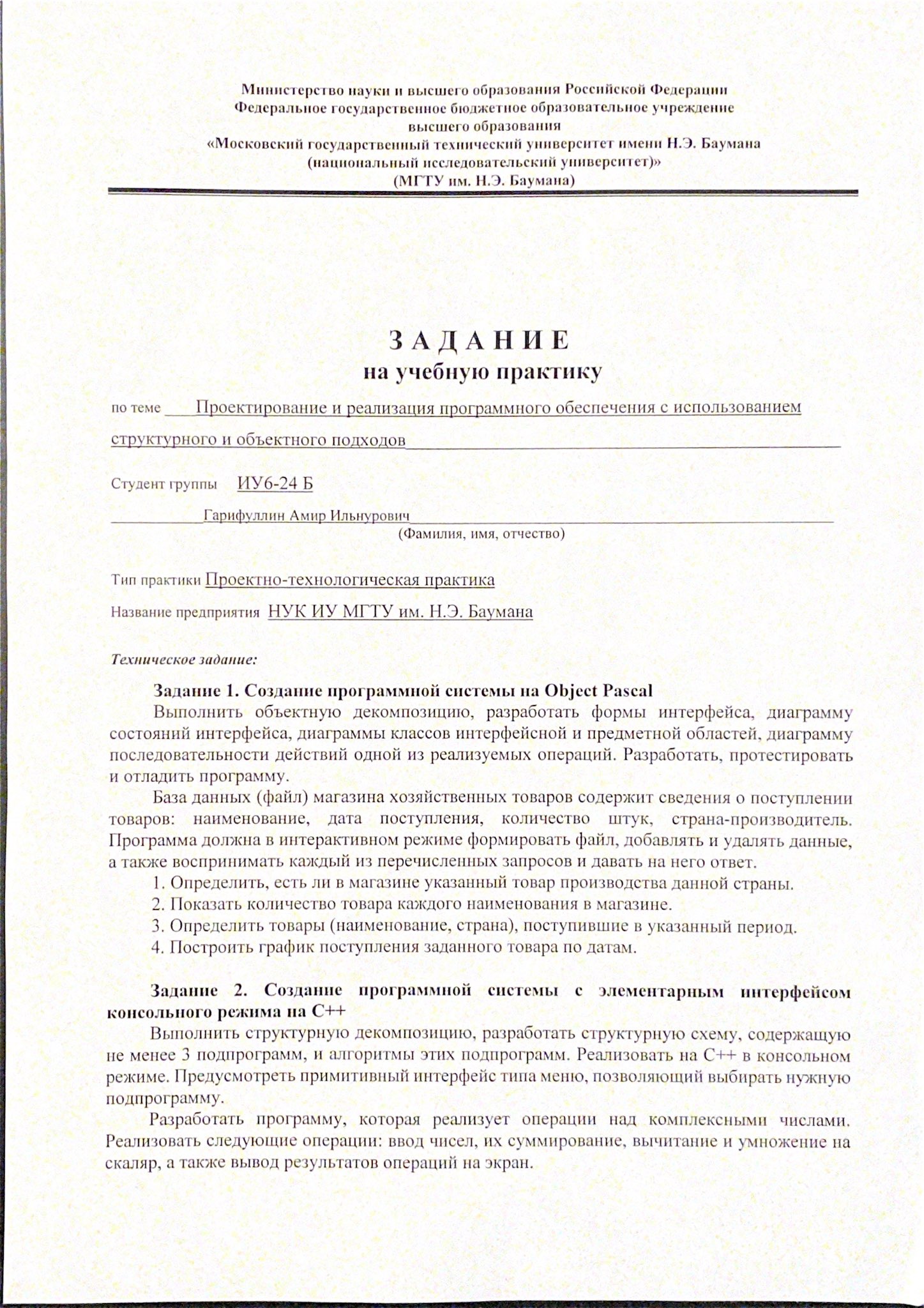
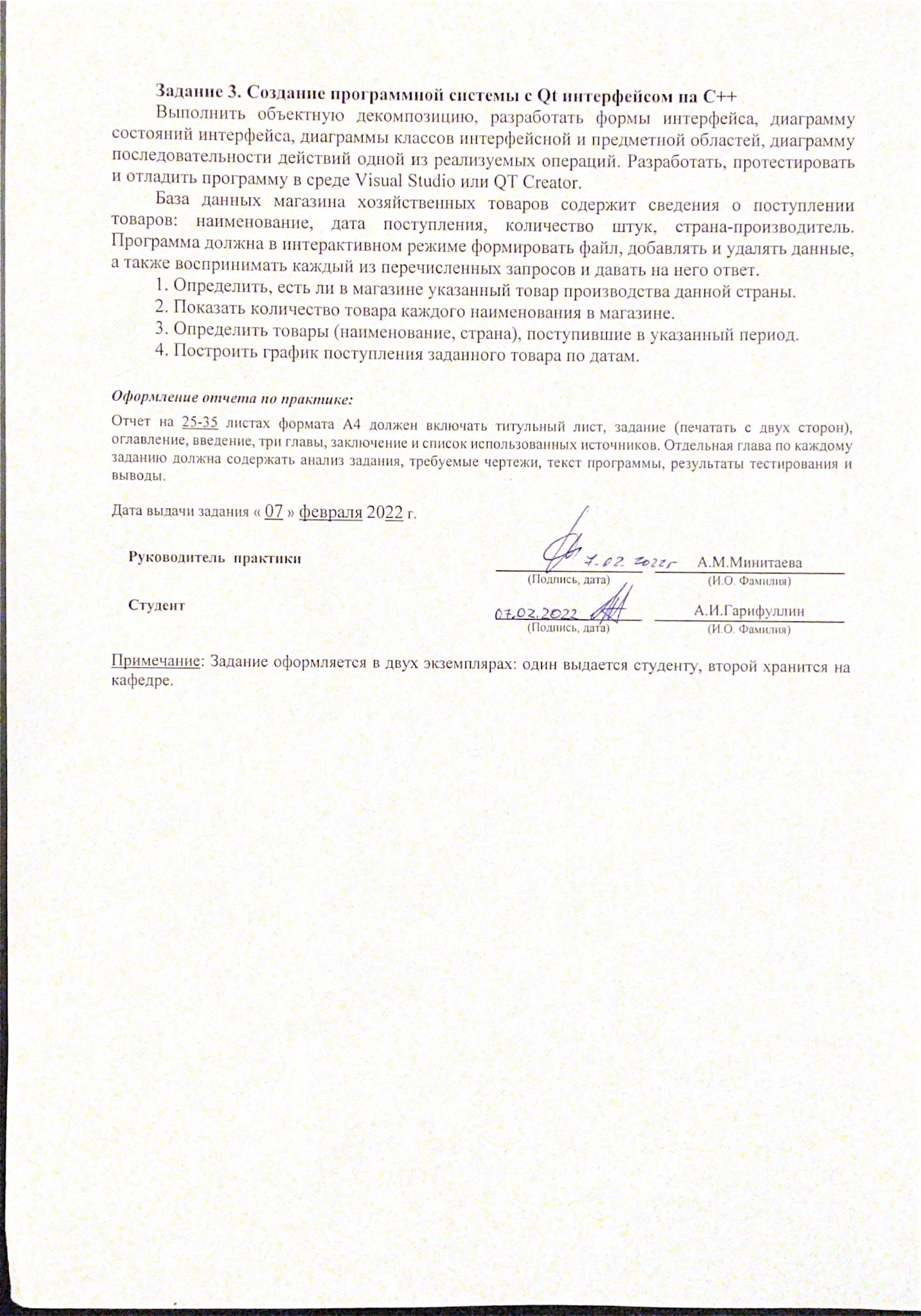


**

**

**Введение**

**Цели практики:**

− получение навыков создания небольших программных систем с оконными и консольными интерфейсами (проектирование, отладку и тестирование);

− создание качественных программных интерфейсов для взаимодействия с пользователем;

− расширение навыков разработки программ различного назначения.

**Задачи практики:**

− овладение методикой и получение практических навыков проектирования небольших программных систем при структурном и объектном подходах;

− воспитание внимания, аккуратности, систематичности, а также формирование интереса к изучаемой профессиональной деятельности.

Выполнение практикума должно способствовать формированию и развитию навыков и умений, обеспечивающих следующие компетенции:

− выделение объектов предметной области, обобщение их в классы, определение связей между классами;

− проектирование эргономичного обеспечения информационных систем;

− разработка и отладка компонентов аппаратно-программных комплексов с помощью современных автоматизированных средств проектирования;

− разработка проектной и эксплуатационной документации на программную и техническую продукцию;

− выполнение контроля разрабатываемых проектов и технической документации на соответствие стандартам и техническим требованиям;

− разработка интерфейсов «человек - ЭВМ».

# **Индивидуальное 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 данных (файл) магазина хозяйственных товаров содержит сведения о поступлении товаров: наименование, дата поступления, количество штук, страна-производитель. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Определить, есть ли в магазине указанный товар производства данной страны.

2. Показать количество товара каждого наименования в магазине.

3. Определить товары (наименование, страна), поступившие в указанный период.

4. Построить график поступления заданного товара по датам.

**Задача 1**

**Анализ задания**

База данных (файл) магазина хозяйственных товаров содержит сведения о поступлении товаров: наименование, дата поступления, количество штук, страна-производитель. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

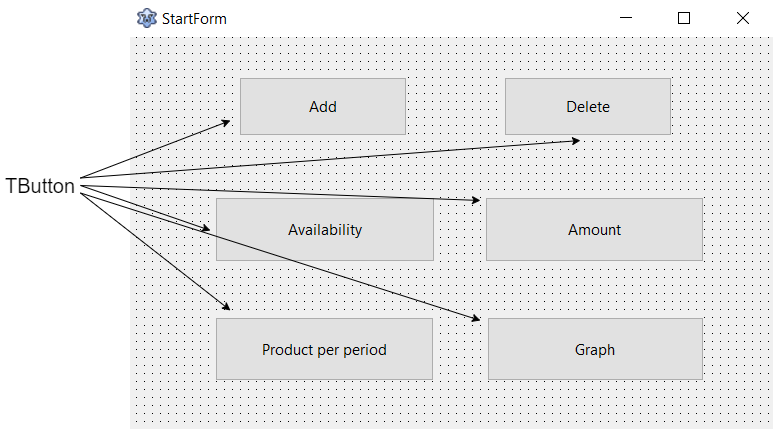
* Определить, есть ли в магазине указанный товар производства данной страны.
* Показать количество товара каждого наименования в магазине.
* Определить товары (наименование, страна), поступившие в указанный период.
* Построить график поступления заданного товара по датам.

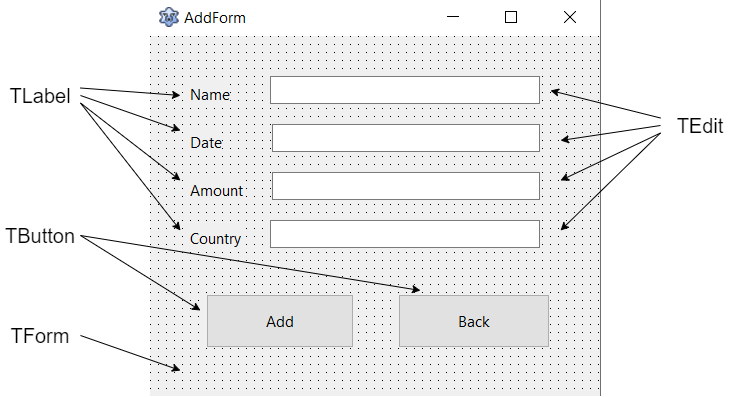
**Схемы**

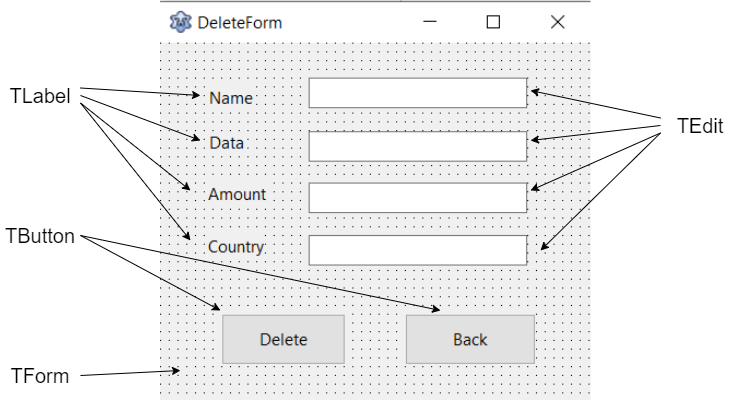
*Объектная декомпозиция:*

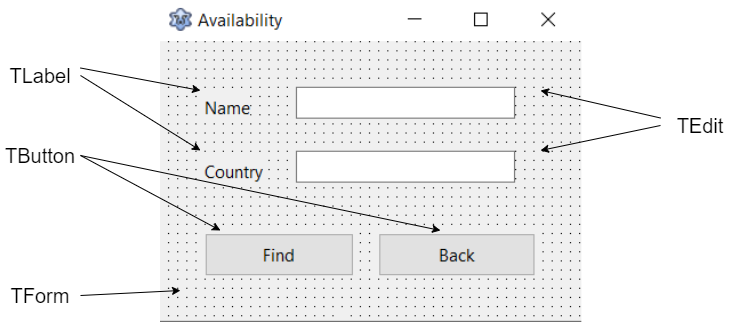


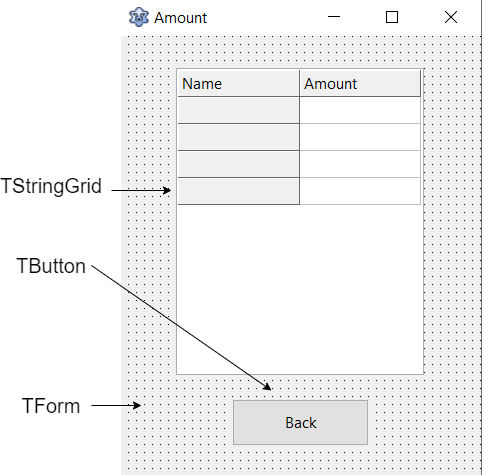
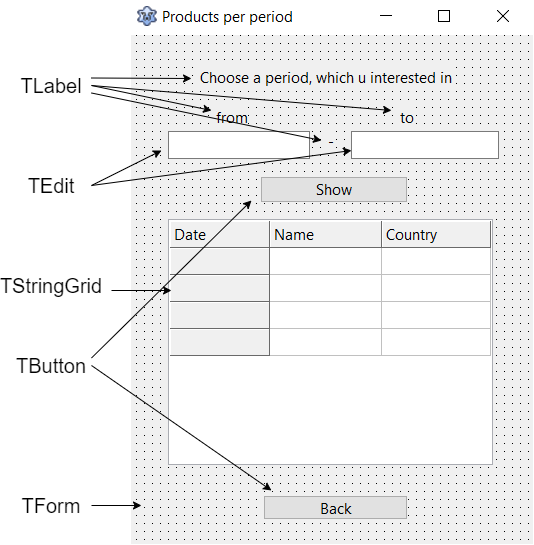
*Формы интерфейса:*

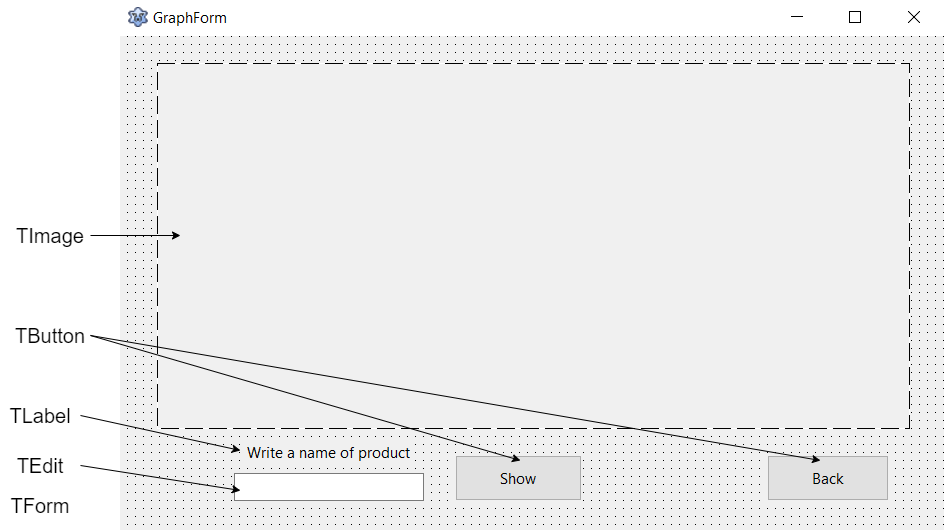
**



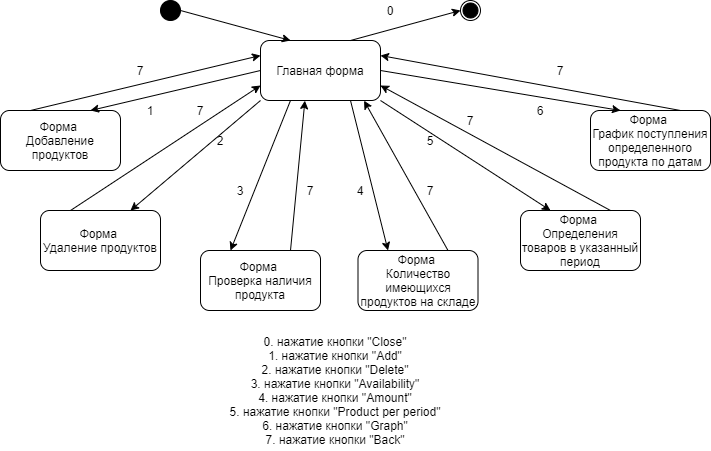




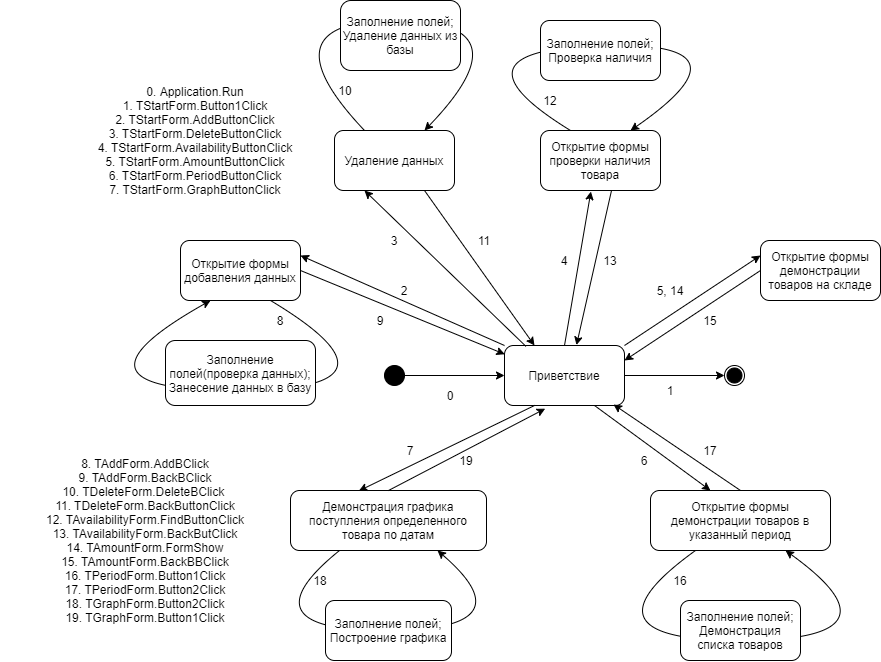
 



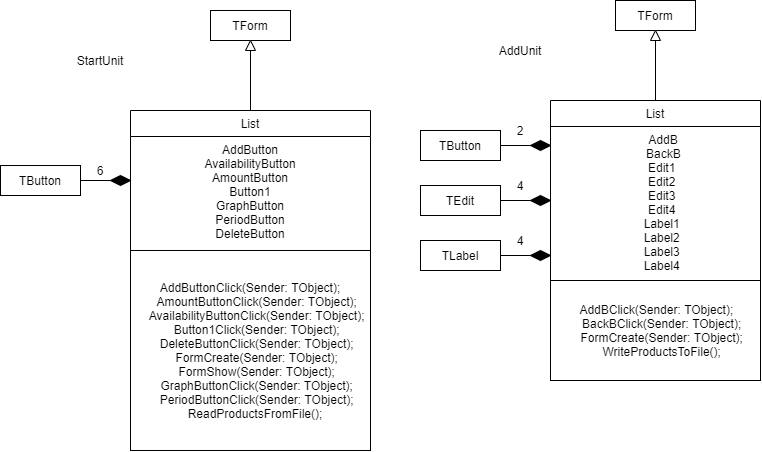
*Диаграмма состояний интерфейса:*

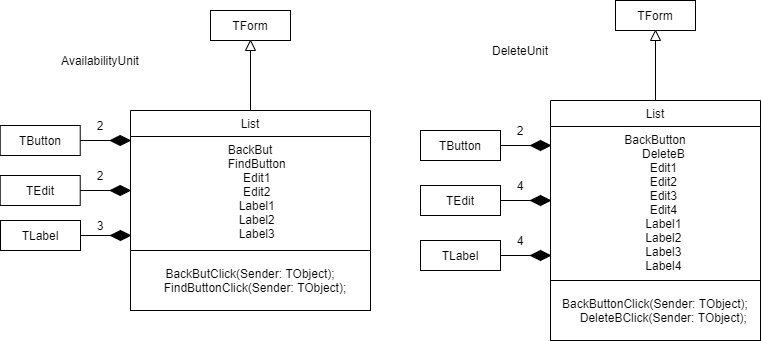
**

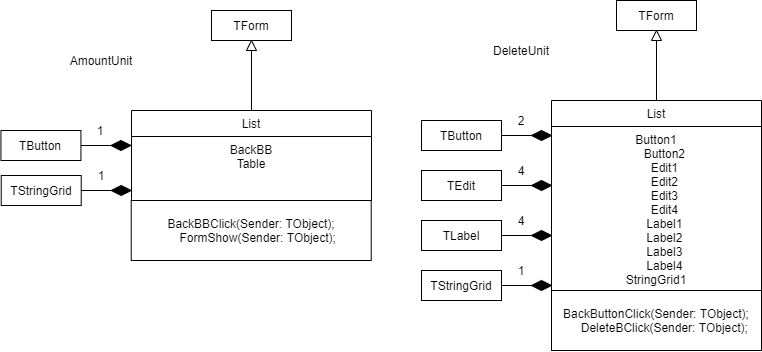
*Расширенная диаграмма состояний интерфейса для всех форм:*

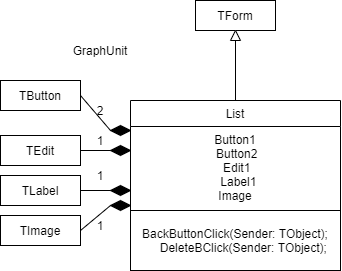
**

*Диаграмма классов интерфейсной и предметной областей:*

**

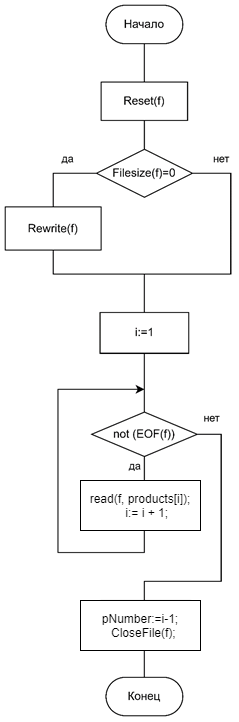
**

**

**

*Диаграмма последовательностей для одной из операций:*

procedure TStartForm.ReadProductsFromFile();

**

**Текст программы:**

program project1;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, StartUnit, AddUnit, DeleteUnit, AvailabilityUnit, AmountUnit,

PeriodUnit, GraphUnit

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TStartForm, StartForm);

Application.CreateForm(TAddForm, AddForm);

Application.CreateForm(TDeleteForm, DeleteForm);

Application.CreateForm(TAvailabilityForm, AvailabilityForm);

Application.CreateForm(TAmountForm, AmountForm);

Application.CreateForm(TPeriodForm, PeriodForm);

Application.CreateForm(TGraphForm, GraphForm);

Application.Run;

end.

unit StartUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, AddUnit, DeleteUnit, AvailabilityUnit, AmountUnit, PeriodUnit, GraphUnit;

type

{ TStartForm }

TStartForm = class(TForm)

AddButton: TButton;

AvailabilityButton: TButton;

AmountButton: TButton;

Button1: TButton;

GraphButton: TButton;

PeriodButton: TButton;

DeleteButton: TButton;

procedure AddButtonClick(Sender: TObject);

procedure AmountButtonClick(Sender: TObject);

procedure AvailabilityButtonClick(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure DeleteButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure GraphButtonClick(Sender: TObject);

procedure PeriodButtonClick(Sender: TObject);

procedure ReadProductsFromFile();

private

public

end;

TProduct = record

name : shortstring;

date : TDateTime;

count: integer;

country : shortstring;

end;

TProducts = array[1..100] of TProduct;

var

StartForm: TStartForm;

f: file of TProduct;

product: TProduct;

products: TProducts;

pNumber:word;

i:integer;

ccproduct: TProduct;

ccproducts: TProducts;

ppNumber: word;

maxcount, mincount : integer;

maxdate, mindate : TDateTime;

implementation

{$R \*.lfm}

{ TStartForm }

procedure TStartForm.AddButtonClick(Sender: TObject);

begin

AddForm.Show;

AddForm.Edit1.Setfocus

end;

procedure TStartForm.ReadProductsFromFile();

begin

Reset(f);

if Filesize(f)=0 then Rewrite(f);

i:=1;

while not (EOF(f)) do

begin

read(f, products[i]);

i:= i + 1;

end;

pNumber:=i-1;

CloseFile(f);

end;

procedure TStartForm.AmountButtonClick(Sender: TObject);

begin

AmountForm.Show;

end;

procedure TStartForm.AvailabilityButtonClick(Sender: TObject);

begin

AvailabilityForm.Show;

AvailabilityForm.Edit1.Setfocus

end;

procedure TStartForm.Button1Click(Sender: TObject);

begin

Close();

end;

procedure TStartForm.DeleteButtonClick(Sender: TObject);

begin

DeleteForm.Show;

DeleteForm.Edit1.Setfocus

end;

procedure TStartForm.FormCreate(Sender: TObject);

begin

Assignfile(f, 'products.txt');

end;

procedure TStartForm.FormShow(Sender: TObject);

begin

StartForm.ReadProductsFromFile();

end;

procedure TStartForm.GraphButtonClick(Sender: TObject);

begin

GraphForm.Show;

GraphForm.Edit1.Setfocus

end;

procedure TStartForm.PeriodButtonClick(Sender: TObject);

begin

PeriodForm.Show;

PeriodForm.Edit1.Setfocus

end;

end.

unit AddUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls;

type

{ TAddForm }

TAddForm = class(TForm)

AddB: TButton;

BackB: TButton;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

procedure AddBClick(Sender: TObject);

procedure BackBClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure WriteProductsToFile();

private

public

end;

var

AddForm: TAddForm;

implementation

uses StartUnit;

{$R \*.lfm}

{ TAddForm }

procedure TAddForm.FormCreate(Sender: TObject);

begin

pNumber := 0;

end;

procedure TAddForm.AddBClick(Sender: TObject);

var correct:boolean = true;

begin

if (Edit1.Text='') then begin

MessageDlg('Incorrect name',mtConfirmation,mbYesNoCancel,0);

correct:=false;

Edit1.Clear;

end;

if (Edit2.Text='') then begin

MessageDlg('Incorrect date',mtConfirmation,mbYesNoCancel,0);

correct:=false;

Edit2.Clear;

end;

if (Edit3.Text='') or (StrToInt(Edit3.Text) < 1) then begin

MessageDlg('Incorrect amount of product', mtConfirmation, mbYesNoCancel,0);

correct:=false;

Edit3.Clear;

end;

if (Edit4.Text='') then begin

MessageDlg('Incorrect country',mtConfirmation,mbYesNoCancel,0);

correct:=false;

Edit4.Clear;

end;

if correct then begin

StartForm.ReadProductsFromFile();

product.name := Edit1.Text;

product.date := StrToDate(Edit2.Text);

product.count := StrToInt(Edit3.Text);

product.country := Edit4.Text;

pNumber := pNumber + 1;

products[pNumber] := product;

Edit1.Clear;

Edit2.Clear;

Edit3.Clear;

Edit4.Clear;

Edit1.SetFocus;

AddForm.WriteProductsToFile();

end;

end;

procedure TAddForm.BackBClick(Sender: TObject);

begin

AddForm.Hide();

StartForm.Show();

end;

procedure TAddForm.WriteProductsToFile();

begin

rewrite(f);

for i:=1 to pNumber do begin

write(f, products[i]);

end;

CloseFile(f);

end;

end.

unit DeleteUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, AddUnit;

type

{TDeleteForm}

TDeleteForm = class(TForm)

BackButton: TButton;

DeleteB: TButton;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

procedure BackButtonClick(Sender: TObject);

procedure DeleteBClick(Sender: TObject);

private

public

end;

var

DeleteForm: TDeleteForm;

implementation

uses StartUnit;

{$R \*.lfm}

{ TDeleteForm }

procedure TDeleteForm.DeleteBClick(Sender: TObject);

var delNumber, del : integer;

delProduct, currentProduct : TProduct;

begin

product.name := Edit1.Text;

product.date := StrToDate(Edit2.Text);

product.count := StrToInt(Edit3.Text);

product.country := Edit4.Text;

Reset(f);

if Filesize(f)=0 then Rewrite(f);

i:=1;

del := 0;

while not (EOF(f)) do

begin

read(f, currentProduct);

if(currentProduct.name = product.name) and (currentProduct.date = product.date)and (currentProduct.country = product.country) then begin

if(currentProduct.count < product.count) then begin

MessageDlg('Not enough goods in stock', mtConfirmation, mbYesNoCancel,0);

end;

if(currentProduct.count > product.count) then begin

products[i].count := products[i].count - product.count;

end;

if(currentProduct.count = product.count) then begin

del := 1;

delNumber := i;

delProduct := currentProduct;

end;

end;

i:= i + 1;

end;

reset(f);

while del = 1 do

begin

read(f, currentProduct);

if (currentProduct.name = delProduct.name) and (currentProduct.date = delProduct.date)and (currentProduct.country = delProduct.country) and(currentProduct.count = delProduct.count) then begin

del := 0;

for i:= delNumber to pNumber-1 do

begin

products[i]:=products[i+1];

end;

pNumber := pNumber - 1;

end;

end;

AddForm.WriteProductsToFile();

Edit1.Clear;

Edit2.Clear;

Edit3.Clear;

Edit4.Clear;

Edit1.SetFocus;

end;

procedure TDeleteForm.BackButtonClick(Sender: TObject);

begin

DeleteForm.Hide();

StartForm.Show();

end;

end.

unit AvailabilityUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, AddUnit;

type

{ TAvailabilityForm }

TAvailabilityForm = class(TForm)

BackBut: TButton;

FindButton: TButton;

Edit1: TEdit;

Edit2: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

procedure BackButClick(Sender: TObject);

procedure FindButtonClick(Sender: TObject);

private

public

end;

var

AvailabilityForm: TAvailabilityForm;

implementation

uses StartUnit;

{$R \*.lfm}

{ TAvailabilityForm }

procedure TAvailabilityForm.FindButtonClick(Sender: TObject);

var curProduct: TProduct;

dell: integer;

begin

product.name := Edit1.Text;

product.country := Edit2.Text;

dell := 0;

reset(f);

while not (EOF(f)) do

begin

read(f, curProduct);

if (curProduct.name = product.name) and (curProduct.country = product.country) then dell := 1;

end;

if dell = 1 then MessageDlg('Yes, are available', mtConfirmation, mbYesNoCancel,0);

if dell = 0 then MessageDlg('No, are not available', mtConfirmation, mbYesNoCancel,0);

end;

procedure TAvailabilityForm.BackButClick(Sender: TObject);

begin

AvailabilityForm.Hide();

StartForm.Show();

end;

end.

unit AmountUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, Grids, StdCtrls, AddUnit;

type

{ TAmountForm }

TAmountForm = class(TForm)

BackBB: TButton;

Table: TStringGrid;

procedure BackBBClick(Sender: TObject);

procedure FormShow(Sender: TObject);

private

public

end;

var

AmountForm: TAmountForm;

implementation

uses StartUnit;

{$R \*.lfm}

{ TAmountForm }

procedure TAmountForm.BackBBClick(Sender: TObject);

begin

AmountForm.Hide();

StartForm.Show();

end;

procedure TAmountForm.FormShow(Sender: TObject);

begin

Table.RowCount := pNumber + 1;

for i:=1 to pNumber do begin

Table.Cells[0, i]:= products[i].name;

Table.Cells[1, i]:= IntToStr(products[i].count);

end;

end;

end.

unit PeriodUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Grids, AddUnit;

type

{ TPeriodForm }

TPeriodForm = class(TForm)

Button1: TButton;

Button2: TButton;

Edit1: TEdit;

Edit2: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

StringGrid1: TStringGrid;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormShow(Sender: TObject);

private

public

end;

var

PeriodForm: TPeriodForm;

implementation

uses StartUnit;

{$R \*.lfm}

{ TPeriodForm }

procedure TPeriodForm.Button1Click(Sender: TObject);

var t1, t2: TDateTime;

cproduct : TProduct;

begin

t1:= StrToDate(Edit1.Text);

t2:= StrToDate(Edit2.TExt);

reset(f);

i := 1;

while not (EOF(f)) do

begin

read(f, cproduct);

if(cproduct.date >= t1) and (cproduct.date <= t2) then begin

StringGrid1.RowCount := StringGrid1.RowCount + 1;

StringGrid1.Cells[0, StringGrid1.RowCount - 1]:= DateToStr(products[i].date);

StringGrid1.Cells[1, StringGrid1.RowCount - 1]:= products[i].name;

StringGrid1.Cells[2, StringGrid1.RowCount - 1]:= products[i].country;

end;

i:= i + 1;

end;

end;

procedure TPeriodForm.Button2Click(Sender: TObject);

begin

PeriodForm.Hide();

StartForm.Show();

end;

procedure TPeriodForm.FormShow(Sender: TObject);

begin

StringGrid1.RowCount := 1;

end;

end.

unit GraphUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls, AddUnit;

type

{ TGraphForm }

TGraphForm = class(TForm)

Button1: TButton;

Button2: TButton;

Edit1: TEdit;

Image: TImage;

Label1: TLabel;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure FormShow(Sender: TObject);

private

public

end;

var

GraphForm: TGraphForm;

x0,y0,x1,y1,xmin,xmax,ymin,ymax, dx, dy, x, y: integer;

daymax, daymin, yearmax, yearmin, monthmax, monthmin, yearc, monthc, dayc: word;

implementation

uses StartUnit;

{$R \*.lfm}

{ TGraphForm }

procedure TGraphForm.Button1Click(Sender: TObject);

begin

GraphForm.Hide();

StartForm.Show();

end;

procedure TGraphForm.Button2Click(Sender: TObject);

begin

ppNumber := 0;

maxcount:= 0;

mincount:= 9999999;

maxdate:=StrToDate('30.12.2000');

mindate:=StrToDate('30.12.2222');

product.name := Edit1.Text;

Reset(f);

while not (EOF(f)) do

begin

read(f, ccproduct);

if(ccProduct.name = product.name) then begin

ppNumber := ppNumber + 1;

ccproducts[ppNumber] := ccproduct;

if(ccproduct.date < mindate) then mindate:=ccproduct.date;

if(ccproduct.date > maxdate) then maxdate:=ccproduct.date;

if(ccproduct.count < mincount) then mincount:=ccproduct.count;

if(ccproduct.count > maxcount) then maxcount:=ccproduct.count;

end;

end;

if(ppNumber < 2) then begin

MessageDlg('Not enough incomes!',mtConfirmation,mbYesNoCancel,0);

exit();

end;

x0:=Image.Width div 10;

x1:=Image.Width div 10\*9;

xmin:=Image.Width div 10\*2;

xmax:=Image.Width div 10\*8;

y0:=Image.Height\*9 div 10;

y1:=Image.Height div 10;

ymin:=Image.Height\*8 div 10;

ymax:=Image.Height\*2 div 10;

DecodeDate(maxdate, yearmax, monthmax, daymax);

DecodeDate(mindate, yearmin, monthmin, daymin);

dx:= (xmax - xmin) div (((((yearmax-1)\*12)+monthmax-1)\*30+daymax) - ((((yearmin-1)\*12)+monthmin-1)\*30+daymin));

dy:= (ymin - ymax) div (maxcount - mincount);

Image.Canvas.Pen.Color:=clBlack;

Image.Canvas.MoveTo(x0,y0);

Image.Canvas.LineTo(x0,y1);

Image.Canvas.MoveTo(x0,y0);

Image.Canvas.LineTo(x1,y0);

Image.Canvas.TextOut(x0,y1,'amount');

Image.Canvas.TextOut(x1,y0,'date');

for i:=1 to ppNumber do begin

ccproduct := ccproducts[i];

DecodeDate(ccproduct.date, yearc, monthc, dayc);

x:= dx\*(((((yearc-1)\*12)+monthc-1)\*30+dayc) - ((((yearmin-1)\*12)+monthmin-1)\*30+daymin));

y:= dy\*(ccproducts[i].count - mincount);

if(i > 1) then Image.Canvas.LineTo(x + xmin ,ymin - y);

Image.Canvas.MoveTo(x + xmin, ymin - y);

Image.Canvas.Brush.Color:=clBlack;

Image.Canvas.Ellipse(x + xmin -3,ymin - y -3,x +xmin +3,ymin - y +3);

Image.Canvas.Brush.Color:=clWhite;

Image.Canvas.TextOut(x + xmin - Image.Canvas.TextWidth(DateToStr(ccproduct.date)) div 2, y0 + Image.Canvas.TextHeight(DateToStr(ccproduct.date)) div 2, DateToStr(ccproduct.date));

Image.Canvas.TextOut(x0 - Image.Canvas.TextWidth(IntToStr(ccproduct.count))\*2, ymin - y - Image.Canvas.TextHeight(DateToStr(ccproduct.date)) div 2, IntToStr(ccproduct.count));

Image.Canvas.MoveTo(x + xmin, ymin - y);

end;

end;

procedure TGraphForm.FormActivate(Sender: TObject);

begin

Image.Canvas.Brush.Color:=clWhite;

Image.Canvas.FillRect(0,0,Width,Height);

end;

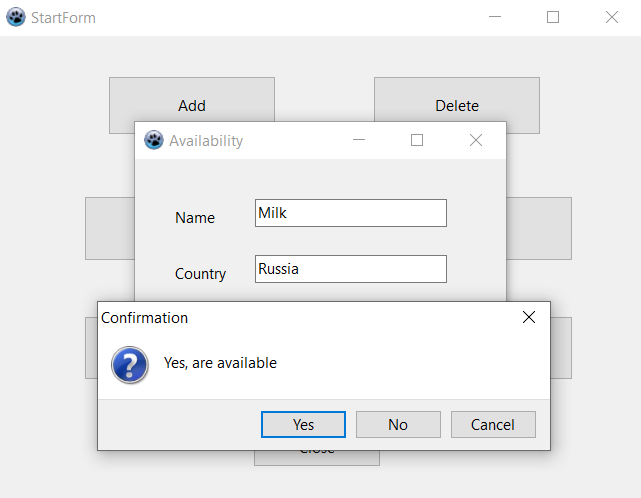
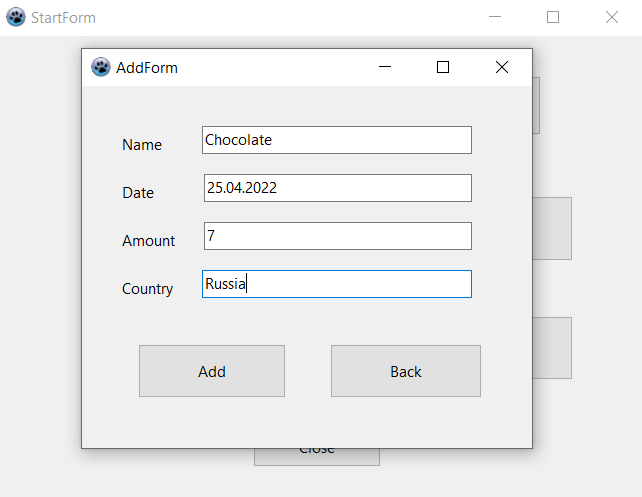
procedure TGraphForm.FormShow(Sender: TObject);

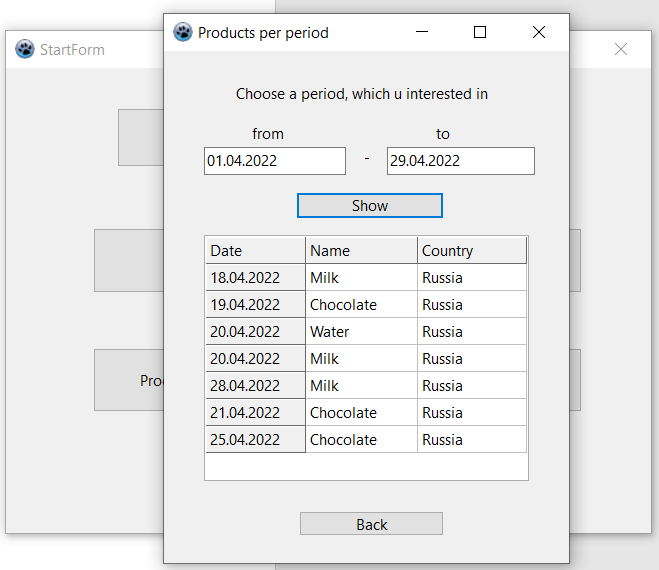
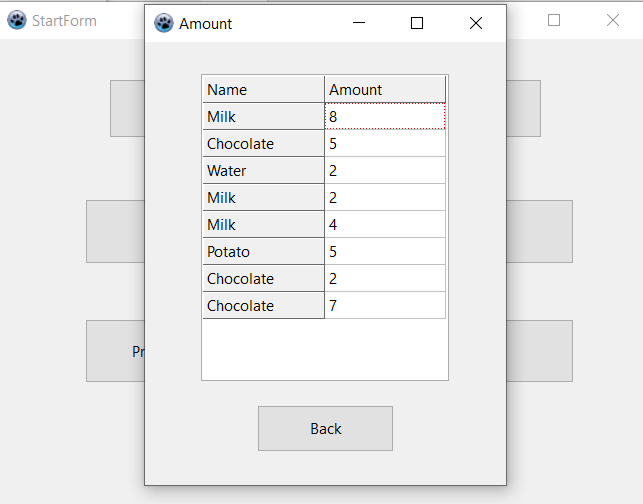
begin

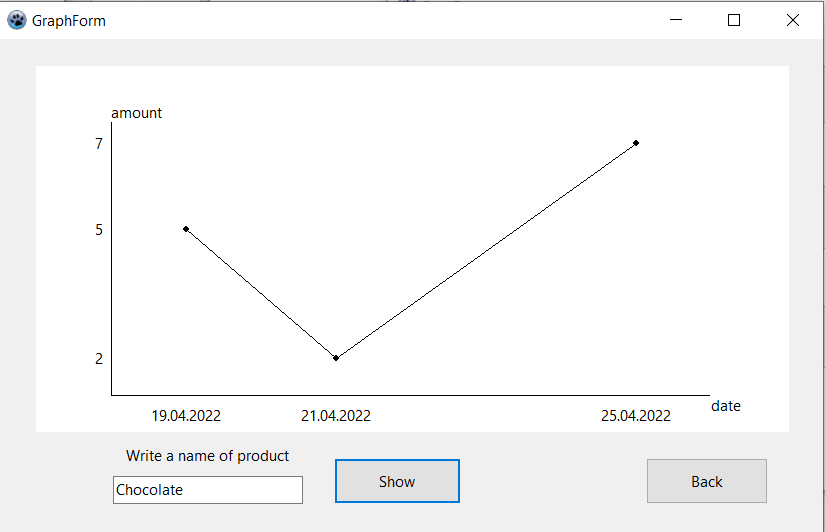
end;

end.

**Результаты тестирования:**

****

****

****

**Вывод:** Получил навыки создания небольшой программной системы с оконным интерфейсом (проектирование, отладка и тестирование). Разработал программу, работающую в паре с базой данных (файлом), содержащем сведения о товарах. Программа в интерактивном режиме формирует файл, добавляет и удаляет данные, а также воспринимает некоторый ряд запросов и выдает на них ответы.

**Задача 2**

**Анализ задания**

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

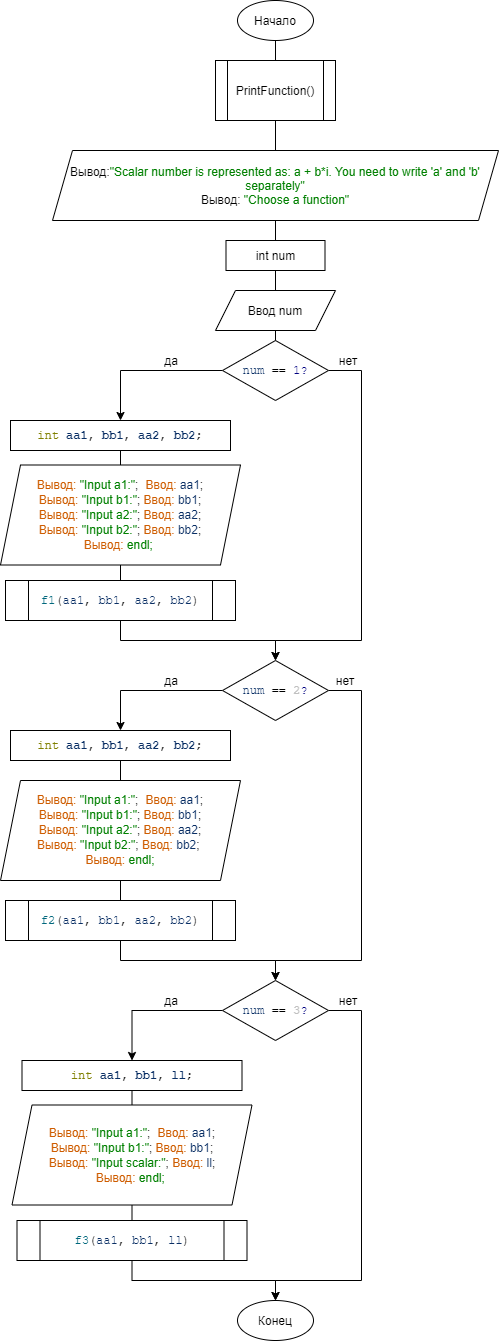
*Разработать программу, которая реализует операции над комплексными числами. Реализовать следующие операции: ввод чисел, их суммирование, вычитание и умножение на скаляр, а также вывод результатов операций на экран.*

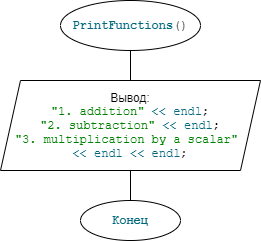
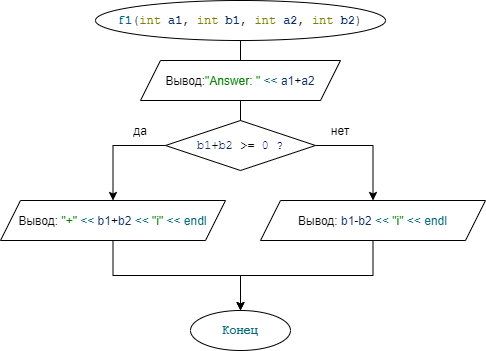
**Схемы**

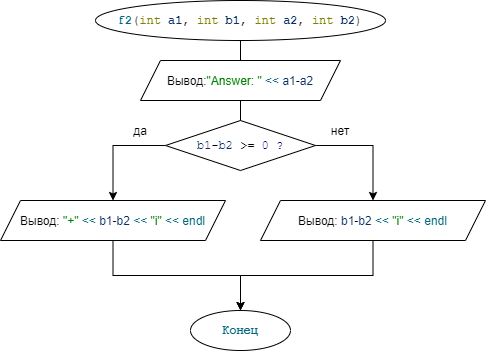
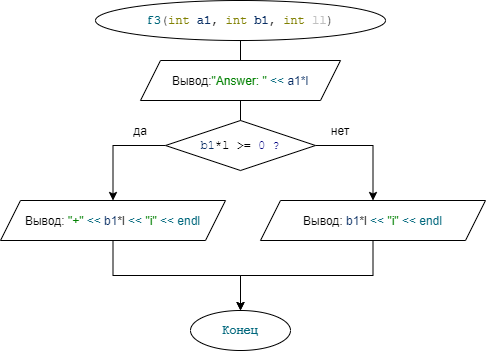
*Структурная декомпозиция:*



*Схема алгоритма:*



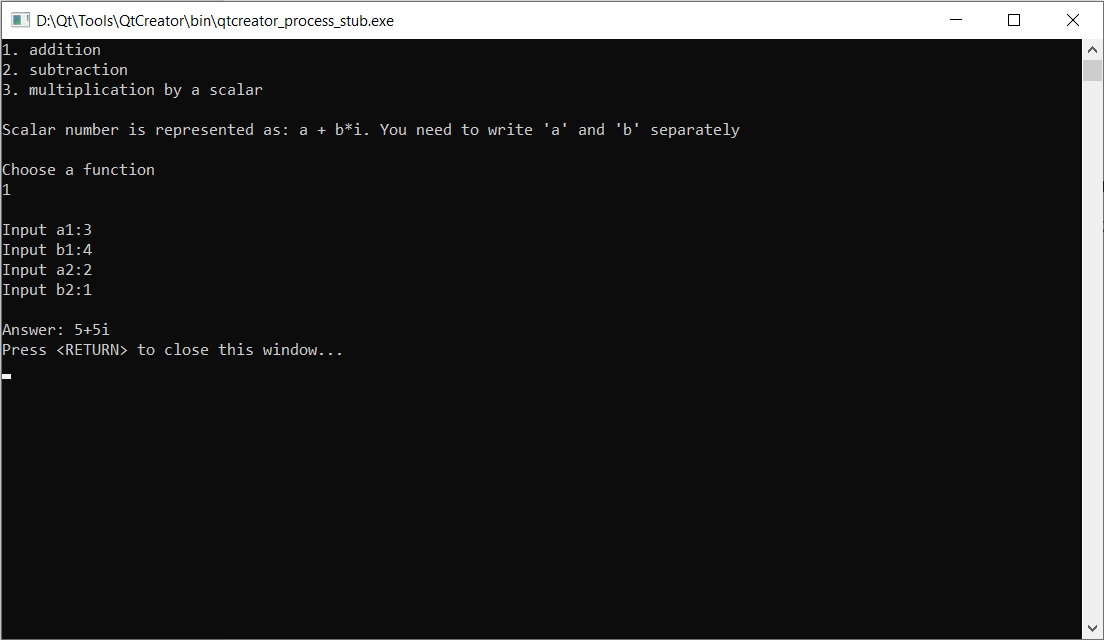
*Диаграмма состояний интерфейса:*

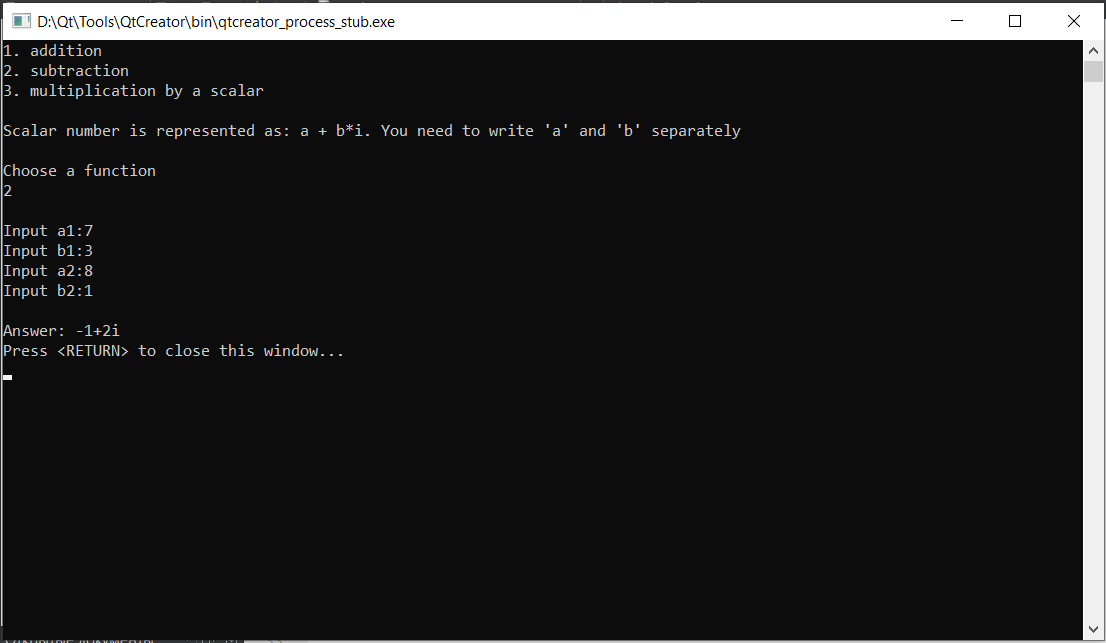
C:\Users\vasha\Desktop\птп\2\диаграмма состояний.png

**Код программы:**

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  void **PrintFunctions**(){  cout << "1. addition" << endl;  cout << "2. subtraction" << endl;  cout << "3. multiplication by a scalar" << endl << endl;  }  void **f1**(int **a1**, int **b1**, int **a2**, int **b2**)  {  cout << "Answer: " << a1+a2;  if (b1+b2 >= 0) cout << "+" << b1+b2 << "i" << endl;  else cout << b1+b2 << "i" << endl;  }  void **f2**(int **a1**, int **b1**, int **a2**, int **b2**)  {  cout << "Answer: " << a1-a2;  if (b1-b2 >= 0) cout << "+" << b1-b2 << "i" << endl;  else cout << b1-b2 << "i" << endl;  }  void **f3**(int **a1**, int **b1**, int **l**)  {  cout << "Answer: " << a1\*l;  if (b1\*l >= 0) cout << "+" << b1\*l << "i" << endl;  else cout << b1\*l << "i" << endl;  }  int **main**()  {  PrintFunctions();  cout << "Scalar number is represented as: a + b\*i. You need to write 'a' and 'b' separately" << endl << endl;  cout << "Choose a function" << endl;  int **num**;  cin >> num;  cout << endl;  if(num == 1)  {  int **aa1**, **bb1**, **aa2**, **bb2**;  cout << "Input a1:"; cin >> aa1;  cout << "Input b1:"; cin >> bb1;  cout << "Input a2:"; cin >> aa2;  cout << "Input b2:"; cin >> bb2; cout << endl;  f1(aa1, bb1, aa2, bb2);  }  if(num == 2)  {  int **aa1**, **bb1**, **aa2**, **bb2**;  cout << "Input a1:"; cin >> aa1;  cout << "Input b1:"; cin >> bb1;  cout << "Input a2:"; cin >> aa2;  cout << "Input b2:"; cin >> bb2; cout << endl;  f2(aa1, bb1, aa2, bb2);  }  if(num == 3)  {  int **aa1**, **bb1**, **ll**;  cout << "Input a1:"; cin >> aa1;  cout << "Input b1:"; cin >> bb1;  cout << "Input scalar:"; cin >> ll; cout << endl;  f3(aa1, bb1, ll);  }  return 0;  } |

**Результаты тестирования:**





**Вывод:**

Получил навыки создания небольшой программной системы с консольным интерфейсом на C++. Разработалпрограмму, которая в консольном режиме позволяет вычислять интеграл функции на введенном интервале и с заданной точностью.

**Задача 3**

**Анализ задания**

База данных (файл) магазина хозяйственных товаров содержит сведения о поступлении товаров: наименование, дата поступления, количество штук, страна-производитель. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

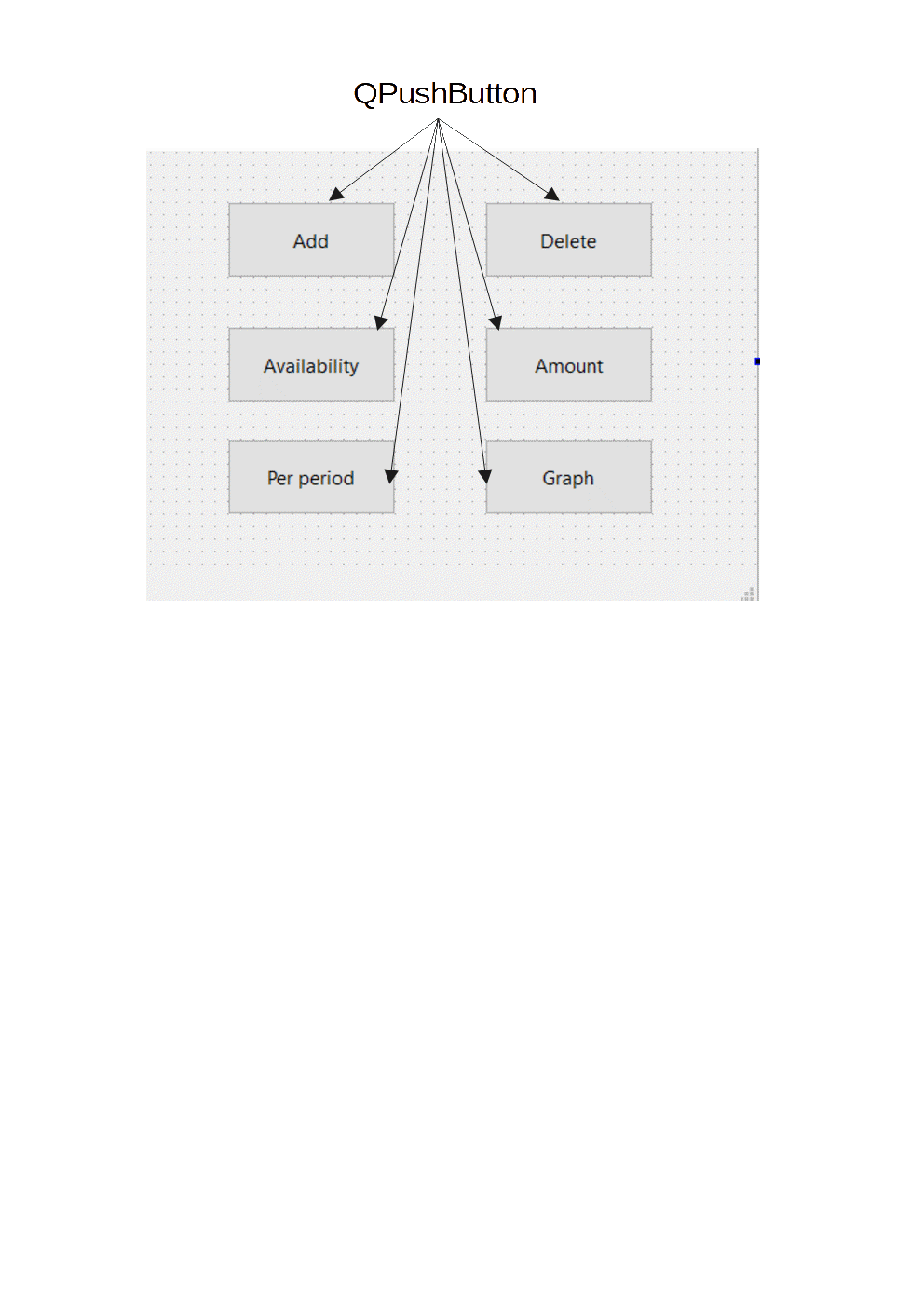
* Определить, есть ли в магазине указанный товар производства данной страны.
* Показать количество товара каждого наименования в магазине.
* Определить товары (наименование, страна), поступившие в указанный период.
* Построить график поступления заданного товара по датам.

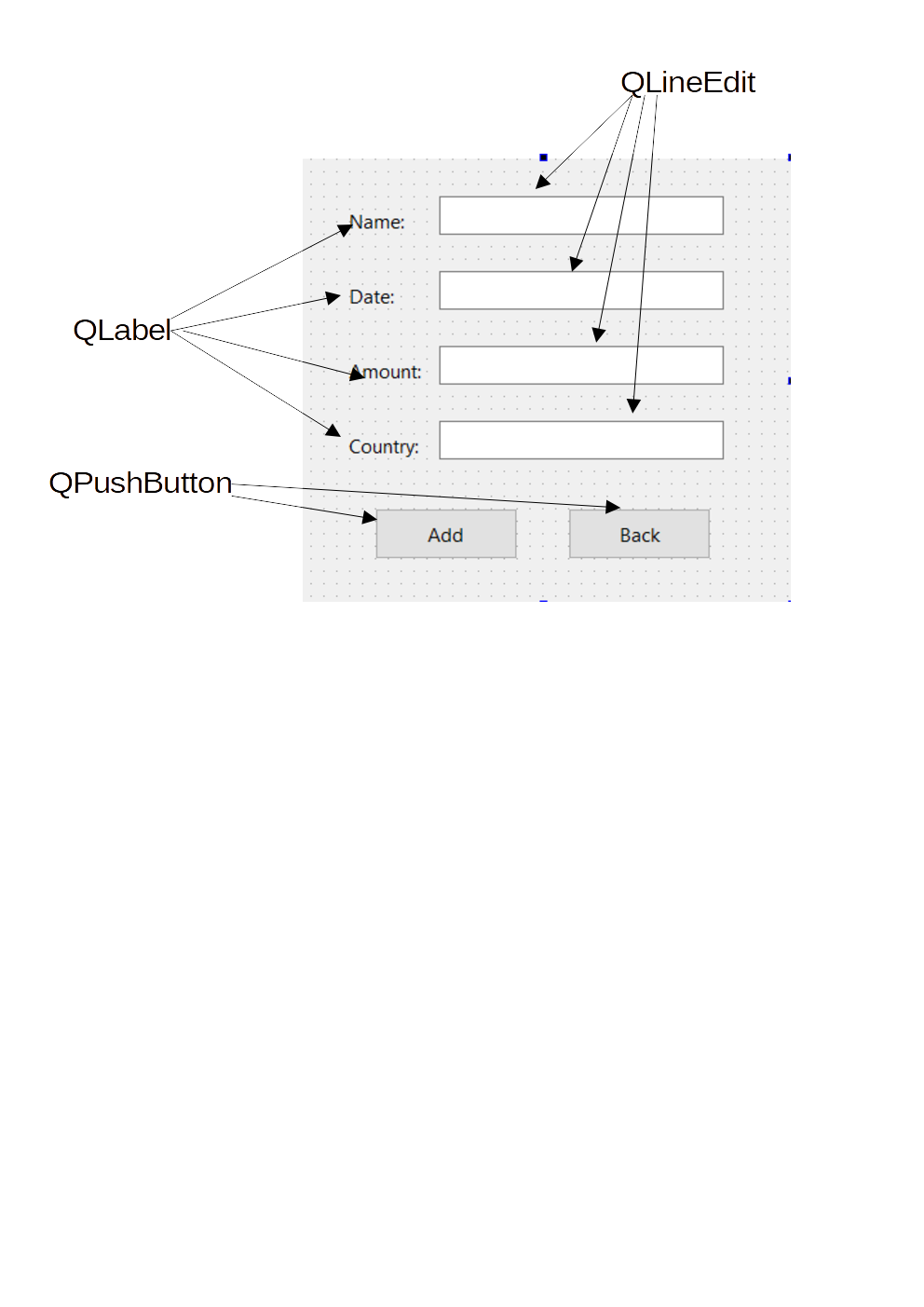
**Схемы**

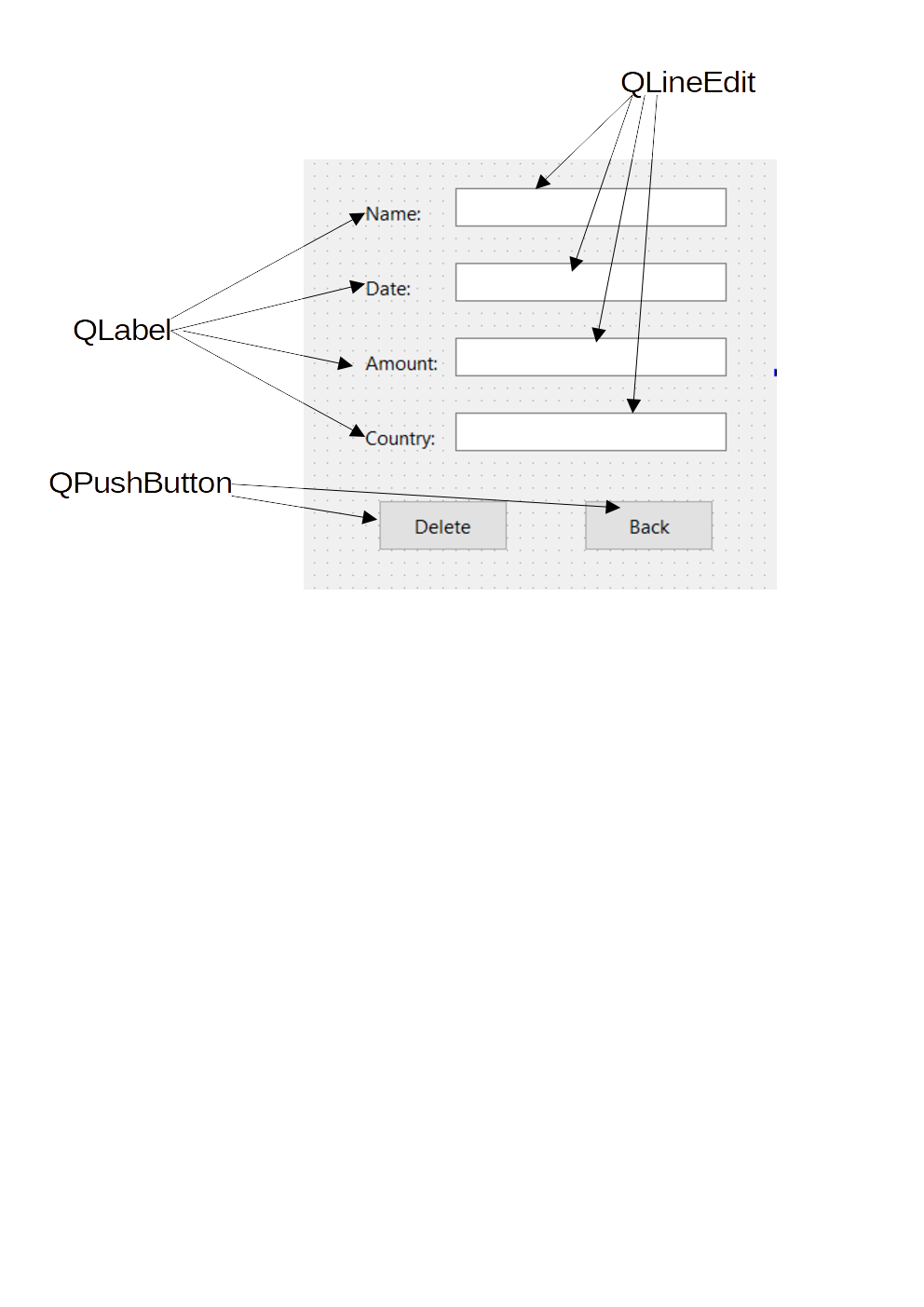
*Объектная декомпозиция:*

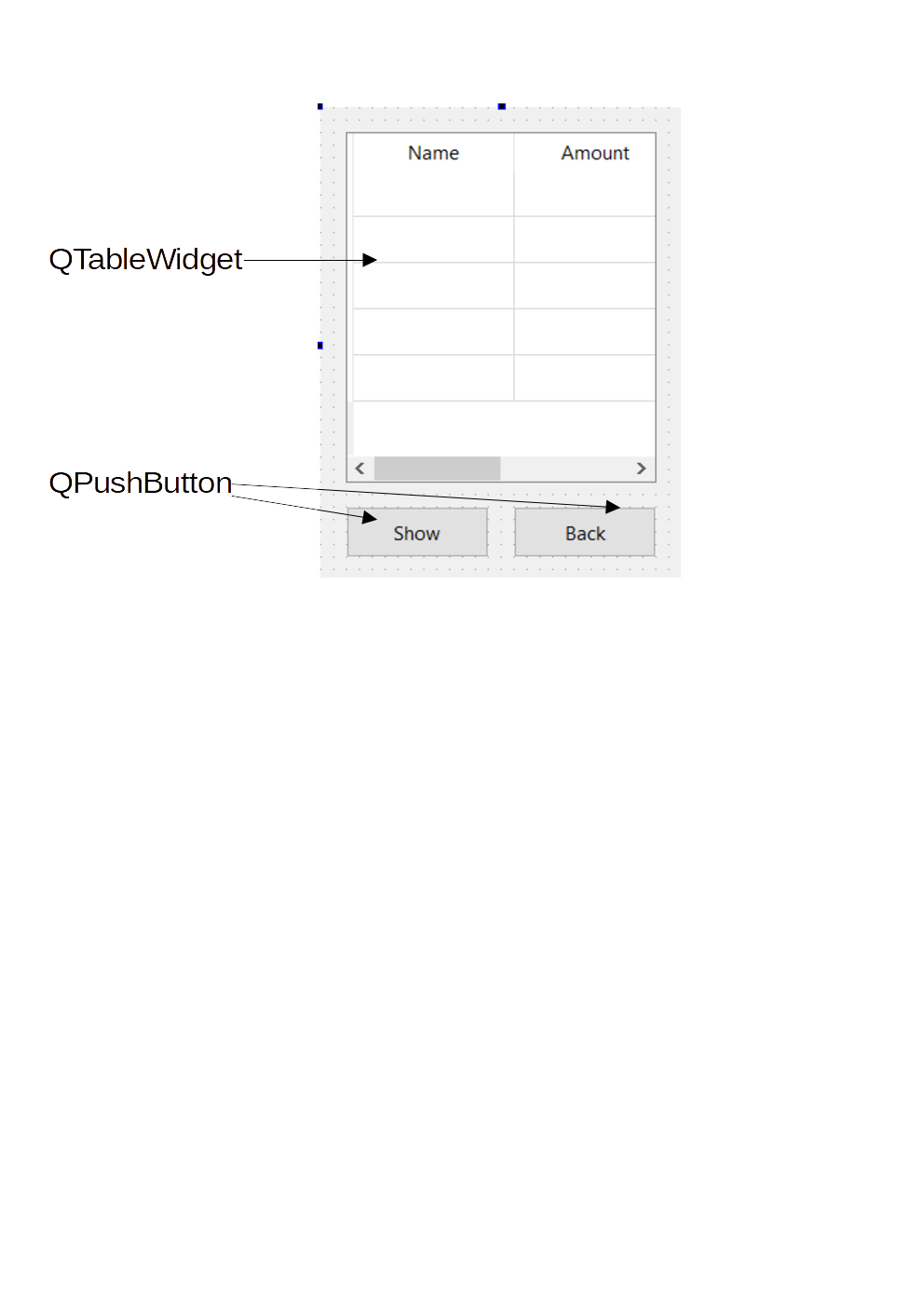


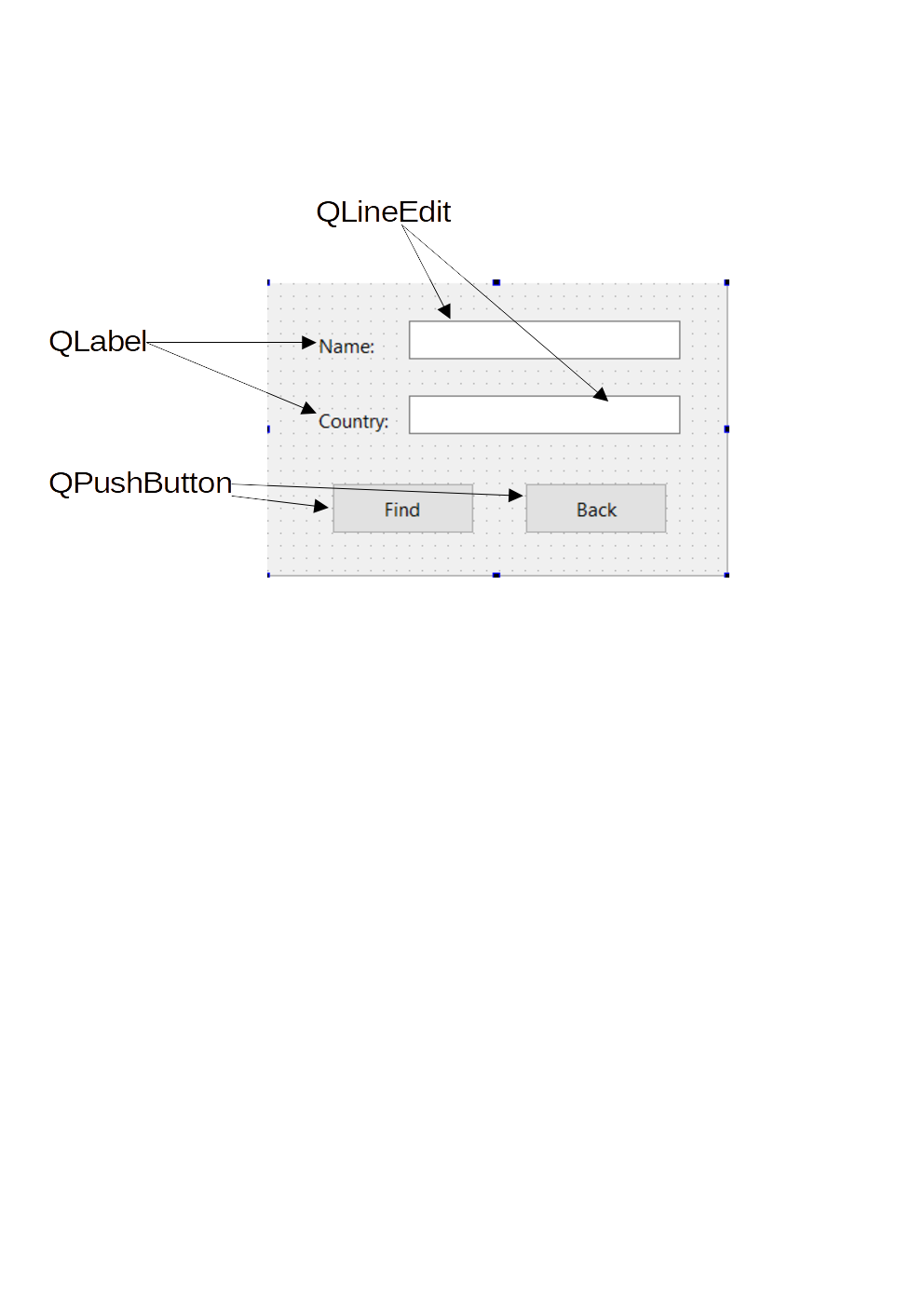
*Формы интерфейса*

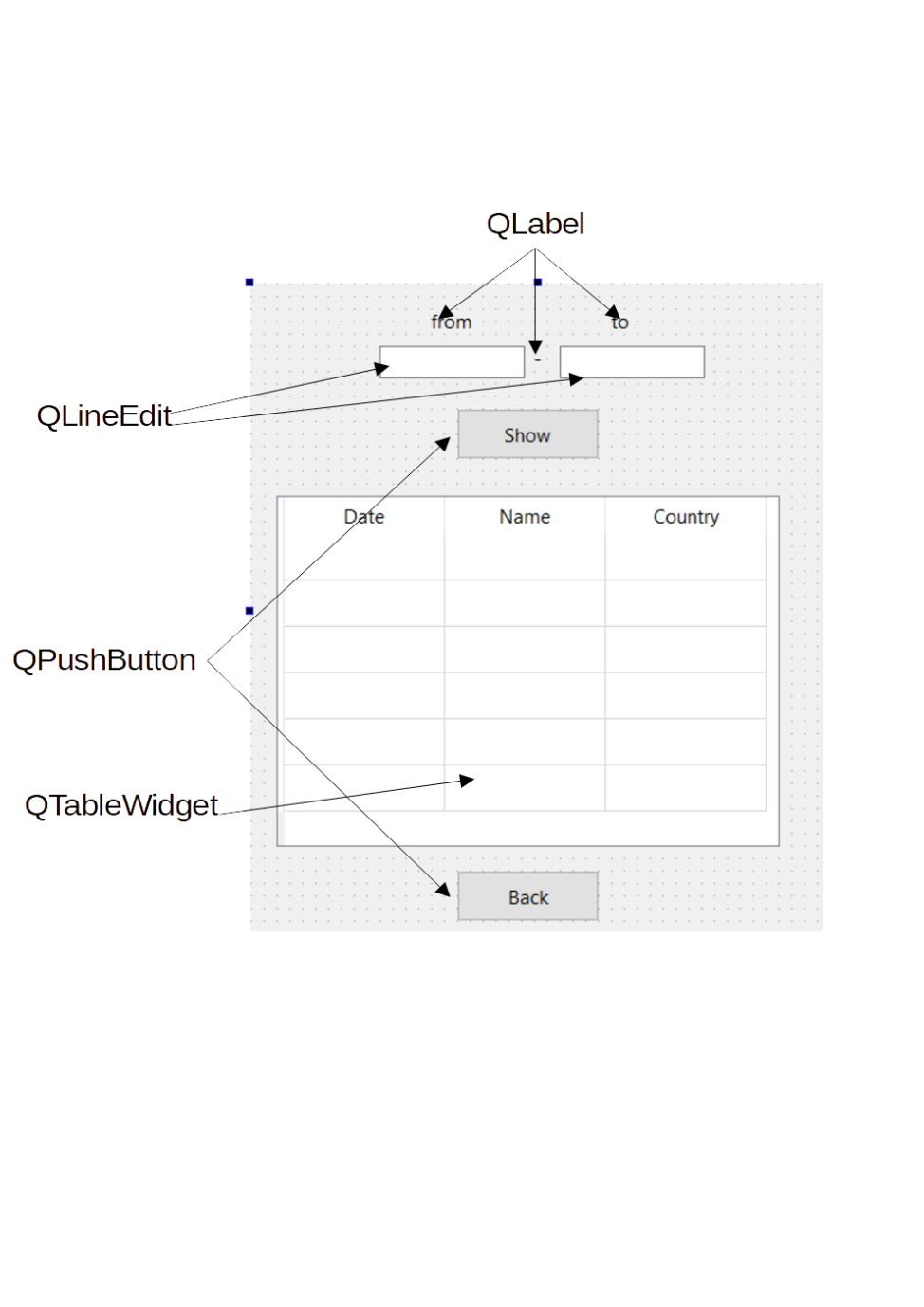
*Start Form*

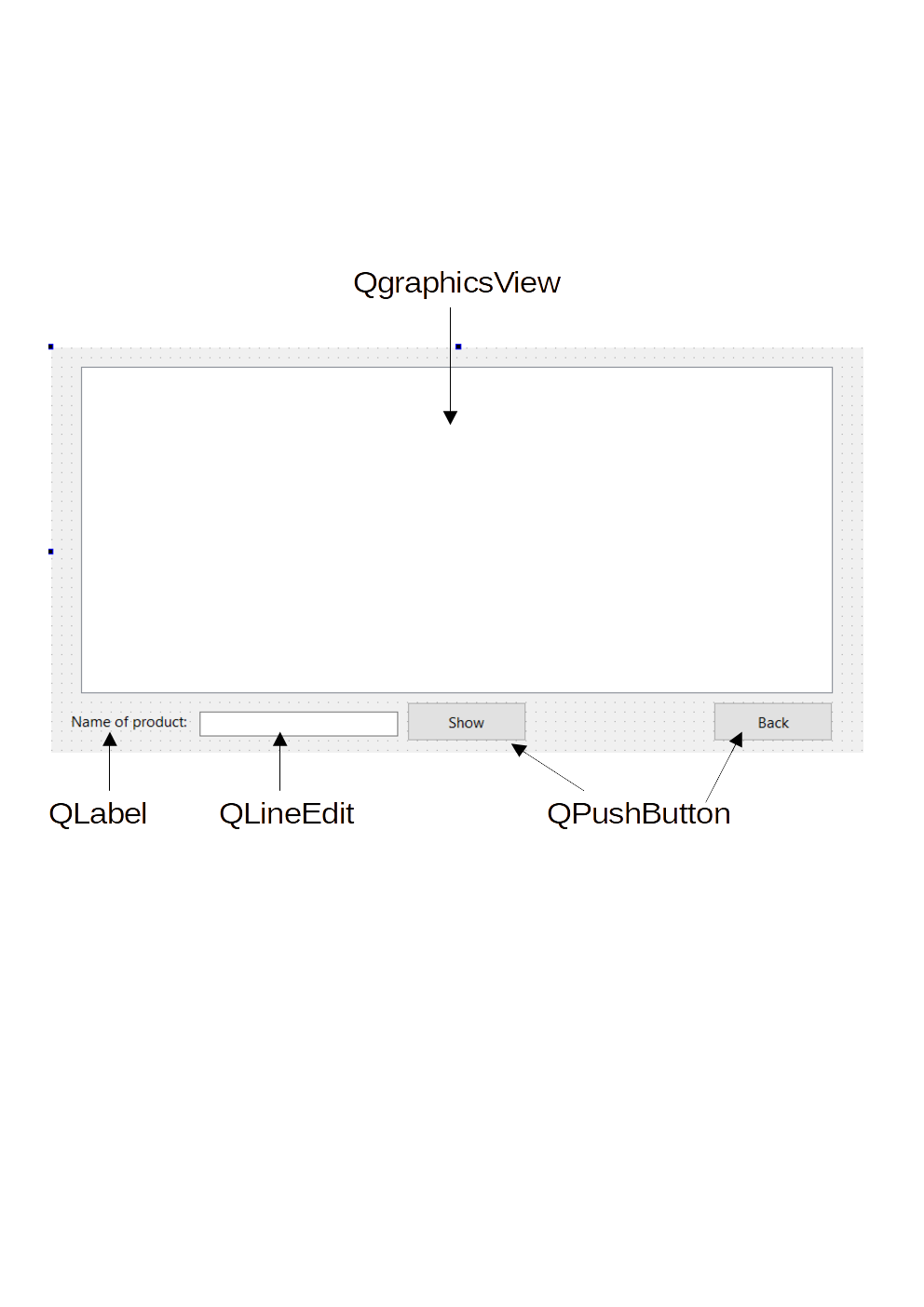
* Add Form*

* Delete Form*

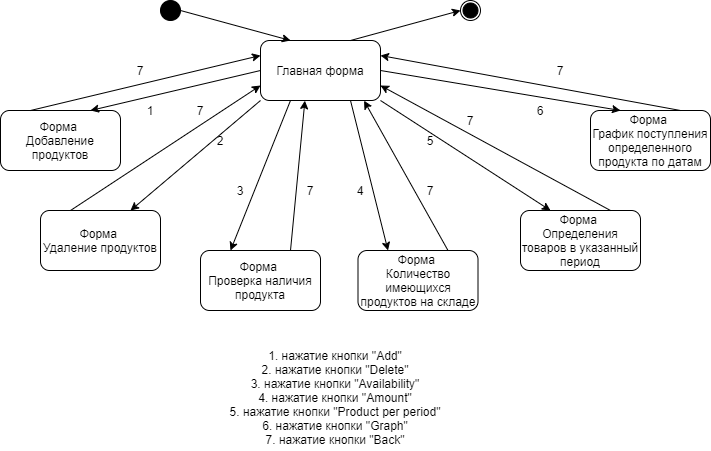
* Amount Form*

* Availability Form*

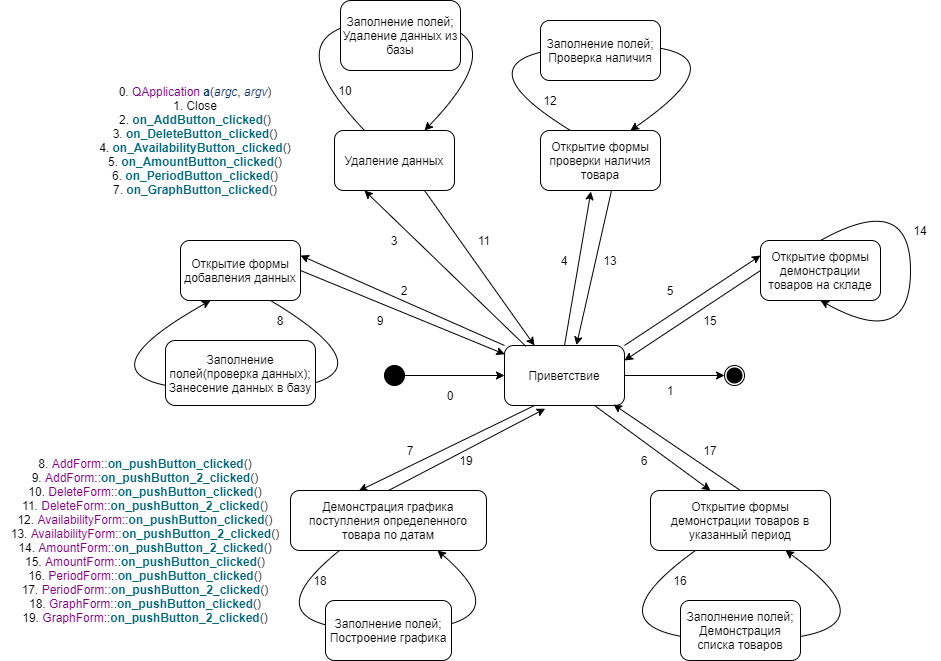
* Period Form*

* Graph Form*

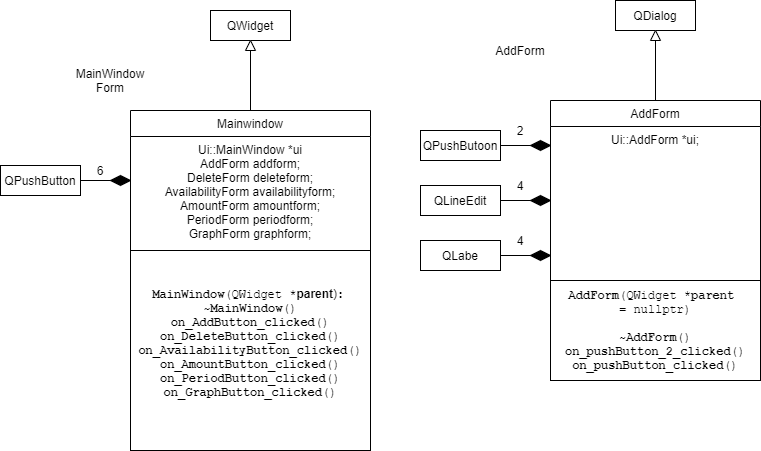
*Диаграмма состояний интерфейса*

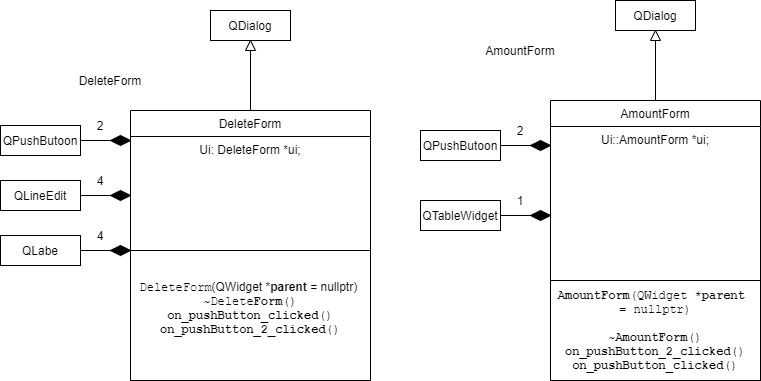
**

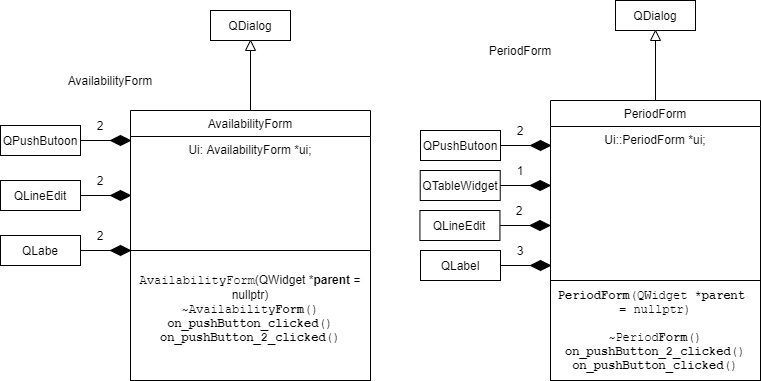
*Расширенная диаграмма состояний интерфейса для всех форм:*

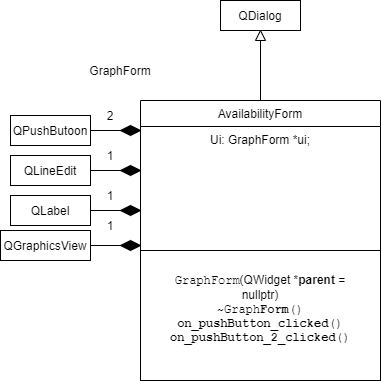
**

*Диаграмма классов интерфейсной и предметной областей:*

**

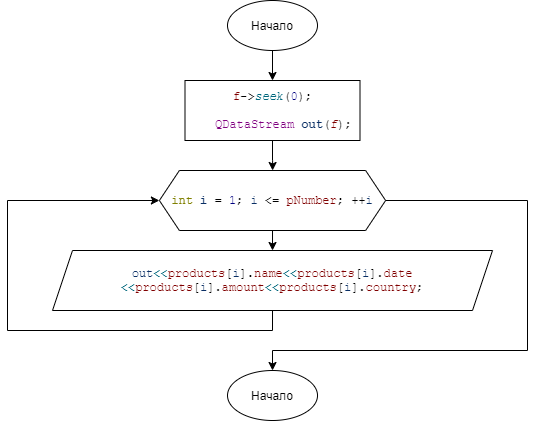
**

**

**

*Диаграмма последовательностей для одной из операций:*

void bookFile::**writeProducts**()

**

**Текст программы:**

*Mainwindow.h*

|  |
| --- |
| #ifndef MAINWINDOW\_H  #define **MAINWINDOW\_H**  #include <QMainWindow>  #include "addform.h"  #include "ui\_mainwindow.h"  #include "deleteform.h"  #include "availabilityform.h"  #include "amountform.h"  #include "periodform.h"  #include "graphform.h"  #include <QFile>  QT\_BEGIN\_NAMESPACE  namespace **Ui** { class **MainWindow**; }  QT\_END\_NAMESPACE  class **MainWindow** : public QMainWindow  {  Q\_OBJECT  public:  **MainWindow**(QWidget \***parent** = nullptr);  ~**MainWindow**();  public slots:  void **on\_AddButton\_clicked**();  void **on\_DeleteButton\_clicked**();  void **on\_AvailabilityButton\_clicked**();  void **on\_AmountButton\_clicked**();  void **on\_PeriodButton\_clicked**();  void **on\_GraphButton\_clicked**();  private:  Ui::MainWindow \***ui**;  AddForm **addform**;  DeleteForm **deleteform**;  AvailabilityForm **availabilityform**;  AmountForm **amountform**;  PeriodForm **periodform**;  GraphForm **graphform**;  };  #endif // MAINWINDOW\_H |

*Mainwindow.cpp*

|  |
| --- |
| #include "mainwindow.h"  #include "addform.h"  #include "deleteform.h"  MainWindow::**MainWindow**(QWidget \***parent**)  : QMainWindow(*parent*)  , ui(new Ui::MainWindow)  {  ui->setupUi(this);  }  MainWindow::~**MainWindow**()  {  delete ui;  }  void MainWindow::**on\_AddButton\_clicked**()  {  addform.show();  }  void MainWindow::**on\_DeleteButton\_clicked**()  {  deleteform.show();  }  void MainWindow::**on\_AvailabilityButton\_clicked**()  {  availabilityform.show();  }  void MainWindow::**on\_AmountButton\_clicked**()  {  amountform.show();  }  void MainWindow::**on\_PeriodButton\_clicked**()  {  periodform.show();  }  void MainWindow::**on\_GraphButton\_clicked**()  {  graphform.show();  } |

*Addform.h*

|  |
| --- |
| #ifndef ADDFORM\_H  #define **ADDFORM\_H**  #include <QDialog>  #include "bookfile.h"  namespace **Ui** {  class **AddForm**;  }  class **AddForm** : public QDialog  {  Q\_OBJECT  public:  explicit **AddForm**(QWidget \***parent** = nullptr);  ~**AddForm**();  private slots:  void **on\_pushButton\_2\_clicked**();  void **on\_pushButton\_clicked**();  private:  Ui::AddForm \***ui**;  };  #endif // ADDFORM\_H |

*Addform.cpp*

|  |
| --- |
| #include "addform.h"  #include "ui\_addform.h"  #include <QString>  #include <QDate>  #include "mainwindow.h"  #include "bookfile.h"  #include <bits/stdc++.h>  AddForm::**AddForm**(QWidget \***parent**) :  QDialog(*parent*),  ui(new Ui::AddForm)  {  ui->setupUi(this);  }  AddForm::~**AddForm**()  {  delete ui;  }  void AddForm::**on\_pushButton\_2\_clicked**()  {  this->close();  }  void AddForm::**on\_pushButton\_clicked**()  {  bookFile **book**;  productType **pr**;  pr.name = ui->lineEdit->text();  pr.date = QDate::fromString(ui->lineEdit\_2->text(), "dd.MM.yyyy");  QString **s**;  s = ui->lineEdit\_3->text();  pr.amount = s.toInt();  pr.country = ui->lineEdit\_4->text();  book.addProduct(pr);  ui->lineEdit->clear();  ui->lineEdit\_2->clear();  ui->lineEdit\_3->clear();  ui->lineEdit\_4->clear();  } |

*Deleteform.h*

|  |
| --- |
| #ifndef DELETEFORM\_H  #define **DELETEFORM\_H**  #include <QDialog>  namespace **Ui** {  class **DeleteForm**;  }  class **DeleteForm** : public QDialog  {  Q\_OBJECT  public:  explicit **DeleteForm**(QWidget \***parent** = nullptr);  ~**DeleteForm**();  public slots:  void **on\_pushButton\_2\_clicked**();  void **on\_pushButton\_clicked**();  private:  Ui::DeleteForm \***ui**;  };  #endif // DELETEFORM\_H |

*Deleteform.cpp*

|  |
| --- |
| #include "deleteform.h"  #include "ui\_deleteform.h"  #include <QString>  #include <QDate>  #include "mainwindow.h"  #include "bookfile.h"  #include <bits/stdc++.h>  DeleteForm::**DeleteForm**(QWidget \***parent**) :  QDialog(*parent*),  ui(new Ui::DeleteForm)  {  ui->setupUi(this);  }  DeleteForm::~**DeleteForm**()  {  delete ui;  }  void DeleteForm::**on\_pushButton\_2\_clicked**()  {  this->close();  }  void DeleteForm::**on\_pushButton\_clicked**()  {  bookFile **book**;  productType **pr**;  pr.name = ui->lineEdit->text();  pr.date = QDate::fromString(ui->lineEdit\_2->text(), "dd.MM.yyyy");  QString **s**;  s = ui->lineEdit\_3->text();  pr.amount = s.toInt();  pr.country = ui->lineEdit\_4->text();  book.deleteProduct(pr);  ui->lineEdit->clear();  ui->lineEdit\_2->clear();  ui->lineEdit\_3->clear();  ui->lineEdit\_4->clear();  } |

*Amountform.h*

|  |
| --- |
| #ifndef AMOUNTFORM\_H  #define **AMOUNTFORM\_H**  #include <QDialog>  namespace **Ui** {  class **AmountForm**;  }  class **AmountForm** : public QDialog  {  Q\_OBJECT  public:  explicit **AmountForm**(QWidget \***parent** = nullptr);  ~**AmountForm**();  private slots:  void **on\_pushButton\_clicked**();  void **on\_pushButton\_2\_clicked**();  private:  Ui::AmountForm \***ui**;  };  #endif // AMOUNTFORM\_H |

*Amountform.cpp*

|  |
| --- |
| #include "amountform.h"  #include "ui\_amountform.h"  #include <QString>  #include <QDate>  #include "mainwindow.h"  #include "bookfile.h"  #include <bits/stdc++.h>  #include <QTabWidget>  AmountForm::**AmountForm**(QWidget \***parent**) :  QDialog(*parent*),  ui(new Ui::AmountForm)  {  ui->setupUi(this);  }  AmountForm::~**AmountForm**()  {  delete ui;  }  void AmountForm::**on\_pushButton\_clicked**()  {  this->close();  }  void AmountForm::**on\_pushButton\_2\_clicked**()  {  struct **productType**  {  QString **name**, **country**;  int **amount**;  QDate **date**;  };  QFile \***f**;  productType **products**[100];  int **pNumber**;  productType **product**;  f = new QFile("f.txt");  f->*open*(QFile::ReadWrite);  pNumber = 0;  int **i** = 1;  f->*seek*(0);  QDataStream **in**(*f*);  while(!in.atEnd())  {  in>>product.name>>product.date>>product.amount>>product.country;  products[i].name = product.name;  products[i].date = product.date;  products[i].amount = product.amount;  products[i].country = product.country;  i++;  }  pNumber = i - 1;  for(int **i** = 1; i <= pNumber; ++i)  {  QTableWidgetItem \***item**; // элемент таблицы  item = new QTableWidgetItem(); // создаем элемент  item->setFlags(Qt::NoItemFlags);//запрещаем  // выделение  item->setText(products[i].name); // записываем текст  ui->tableWidget->setItem(i-1,0,*item*);  // к таблице  item = new QTableWidgetItem();// создаем элемент  item->setFlags(Qt::NoItemFlags); //запрещаем  // выделение  item->setText(QString::number(products[i].amount));  ui->tableWidget->setItem(i-1,1,*item*);  }  } |

*Availabilityform.h*

|  |
| --- |
| #ifndef AVAILABILITYFORM\_H  #define **AVAILABILITYFORM\_H**  #include <QDialog>  namespace **Ui** {  class **AvailabilityForm**;  }  class **AvailabilityForm** : public QDialog  {  Q\_OBJECT  public:  explicit **AvailabilityForm**(QWidget \***parent** = nullptr);  ~**AvailabilityForm**();  private slots:  void **on\_pushButton\_2\_clicked**();  void **on\_pushButton\_clicked**();  private:  Ui::AvailabilityForm \***ui**;  };  #endif // AVAILABILITYFORM\_H |

*Availabilityform.cpp*

|  |
| --- |
| #include "availabilityform.h"  #include "ui\_availabilityform.h"  #include <QString>  #include <QDate>  #include "mainwindow.h"  #include "bookfile.h"  #include <bits/stdc++.h>  #include <QMessageBox>  AvailabilityForm::**AvailabilityForm**(QWidget \***parent**) :  QDialog(*parent*),  ui(new Ui::AvailabilityForm)  {  ui->setupUi(this);  }  AvailabilityForm::~**AvailabilityForm**()  {  delete ui;  }  void AvailabilityForm::**on\_pushButton\_2\_clicked**()  {  this->close();  }  void AvailabilityForm::**on\_pushButton\_clicked**()  {  bookFile **book**;  QString **nn**, **cc**;  nn = ui->lineEdit->text();  cc = ui->lineEdit\_2->text();  ui->lineEdit->clear();  ui->lineEdit\_2->clear();  if(book.findProduct(nn, cc))  {  QMessageBox **msgBox**(QMessageBox::Information,  "In stock",  "Yes, in stock",  QMessageBox::Ok);  msgBox.*exec*();  }  else{  QMessageBox **msgBox**(QMessageBox::Information,  "Not available",  "No, not available",  QMessageBox::Ok);  msgBox.*exec*();  }  } |

*Periodform.h*

|  |
| --- |
| #ifndef PERIODFORM\_H  #define **PERIODFORM\_H**  #include <QDialog>  namespace **Ui** {  class **PeriodForm**;  }  class **PeriodForm** : public QDialog  {  Q\_OBJECT  public:  explicit **PeriodForm**(QWidget \***parent** = nullptr);  ~**PeriodForm**();  private slots:  void **on\_pushButton\_2\_clicked**();  void **on\_pushButton\_clicked**();  private:  Ui::PeriodForm \***ui**;  };  #endif // PERIODFORM\_H |

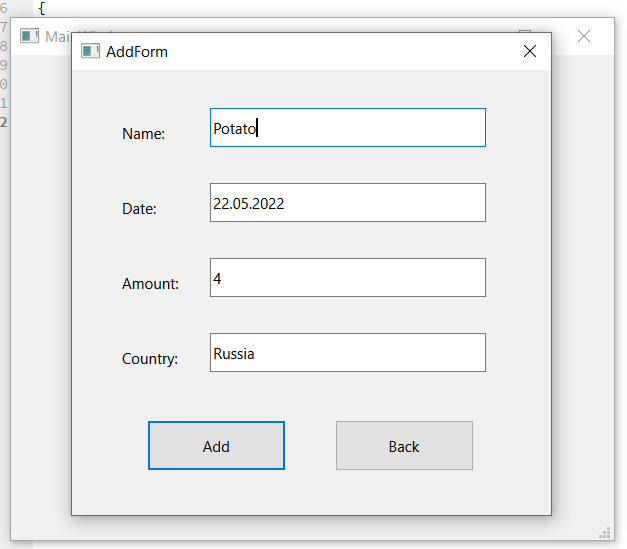
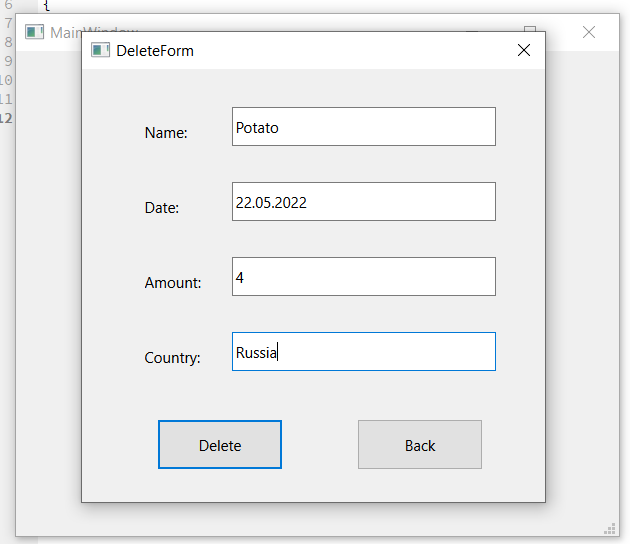
*Periodform.cpp*

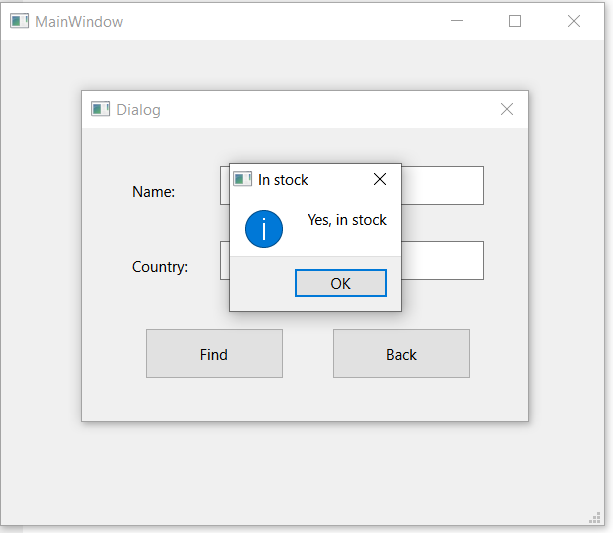
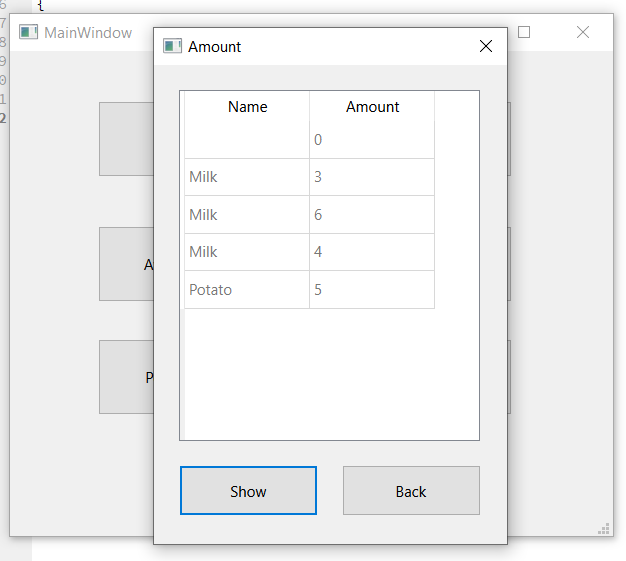
|  |
| --- |
| #include "periodform.h"  #include "ui\_periodform.h"  #include <QString>  #include <QDate>  #include "mainwindow.h"  #include "bookfile.h"  #include <bits/stdc++.h>  #include <QTabWidget>  PeriodForm::**PeriodForm**(QWidget \***parent**) :  QDialog(*parent*),  ui(new Ui::PeriodForm)  {  ui->setupUi(this);  }  PeriodForm::~**PeriodForm**()  {  delete ui;  }  void PeriodForm::**on\_pushButton\_2\_clicked**()  {  this->close();  }  void PeriodForm::**on\_pushButton\_clicked**()  {  struct **productType**  {  QString **name**, **country**;  int **amount**;  QDate **date**;  };  QFile \***f**;  productType **products**[100];  int **pNumber**;  productType **product**;  f = new QFile("f.txt");  f->*open*(QFile::ReadWrite);  pNumber = 0;  int **i** = 1;  f->*seek*(0);  QDataStream **in**(*f*);  while(!in.atEnd())  {  in>>product.name>>product.date>>product.amount>>product.country;  products[i].name = product.name;  products[i].date = product.date;  products[i].amount = product.amount;  products[i].country = product.country;  i++;  }  pNumber = i - 1;  int **lnum** = 0;  for(int **i** = 1; i <= pNumber; ++i)  {  if((products[i].date > QDate::fromString(ui->lineEdit->text(), "dd.MM.yyyy")) && (products[i].date < QDate::fromString(ui->lineEdit\_2->text(), "dd.MM.yyyy")))  {  QTableWidgetItem \***item**; // элемент таблицы  item = new QTableWidgetItem(); // создаем элемент  item->setFlags(Qt::NoItemFlags);//запрещаем  // выделение  item->setText(products[i].date.toString("dd.MM.yyyy")); // записываем текст  ui->tableWidget->setItem(lnum,0,*item*);  // к таблице  item = new QTableWidgetItem();// создаем элемент  item->setFlags(Qt::NoItemFlags); //запрещаем  // выделение  item->setText(products[i].name);  ui->tableWidget->setItem(lnum,1,*item*);  item = new QTableWidgetItem();// создаем элемент  item->setFlags(Qt::NoItemFlags); //запрещаем  // выделение  item->setText(products[i].country);  ui->tableWidget->setItem(lnum,2,*item*);  lnum++;  }  }  } |

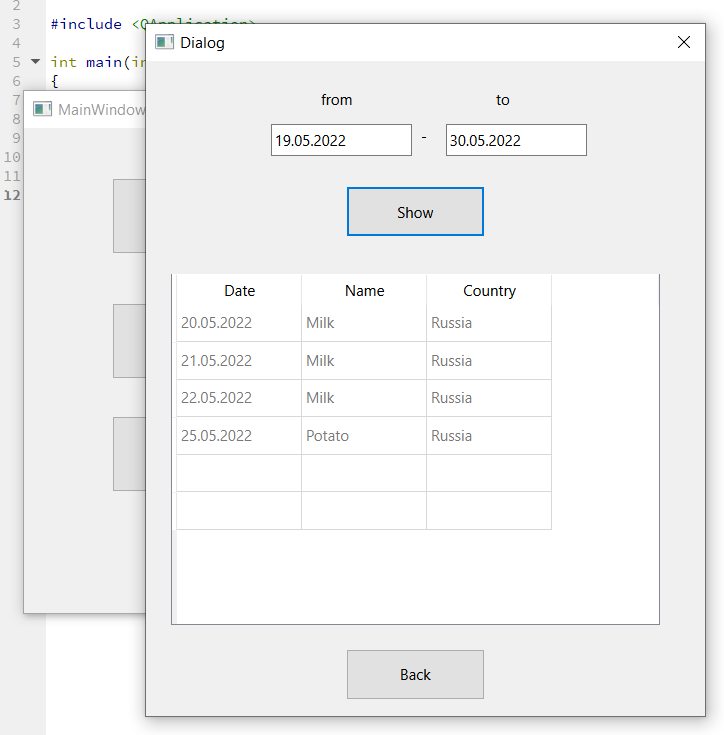
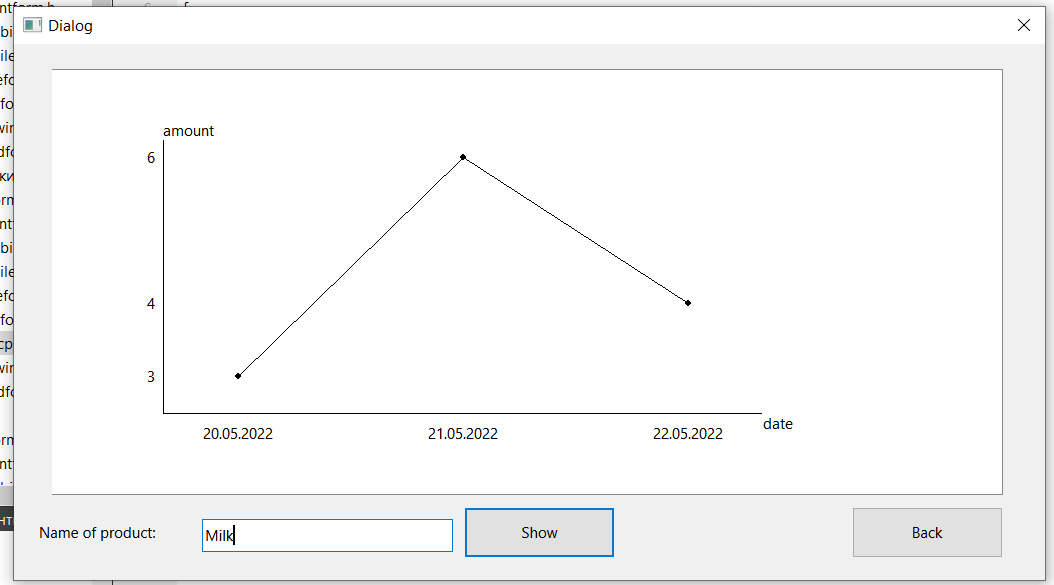
*Main.cpp*

|  |
| --- |
| #include "mainwindow.h"  #include <QApplication>  int main(int **argc**, char \***argv**[])  {  QApplication **a**(*argc*, *argv*);  MainWindow **w**;  w.show();  return a.exec();  } |

**Результаты тестирования:**

** **

** **

** **

**Вывод:** Получил навыки создания программной системы с оконным интерфейсом (проектирование, отладка и тестирование). Разработал программу, работающую в паре с базой данных (файлом), содержащем сведения о товарах. Программа в интерактивном режиме формирует файл, добавляет и удаляет данные, а также воспринимает некоторый ряд запросов и выдает на них ответы.