

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет Энергетический факультет

Кафедра Информатики, вычислительной техники и прикладной математики

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

(наименование дисциплины)

на тему «Создание базы данных банка»

Выполнил ст. гр. ИВТ 19-1
(группа)

Степанов В.Е.
(фамилия, инициалы)

Проверил: доцент кафедры
ИВТ и ПМ Соловьёв В.А.
(должность, ученая степень,
звание, фамилия, инициалы)

Чита
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет Энергетический факультет

Кафедра Информатики, вычислительной техники и прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Программирование»

(наименование направления подготовки)

на тему «Создание базы данных банка»

Выполнил студент группы ИВТ-19-1 Степанов Валерий Евгеньевич

(группа, фамилия, имя, отчество)

Руководитель работы: доцент кафедры ИВТ и ПМ Соловьёв В.А.

(должность, ученая степень, звание, фамилия, имя, отчество)

Календарный план

№	Наименование раздела курсовой работы	Неделя					
		1	2	3	4	5	6
1	Введение	25.04.20					
2	Глава 1		30.04.20				
3	Глава 2			11.05.20			
4	Глава 3				16.05.20		
5	Заключение					28.05.20	
6	Литература и приложение						01.06.20

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ГЛАВА 1. ТИПЫ ДАННЫХ И ОПЕРАЦИИ, РЕАЛИЗУЕМЫЕ В КР.....	4
1.1 Типы данных, используемые в КР.	4
1.2 Операции, реализуемые в КР.....	5
ГЛАВА 2.ИНТЕРФЕЙС ПРИЛОЖЕНИЯ.....	9
2.1 Описание компонентов, используемых в КР.	9
2.2 Реализация обработчиков событий	11
ГЛАВА 3. ТЕСТИРОВАНИЕ СОЗДАННОГО ПРИЛОЖЕНИЯ, ПРОВЕРКА	24
ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ.....	24
ЛИТЕРАТУРА.....	35
ПРИЛОЖЕНИЕ	37

ВВЕДЕНИЕ

Будет создано приложение – база данных, содержащая сведения о клиентах банка. База данных это информационная модель, позволяющая хранить данные о группе объектов, обладающих одинаковым набором свойств. В данной работе база данных имеет следующие сведения: код клиента, лицевой счёт, ФИО клиента, величина вклада, даты и суммы вложений или изыманий в течении года. Для реализации такого приложения используются простые и составные типы данных: текстовые и типизированные файлы, записи и строки. Так же в программе используются процедуры, функции, списки. В программе присутствует работа с несколькими модулями: модуль главной формы, модуль формы заполнения, модуль заполнения дополнительных данных и модуль основных процедур. Для создания приложения будет использована среда программирования Delphi 10 на основе языка Object Pascal.

Глава 1. Типы данных и операции, реализуемые в КР.

1.1 Типы данных, используемые в КР.

В данной работе используется список записей. Запись-структурированный, комбинированный тип данных, состоящий из фиксированного числа компонент (полей) различного типа [2].

Поэтому необходим тип записи, который будет содержать поля для хранения данных, такие как код клиента, лицевой счёт, ФИО клиента, величина вклада, даты и суммы вложений или изыманий в течении года. Для работы со списком необходима ещё одна запись, которая содержит следующие поля: указатель на следующий узел списка, указатель на предыдущий узел списка и запись, которую описали ранее.

Ещё один тип данных, который понадобится, это файловый тип. Он необходим для сохранения данных.

Секция type для данной работы:

type

Date = record

DT:TDateTime;//дата вложений или изыманий

SumVlojIzim:integer;//Сумма вложений и изыманий

end;

Client = record

{Фамилия клиента,Имя клиента,Отчество клиента}

Surname,Name,Patronymic:string[20];

Number:string[5];//Код клиента

PersonalAccount:string[16]; // лицевой счёт

Contribution:integer; // Величина вклада

Percent:1..100;// Проценты по вкладу

Actions : array [1..100] of Date;// действия клиента

```

end;

PUzel = ^Zp;
Zp=record
  x:Client;//информация хранящаяся в узлах списка
  next:PUzel;//указатель на следующий узел
  pred:PUzel;//указатель на предыдущий узел
end;
FZap = file of Client; //Файловый тип для хранения данных

```

1.2 Операции, реализуемые в КР.

В приложение реализованы следующие операции: сохранение списка в текстовый и типизированный файлы, добавление записи вручную, загрузка записи из типизированного файла, вывод отчёта на экран и в текстовый файл, удаление всего списка и удаление определённого элемента списка, поиск по коду клиента и по лицевому счёту, добавление дополнительных данных в список.

Интерфейсная часть созданного модуля состоит из следующих процедур:

```

procedure AddFirst(var f: PUzel; a: PUzel);//процедура добавления первого узла
procedure DelElemSp(var f,u: PUzel);//процедура удаления узла
procedure AddAfter(var old:PUzel; a: PUzel);//процедура добавления узла в
конец
Procedure AddLast(var old:PUzel; a: PUzel);//процедура добавления узла
Procedure ZapSpisok(var f:PUzel);// процедура для заполнения списка и вывода
данных в TStringGrid
procedure DelSpisok(var f: PUzel);//процедура удаления всего списка
procedure DelElement(var old,u: PUzel);//процедура удаления отдельного узла
procedure DelSp(var f: PUzel; var n:integer); //процедура удаления отдельного

```


элемента списка

procedure WriteSpText(var f: PUzel; var ftxt:Text);//процедура создания отчёта в текстовом файле

procedure WriteSpTip(var f: PUzel; var ftip:Fzap); // процедура записи списка в типизированный файл

procedure BuildTip (var ftip:Fzap); // процедура добавления записей в список из типизированного файла

procedure SearchNumberProc (var a:PUzel; var s:string);//процедура поиска по коду клиента

procedure SearchPersonalAccount(var a:PUzel; var s:string);//поиск по личному счёту

Procedure AddData(var u:PUzel; var s:string); //добавление операций

procedure AddDataa(var u:PUzel; var s:string);//процедура добавления дополнительных данных в список

Имеется база данных, состоящая из трёх записей.

- Сохранение записей в типизированный файл.

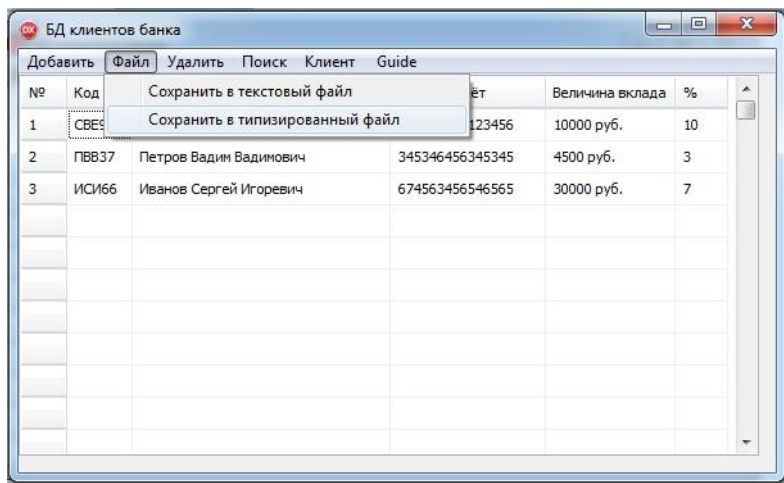


Рисунок 1.1.1 — Процедура сохранения в типизированный файл

Далее указываем файл и нажимаем кнопку «Открыть».

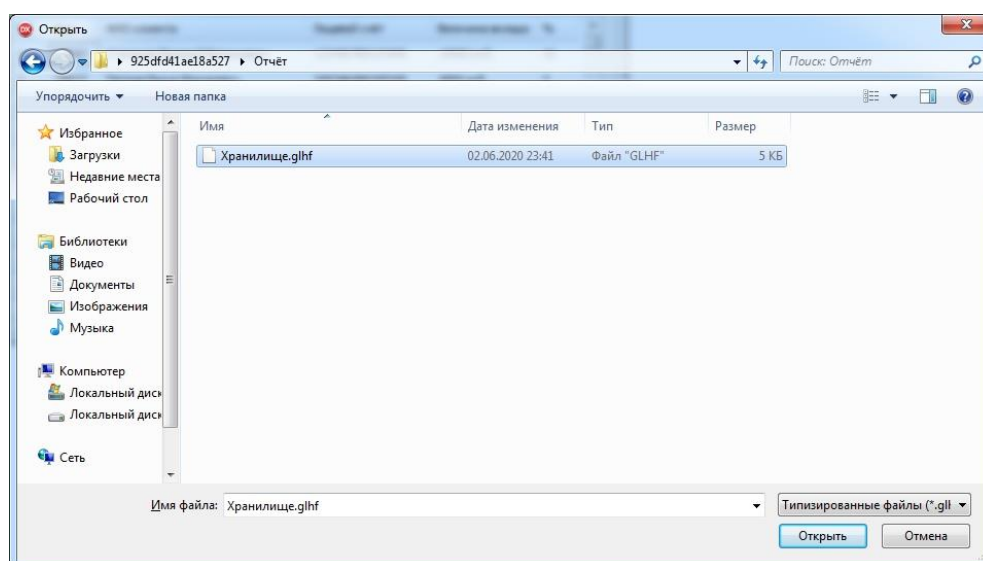


Рисунок 1.1.2 — Процедура сохранения в типизированный файл

Получаем подтверждение о сохранении данных.



Рисунок 1.1.3 — Процедура сохранения в типизированный файл

- Удаление всего списка.

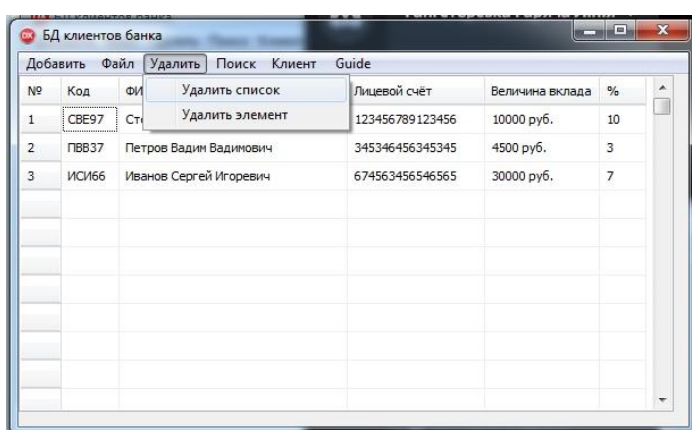


Рисунок 1.2.1 — Процедура удаления списка

После нажатия на кнопку меню “Удалить список”, данные списка будут стёрты и будет показано подтверждение о выполнении операции.

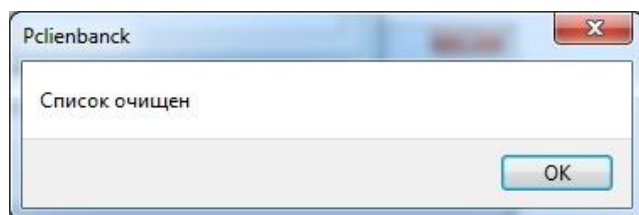


Рисунок 1.2.2 — Процедура удаления списка

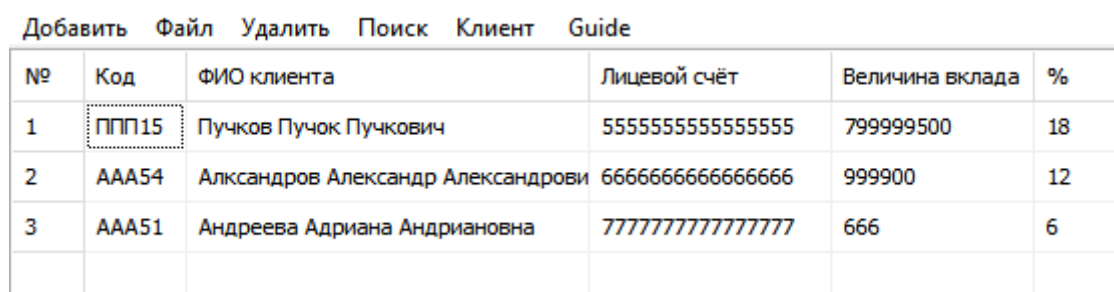
Глава 2.Интерфейс приложения.

Интерфейс должен быть простым и эффективным, так как является средством взаимодействия пользователя с программой. Интерфейс программы для курсовой работы сделан именно так, чтобы пользователь смог выполнить все необходимые для него операции.

2.1 Описание компонентов, используемых в КР.

Список компонентов, которые использовались в данной работе: StringGrid, MainMenu, Edit, MaskEdit, Label, SaveDialog, OpenFileDialog, Button.

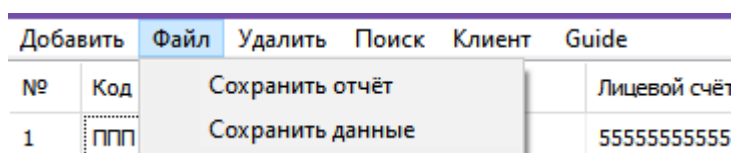
StringGrid - компонент для отображения различных данных в табличной форме. Ячейки компонента StringGrid Delphi могут содержать данные, имеющие тип String, а также отображать графику. В данной курсовой работе компонент StringGrid исполняет роль отображения списка, чтобы пользователь всегда мог видеть базу данных [3].



№	Код	ФИО клиента	Лицевой счёт	Величина вклада	%
1	ППП15	Пучков Пучок Пучкович	5555555555555555	799999500	18
2	AAA54	Алксандров Александр Александрови	6666666666666666	999900	12
3	AAA51	Андреева Адриана Андриановна	7777777777777777	666	6

Рисунок 2.1.1 – Описание компонентов. Компонент StringGrid.

MainMenu - это не визуальный компонент delphi, место размещения которого на форме не имеет значения для пользователя, так как он увидит не сам компонент, а меню, сгенерированное им. По внешнему виду оно представляет собой строку с пунктами меню. [3]



Добавить	Файл	Удалить	Поиск	Клиент	Guide
№	Код	Сохранить отчёт			Лицевой счёт
1	ППП	Сохранить данные			5555555555

Рисунок 2.1.2 – Описание компонентов. Компонент MainMenu.

Компонент Label предназначен для отображения статического текста, то есть надписей и меток на Форме, которые не меняются в течение всего времени работы программы. Конечно, текст надписи, отображаемый компонентом Label можно изменить, но не непосредственно, а только программно. [4]

Компонент Edit предназначен для ввода пользовательских данных и представляет собой однострочное поле. Основным свойством edit'а является text типа данных string. В данной программе компонент Edit находится на форме для заполнения данных. [3]

Фамилия:	<input type="text" value="Степанов"/>
Имя:	<input type="text" value="Валерий"/>
Отчество:	<input type="text" value="Евгеньевич"/>

Рисунок 2.1.3 – Описание компонентов. Компонент Edit.

Компонент OpenFileDialog не визуальный компонент предназначенный для поддержки операции открытия файлов способный работать с любыми типами файлов. При обращении к этому компоненту вызывается стандартное диалоговое окно открытия файла.[3]

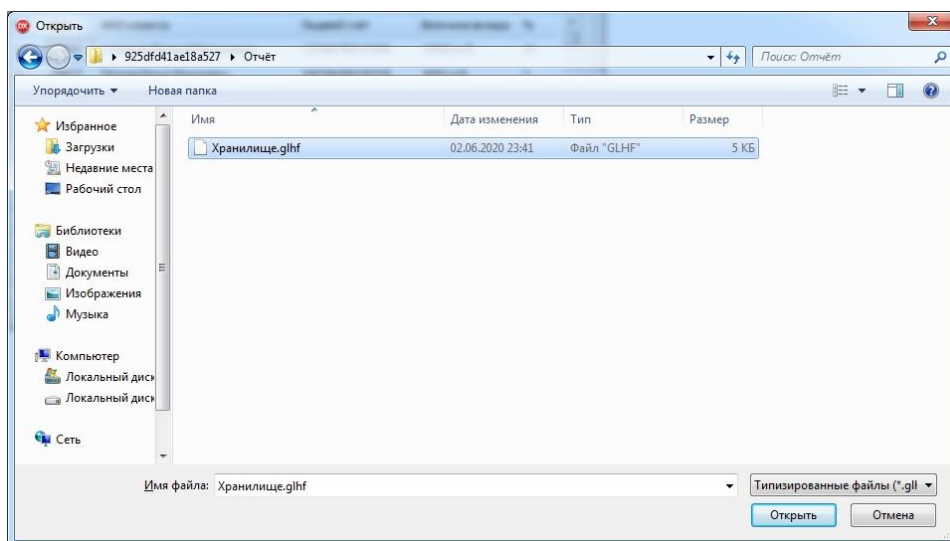


Рисунок 2.1.4 – Описание компонентов. Компонент OpenFileDialog.

Компонент Button это стандартная кнопка Delphi, кнопка имеет на поверхности надпись (описывающая её назначение при нажатии). Основное событие для кнопки является OnClick, выполняемое при нажатии, при этом кнопка меняет внешний вид, подтверждая этим происходящее действие визуально.[3]

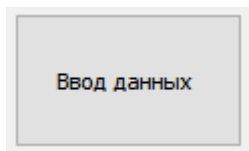


Рисунок 2.1.5 – Описание компонентов. Компонент Button.

2.2 Реализация обработчиков событий

Обработчики событий 1 формы:

Удаление узла списка.

```
procedure TOko.DeleteElClick(Sender: TObject);
var s : string;
    i,n:integer;
    a:PUzel;
    fl:boolean;
begin
    if flag = true then
        begin
            if not InputQuery('Укажите номер записи для удаления', s) then
                else
                    begin
                        fl:=false;
                        n:=1;
                        while TSG.Cells[0,n]<>" do
                            begin
                                if StrToInt(s)=n then
                                    begin
```

```

DelSp(PList,n);
fl:=true;
while TSG.Cells[0,n]<>" do
begin
    TSG.Cells[0,n]:=IntToStr(n);
    TSG.Cells[1,n]:=TSG.Cells[1,n+1];
    TSG.Cells[2,n]:=TSG.Cells[2,n+1];
    TSG.Cells[3,n]:=TSG.Cells[3,n+1];
    TSG.Cells[4,n]:=TSG.Cells[4,n+1];
    TSG.Cells[5,n]:=TSG.Cells[5,n+1];
    inc(n);
end;
TSG.Cells[0,n-1]:="";
if n>4 then
begin
    TSG.RowCount:=TSG.RowCount-1;
end;
if TSG.Cells[0,1]=" then
begin
    flag:=false;
end;
exit;
end;
inc(n);
end;
if fl=false then
begin
    ShowMessage('Записи с таким номером не существует');
end;
end;

```

```

        end
    else
        begin
            ShowMessage('Чтобы очистить список сначала нужно его заполнить');
        end;
    end;
end;

```

Задание параметров таблицы.

```

procedure TOkо.FormCreate(Sender: TObject);
begin
    oko.BorderStyle := bsSingle;
    TSG.ColWidths[0] := 35;
    TSG.ColWidths[1] := 50;
    TSG.ColWidths[2] := 200;
    TSG.ColWidths[3] := 120;
    TSG.ColWidths[4] := 100;
    TSG.ColWidths[5] := 40;
    TSG.Cells[0,0]:='№';
    TSG.Cells[1,0]:='Код';
    TSG.Cells[2,0]:='ФИО клиента';
    TSG.Cells[3,0]:='Лицевой счёт';
    TSG.Cells[4,0]:='Величина вклада';
    TSG.Cells[5,0]:='%';
end;

```

Открытие окна для добавления записи в список

```

procedure TOkо.N2Click(Sender: TObject);
begin
    Oko2.Show;
end;

```



```

        Поиск записи по лицевому счёту
procedure TОko.N5Click(Sender: TObject);
var s:string;
begin
    if flag = true then
        begin
            while true do
                begin
                    if InputQuery('Поиск', 'Введите лицевой счет клиента', s) then
                        begin
                            if s<>" then
                                begin
                                    SearchPersonalAccount(PList,s);
                                    break;
                                end
                            else
                                begin
                                    showmessage('Данные не были введены');
                                end;
                            end
                        else
                            begin
                                exit;
                            end;
                        end;
                    end
                else
                    begin
                        ShowMessage('Список пуст');
                    end;
                end;
            end;
        end;
    end;
end;

```

end;

Поиск записи по коду клиента

```
procedure TOkko.N7Click(Sender: TObject);//hgjkfld;ghfdk,lghfnmk,
var s:string;
begin
  if flag = true then
    begin
      while true do
        begin
          if InputQuery('Поиск', 'Введите код клиента для поиска', s) then
            begin
              if s<>" then
                begin
                  AddData(PList,s);
                  break;
                end
              else
                begin
                  showmessage('Данные не были введены');
                end;
              end
            end
          else
            begin
              exit;
            end;
          end;
        end
      end
    else
      begin
        end
      end
    end;
```

```
    ShowMessage('Список пуст');  
end;  
end;
```

Очистка таблицы и списка

```
procedure TOkko.DaleteSpClick(Sender: TObject);  
var i:integer;  
begin  
    if flag=true then  
        begin  
            i:=1;  
            DelSpisok(PList);  
            while TSG.Cells[1,i]<>" do  
                begin  
                    TSG.Cells[0,i]:= "";  
                    TSG.Cells[1,i]:= "";  
                    TSG.Cells[2,i]:= "";  
                    TSG.Cells[3,i]:= "";  
                    TSG.Cells[4,i]:= "";  
                    TSG.Cells[5,i]:= "";  
                    TSG.Cells[6,i]:= "";  
                    TSG.RowCount:=6;  
                    inc(i);  
                end;  
            flag:=false;  
            ShowMessage('Список очищен');  
        end  
    else  
        begin  
            ShowMessage('Чтобы очистить список сначала нужно его заполнить');
```

```

    end;
end;

    Открытие типизированного файла
procedure TOkko.OpenTypeFileClick(Sender: TObject);
var s:string;
begin
    OpenFileDialog1.Filter:= 'Типизированные файлы|*.glhf; *.glhf|Все файлы|*.*';
    if not OpenFileDialog1.Execute then exit;
    s := OpenFileDialog1.FileName;
    AssignFile(ftype,s);
    reset(ftype);
    if filesize(ftype)<>0 then
    begin
        BuildTip(ftype);
        closefile(ftype);
        flag:=true;
        showmessage('данные загружены');
    end
    else
    begin
        showmessage('Типизированный файл пуст');
    end;
end;

    end;

    Сохранение в текстовый файл.
procedure TOkko.SaveTxtFileClick(Sender: TObject);
var
    s: string;
begin
    if flag=true then
    begin

```

```

    OpenFileDialog1.Filter:= 'Текстовые файлы|*.txt; *.doc|Все файлы|*.*';
    if not OpenFileDialog1.Execute then exit;
    s := OpenFileDialog1.FileName;
    AssignFile(ftxt,s);
    Append(ftxt);
    WriteSpText(PList,ftxt);
    closefile(ftxt);
    ShowMessage('Список сохранён');
end
else
begin
    ShowMessage('Список не заполнен');
end;
end;

    Сохранение типизированного файла
procedure TOkko.SaveTypeFileClick(Sender: TObject);
var s:string;
begin
    if flag=true then
    begin
        OpenFileDialog1.Filter:= 'Типизированные файлы|*.glhf; *.glhf|Все файлы|*.*';
        if not OpenFileDialog1.Execute then exit;
        s := OpenFileDialog1.FileName;
        AssignFile(ftype,s);
        Rewrite(ftype);
        WriteSpTip(PList,ftype);
        Closefile(ftype);
        ShowMessage('Список сохранён');
    end
else

```

```

begin
    ShowMessage('Список не заполнен');
end;
end;

```

```

        Поиск клиента для ввода доп. данных
procedure TOkko.SearchNumberClick(Sender: TObject);
var s:string;
begin
    if flag = true then
        begin
            while true do
                begin
                    if InputQuery('Поиск', 'Введите код клиента для поиска', s) then
                        begin
                            if s<>" then
                                begin
                                    SearchNumberProc(PList,s);
                                    break;
                                end
                            else
                                begin
                                    showmessage('Данные не были введены');
                                end;
                            end
                        end
                    else
                        begin
                            exit;
                        end;
                    end;
                end;
            end;
        end;
end;

```

```

        end
    else
        begin
            ShowMessage('Список пуст');
        end;
    end;

    Обработчики событий 2 формы:

    Добавление записи
    procedure TOk2.Button1Click(Sender: TObject);
    begin
        if (Edit1.Text='') or (Edit2.Text='') or (Edit3.Text='') or
        (Edit4.Text='') or (Edit5.Text='') or (Edit6.Text='') then
            ShowMessage('Не все поля заполнены')
        else
            begin
                ZapSpisok(PList);
                Edit1.Clear;
                Edit2.Clear;
                Edit3.Clear;
                Edit4.Clear;
                Edit5.Clear;
                Edit6.Clear;
                Edit1.SetFocus;//перевод курсора на 1 поле
                Flag:=true;
            end;
        end;

        end;

    Заккрытие 2 формы
    procedure TOk2.Button2Click(Sender: TObject);

```

```
begin
Oko2.Close;
end;
```

Обработчики событий 3 формы:

Ввод данных

```
procedure TOko5.Button1Click(Sender: TObject);
```

```
var
```

```
k,i:integer;
```

```
s:string;
```

```
f:TdateTime;
```

```
begin
```

```
if Edit1.text="" then
```

```
    ShowMessage('Заполните поле суммы')
```

```
else
```

```
    if TryStrToDate(MaskEdit1.Text,f)=false then
```

```
        ShowMessage('Дата введена не верно')
```

```
    else
```

```
begin
```

```
s:=TSG3.Cells[0,1];
```

```
i:=1;
```

```
if (MaskEdit1.text="") or (Edit1.text="") then
```

```
begin
```

```
    ShowMessage('заполните все поля.')
```

```
end
```

```
else
```

```
while k<>1 do
```

```
begin
```



```

if TSG4.Cells[1,i]<>" then
begin
  if TSG4.RowCount<i then
    TSG4.RowCount:=TSG4.RowCount+1;
    inc(i)
  end
else
begin

  if TSG4.RowCount<i then
    TSG4.RowCount:=TSG4.RowCount+1;
    TSG4.Cells[0,i]:=MaskEdit1.Text;
    TSG4.Cells[1,i]:=Edit1.Text;
    AddDataa(PList,s);
    k:=1;
  end;
end;
end;
end;

```

Закрытие 3 формы

```

procedure TOko5.Button3Click(Sender: TObject);
begin
  Oko5.Close;
end;

```

```

procedure TOko5.FormCreate(Sender: TObject);
begin
  oko5.BorderStyle := bsSingle;
  TSG3.ColWidths[0] := 50;
  TSG3.ColWidths[1] := 200;

```

```
TSG3.ColWidths[2] := 120;  
TSG3.ColWidths[3] := 100;  
TSG3.ColWidths[4] := 40;  
TSG3.Cells[0,0]:='Код';  
TSG3.Cells[1,0]:='ФИО клиента';  
TSG3.Cells[2,0]:='Лицевой счёт';  
TSG3.Cells[3,0]:='Величина вклада';  
TSG3.Cells[4,0]:='%';  
TSG4.ColWidths[0] := 70;  
TSG4.ColWidths[1] := 70;  
TSG4.Cells[0,0]:='Дата';  
TSG4.Cells[1,0]:='Действия';  
end;
```

Глава 3. Тестирование созданного приложения, проверка полученных результатов.

Для начала работы необходимо запустить программу “PClienBanck.dproj”

__history	03.06.2020 19:29	Папка с файлами	
__recovery	03.06.2020 19:34	Папка с файлами	
Win32	29.05.2020 21:59	Папка с файлами	
Отчёт	02.06.2020 16:08	Папка с файлами	
Logic.pas	03.06.2020 18:20	Файл "PAS"	14 КБ
PClienBanck.~dsk	02.06.2020 21:47	Файл "~DSK"	14 КБ
PClienBanck.dpr	02.06.2020 11:37	Файл "DPR"	1 КБ
PClienBanck.dproj	03.06.2020 18:24	Delphi Project File	47 КБ

Рисунок 3.1 – Приложение «База данных банка»

Чтобы добавить запись в список, нужно открыть вкладку «Добавить» и выбираем пункт «Добавить запись».

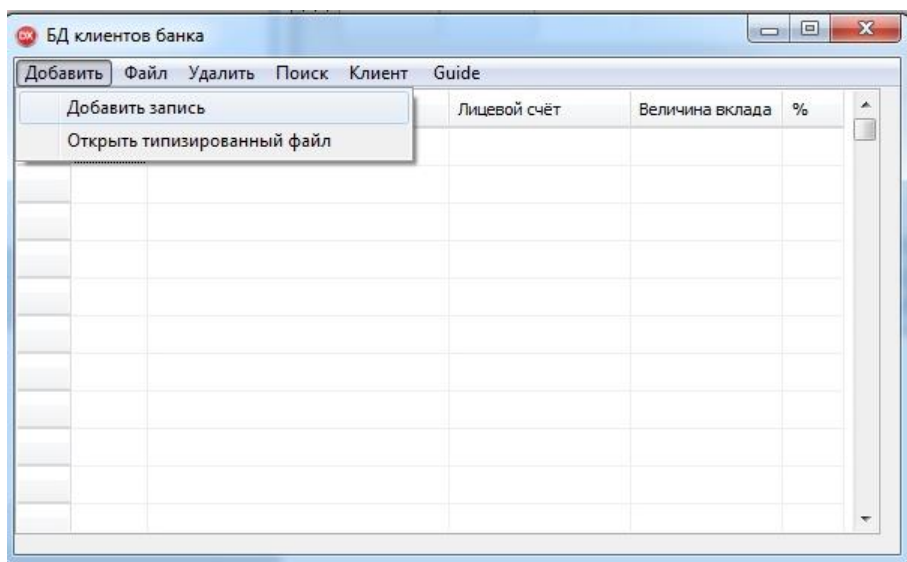


Рисунок 3.2.1 – Добавление записи

Далее откроется окно для заполнения, в нём необходимо ввести данные.

Рисунок 3.2.2 – Добавление записи

Нажмите на кнопку «Ввод данных», чтобы занести данные в программу. Они появятся в таблице.

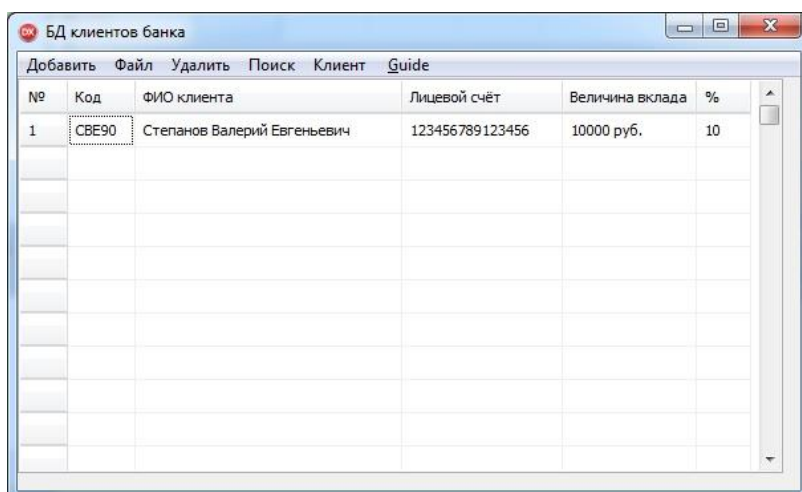


Рисунок 3.2.3 – Добавление записи

Чтобы загрузить данные из типизированного файла, нужно открыть вкладку «Добавить» и выбрать пункт «Открыть типизированный файл».

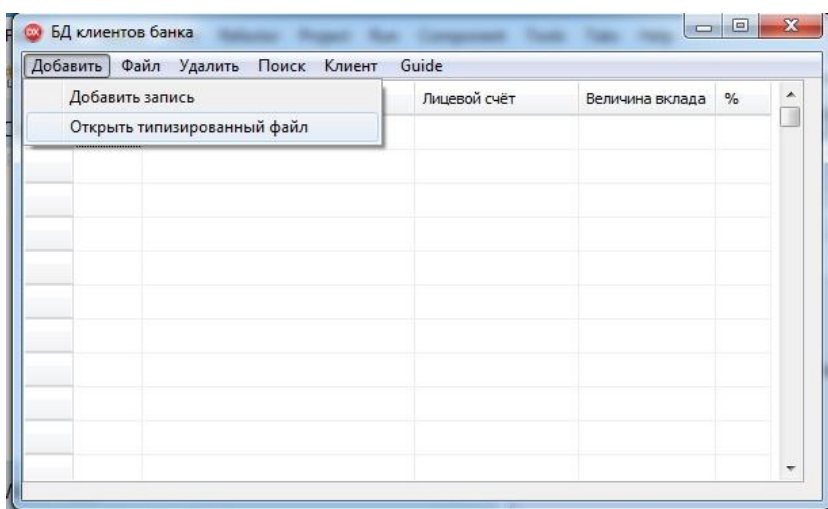


Рисунок 3.3.1 – Загрузка данных из тип. Файла

Далее откроется окно, для выбора файла. Указать файл и нажать кнопку «Открыть».

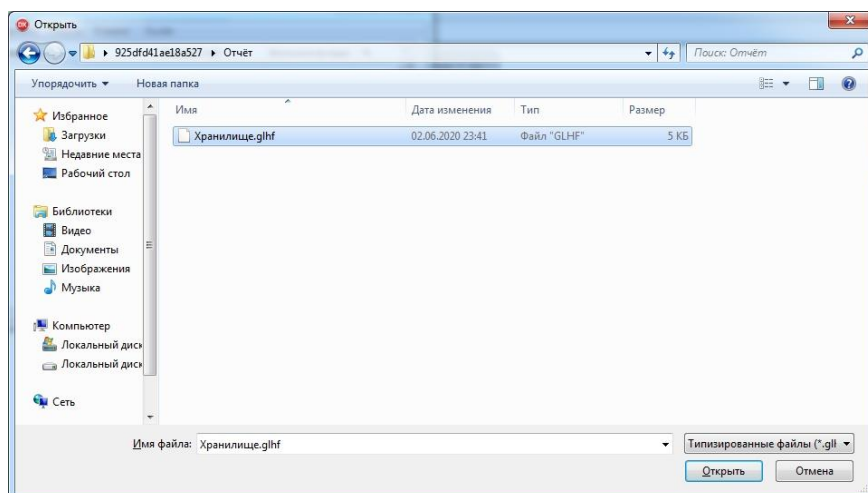


Рисунок 3.3.2 – Загрузка данных из тип. Файла

После открытия файла, данные отобразятся в таблице

№	Код	ФИО клиента	Лицевой счёт	Величина вклада	%
1	СВЕ97	Степанов Валерий Евгеньевич	123456789123456	10000 руб.	10
2	ПВВ37	Петров Вадим Вадимович	345346456345345	4500 руб.	3
3	ИСИ66	Иванов Сергей Игоревич	674563456546565	30000 руб.	7

Рисунок 3.3.3 – Загрузка данных из тип. Файла

Чтобы сохранить данные в текстовый файл, нужно открыть вкладку «Файл» и выбрать пункт «Сохранить отчёт».

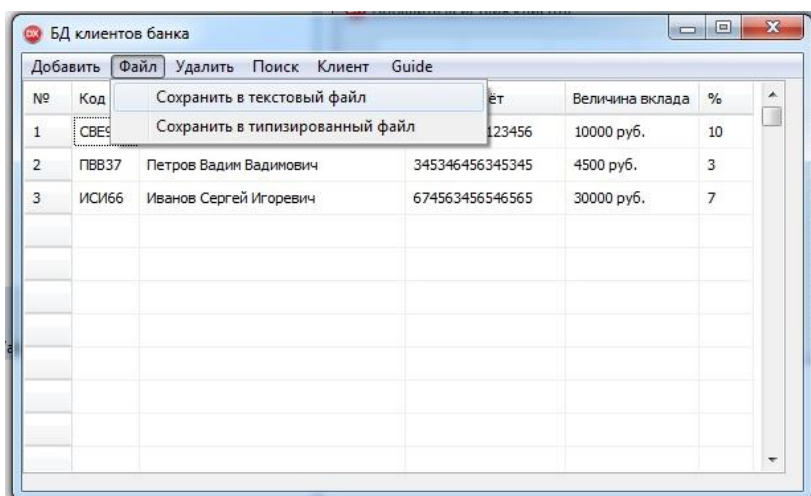


Рисунок 3.4.1 – Сохранение в текстовый файл

Далее откроется окно, для выбора файла. Указать файл и нажать кнопку «Сохранить».

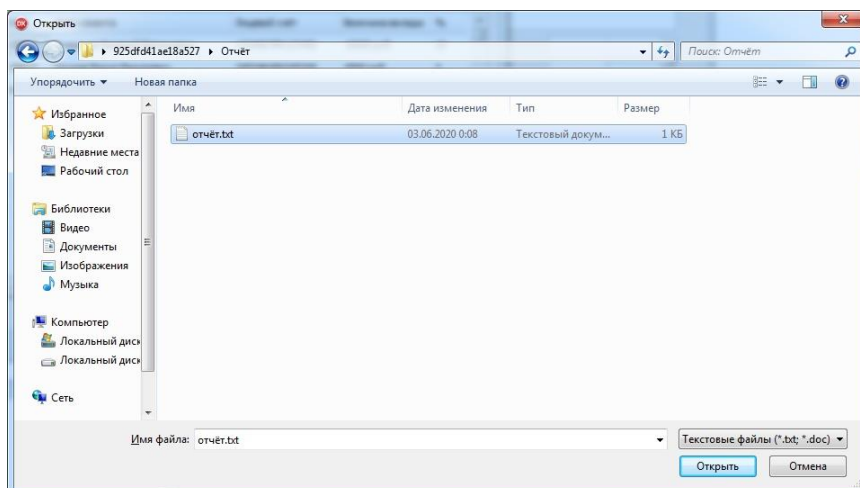


Рисунок 3.4.2 – Сохранение в текстовый файл

После сохранения выйдет окошко, сообщающее о завершении сохранения.

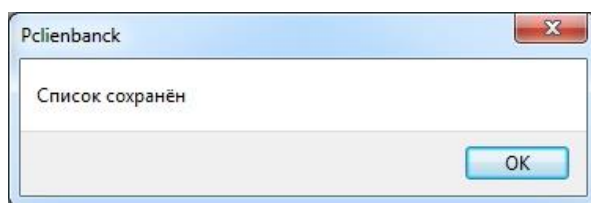


Рисунок 3.4.3 – Сохранение в текстовый файл

Для того чтобы сохранить данные в типизированный файл: Открыть вкладку «Файл» и выбрать пункт «Сохранить данные».

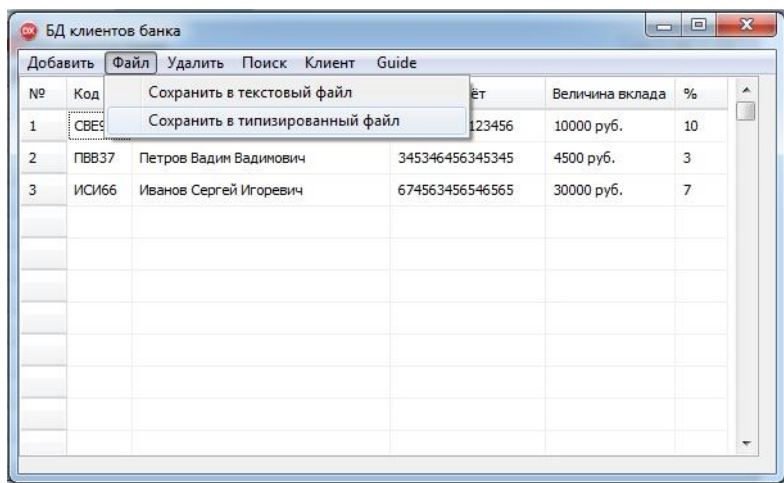


Рисунок 3.5.1 – Сохранение в типизированный файл

Далее откроется окно, для выбора файла. Указать файл и нажать кнопку «Открыть».

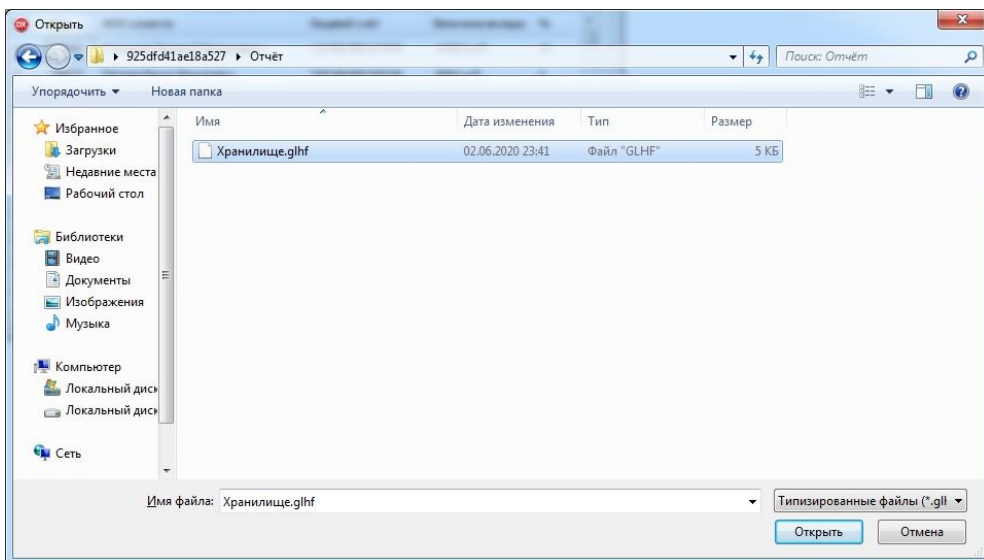


Рисунок 3.5.2 – Сохранение в типизированный файл

После появится окно подтверждающее, что файл записан.

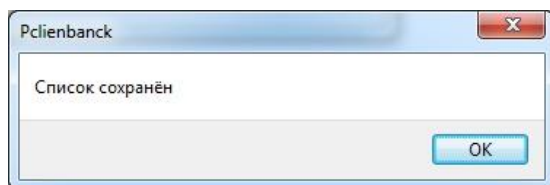


Рисунок 3.5.3 – Сохранение в типизированный файл

Чтобы удалить определённый элемент, нужно открыть вкладку «Удалить» и выбрать пункт «Удалить элемент».

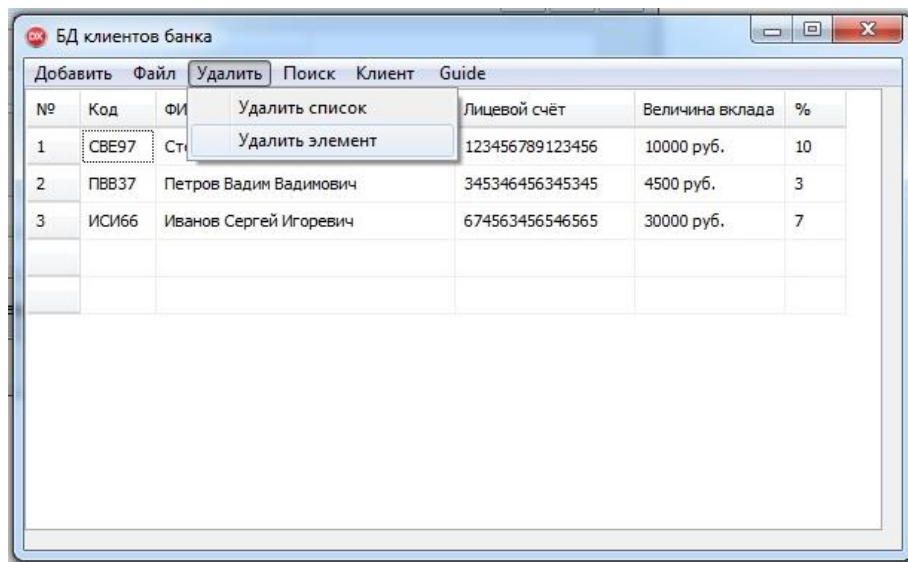


Рисунок 3.6.1 – Удаление определённой строки

Далее программа запросит порядковый номер записи

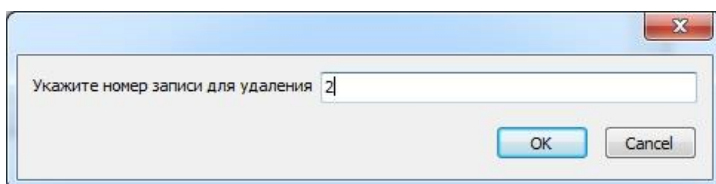


Рисунок 3.6.2 – Удаление определённой строки

После подтверждения программа удалит запись из списка

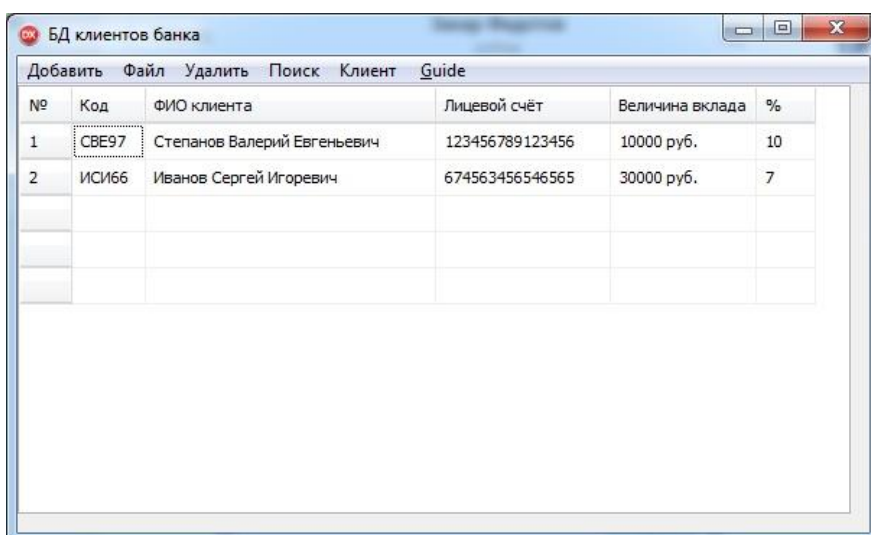


Рисунок 3.6.3 – Удаление определённой строки

Чтобы удалить весь список, нужно открыть вкладку «Удаление списка» и выбрать пункт «Удалить список». После этого список будет очищен.

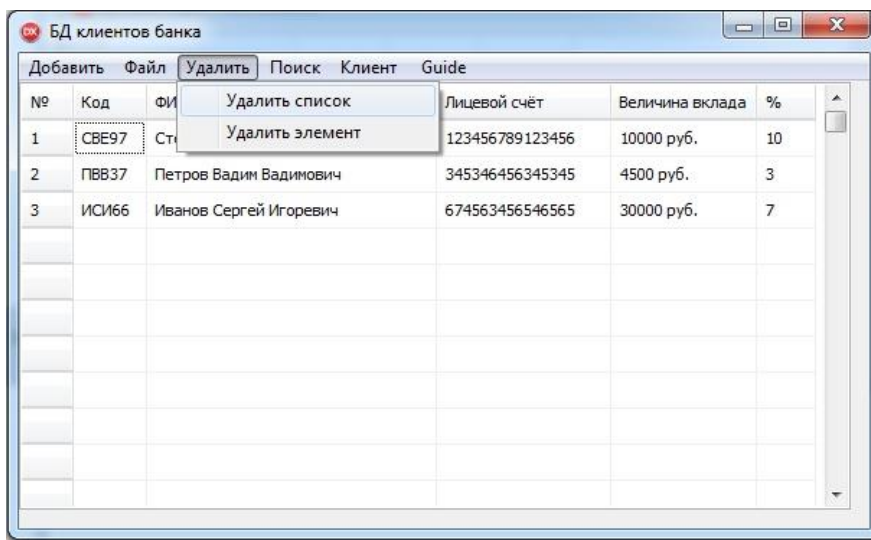


Рисунок 3.7.1 – Удалить весь список.

Появится окно с сообщением о выполненной операции.

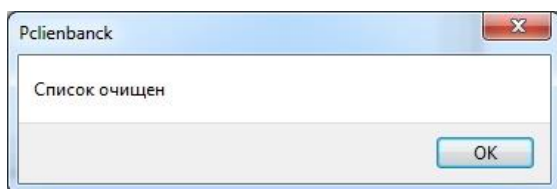


Рисунок 3.7.2 – Удалить весь список.

Что бы найти клиента по коду нужно открыть вкладку «Поиск» и выбрать пункт «поиск по коду».

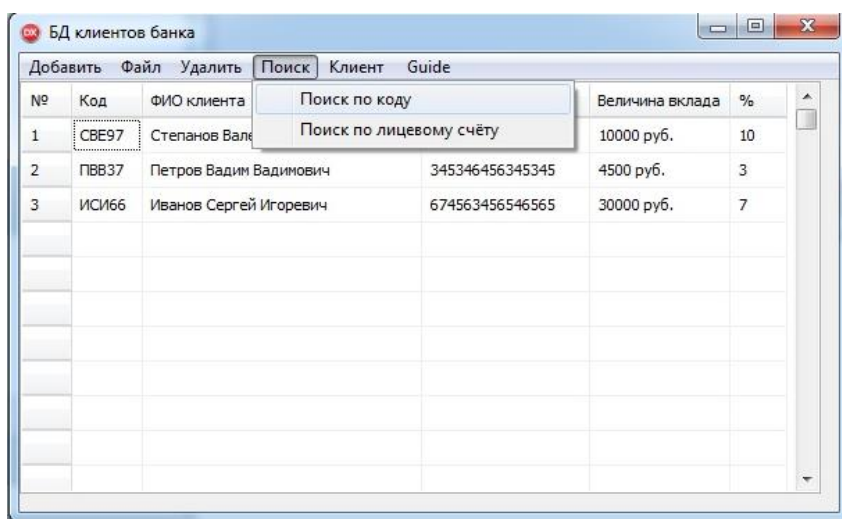


Рисунок 3.8.1 – Поиск по коду.

Далее откроется окно для ввода кода клиента. Вводим.

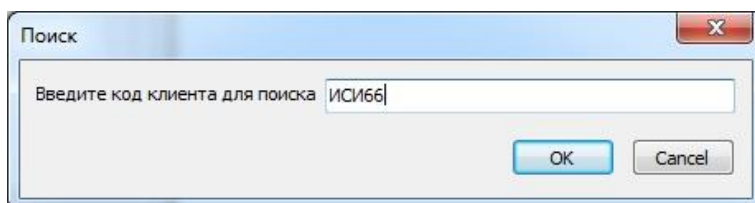


Рисунок 3.8.2 – Поиск по коду.

Открывается окно с краткой информацией о клиенте и предложением сохранить отчёт.

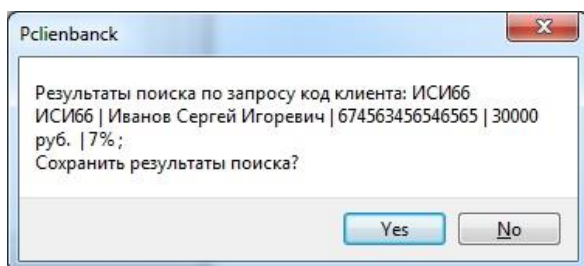


Рисунок 3.8.3 – Поиск по коду.

Далее откроется окно, для выбора файла. Указать файл и нажать кнопку «Открыть».

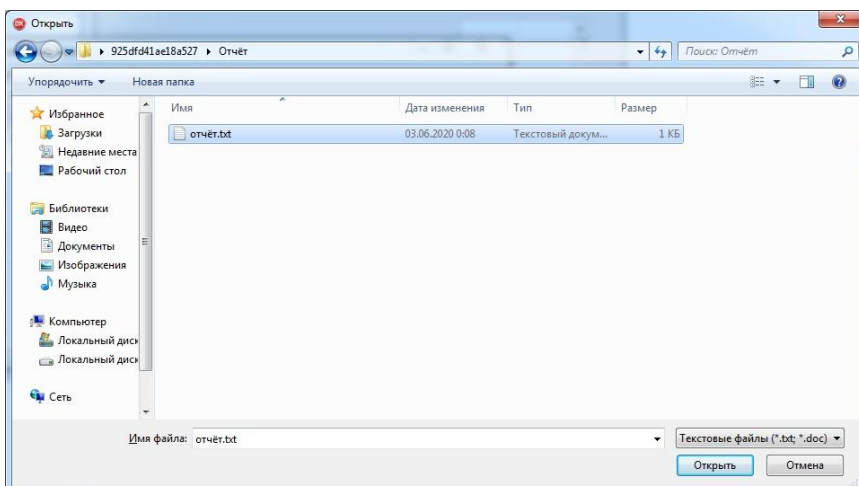


Рисунок 3.8.3 – Поиск по коду.

Появится окно с сообщением о выполненной операции.



Рисунок 3.8.5 – Поиск по коду.

Что бы найти клиента по коду нужно открыть вкладку «Поиск» и выбрать пункт «Поиск по лицевому счёту».

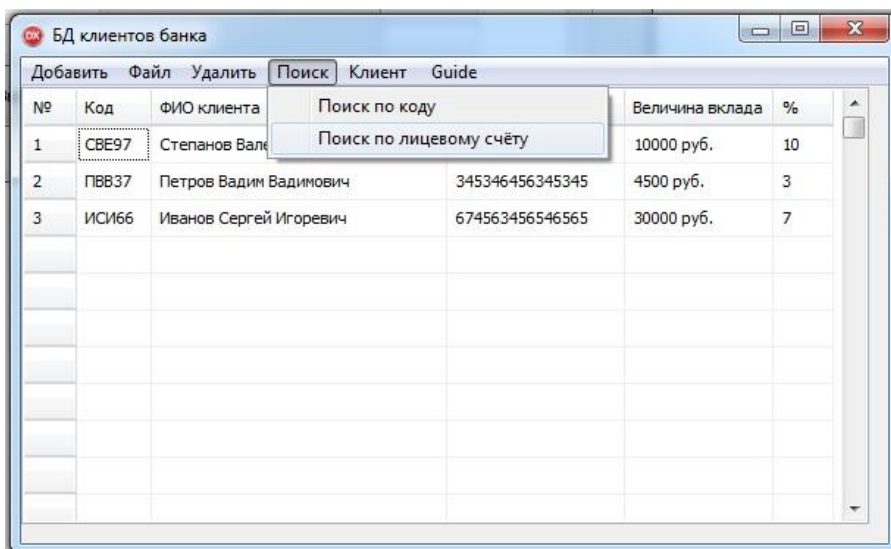


Рисунок 3.9.1 – Поиск по лицевому счёту.

Далее откроется окно для ввода кода клиента. Вводим.

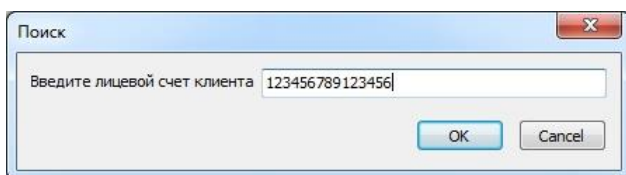


Рисунок 3.9.2 – Поиск по лицевому счёту.

Открывается окно с краткой информацией о клиенте и предложением сохранить отчёт.

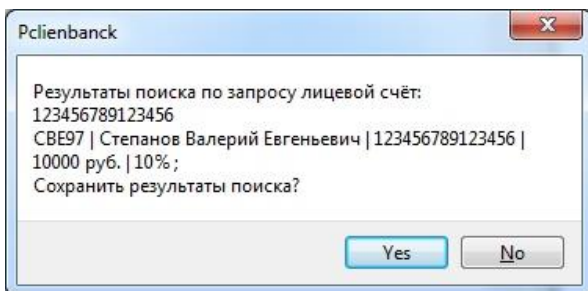


Рисунок 3.9.3 – Поиск по лицевому счёту.

Далее откроется окно, для выбора файла. Указать файл и нажать кнопку «Открыть».

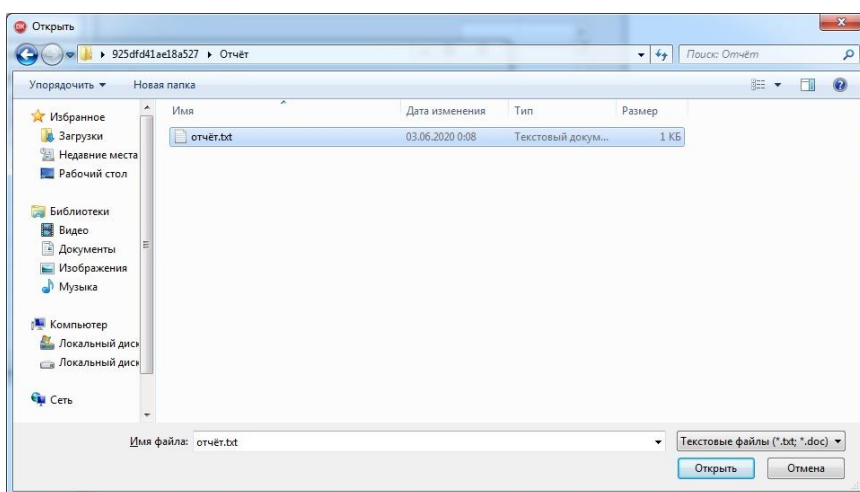


Рисунок 3.9.4 – Поиск по лицевому счёту.

Появится окно с сообщением о выполненной операции.

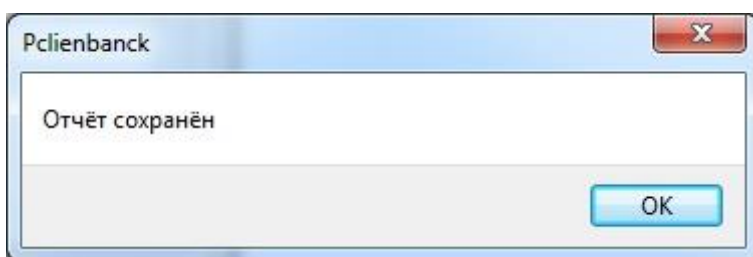


Рисунок 3.9.5 – Поиск по лицевому счёту.

Для добавления дополнительных данных нужно открыть вкладку «Клиент» выбрать «Действия».

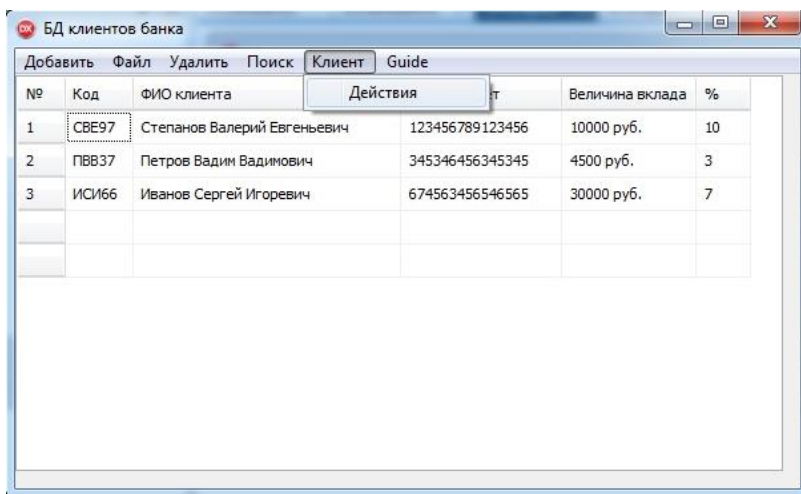


Рисунок 3.9.1 – Добавление операций.

Откроется окно для поиска клиента по коду, вводим код.

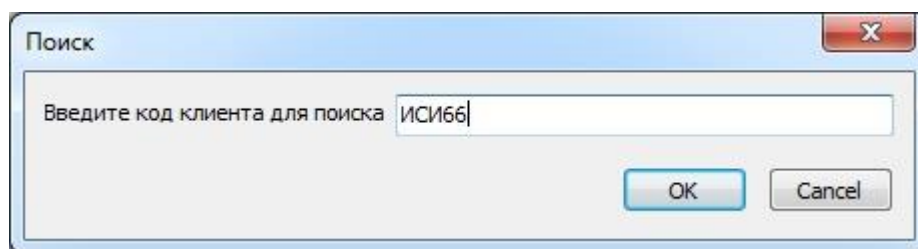


Рисунок 3.9.2 – Добавление операций.

Вводим дату и сумму

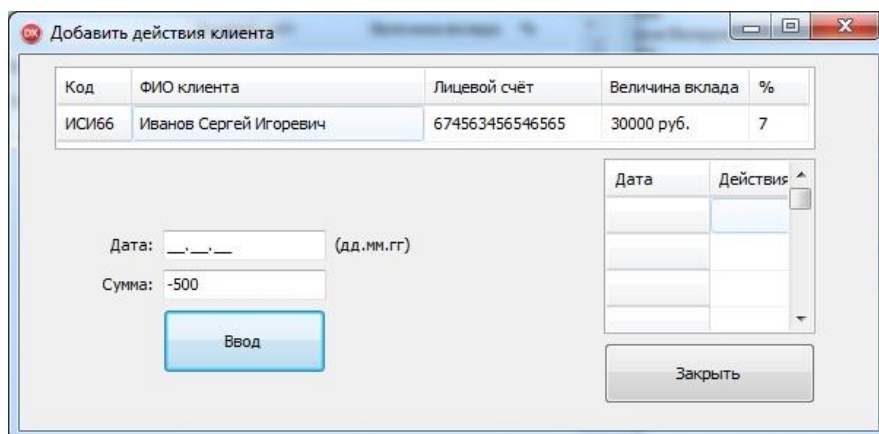


Рисунок 3.9.3 – Добавление операций.

После нажатия на кнопку «Ввод» данные заносятся в список и таблицу. При этом величина вклада изменяется на введенную вами сумму.

ЗАКЛЮЧЕНИЕ

В данном приложении была реализована база данных с помощью двусвязного списка. Интерфейс приложения разрабатывался на основе среды программирования Delphi 10. В приложении были реализованы следующие функции: создание записей, загрузка данных из типизированного файла, сохранение списка в текстовый и типизированные файлы, удаление всего списка и его отдельного элемента, поиск по коду и лицевому счёту клиента, добавление дополнительных данных в список. Также в приложении были добавлены две дополнительные формы: для добавления записей и добавление дополнительных данных в запись. Информация о клиентах выводилась в компонент StringGrid.

Приложение прошло тестирование и работало без сбоев и ошибок.

Было проанализировано 3 источника дополнительной литературы (учебные пособия, учебники и веб-сайты).

ЛИТЕРАТУРА

- 1) Wikipedia База данных [Электронный ресурс] / Режим доступа:
https://ru.wikipedia.org/wiki/База_данных (03.05.20)
- 2) Динамические структуры данных в Паскале [Электронный ресурс] / Режим доступа:
http://www.pascal.helpov.net/index/dynamic_lists_pascal_programming
(02.05.2020)
- 3) Компоненты Delphi [Электронный ресурс] / Режим доступа:
<https://www.h-delphi.ru> (02.05.2020)
- 4) Компоненты Delphi [Электронный ресурс] / Режим доступа:
<http://www.delphi-manual.ru> (27.04.2020)
- 5) Архангельский А.Я. Delphi 7. Справочное пособие: учебник – М.: Бином, 2004. – 1024 с. (04.05.20)
- 6) Оформление курсовой работы [Электронный ресурс] / Общие требования к построению и оформлению текстовой документации ЗабГУ. – Режим доступа:
http://zabgu.ru/files/html_document/pdf_files/fixed/Normativny'e_dokumenty'/MI_01-02-2018_Obshhie_trebovaniya_k_postroeniyu_i_oformleniyu_uchebnoj_tekstovoj_dokumentacii.pdf

ПРИЛОЖЕНИЕ

Интерфейсная часть главного модуля
interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Logic, Vcl.Menus, UF4ZapBD;

type

TOko = class(TForm)
 TSG: TStringGrid;
 MainMenu1: TMainMenu;
 N1: TMenuItem;
 N2: TMenuItem;
 SaveDialog1: TSaveDialog;
 N3: TMenuItem;
 SaveTxtFile: TMenuItem;
 OpenDialog1: TOpenDialog;
 OpenTypeFile: TMenuItem;
 SaveTypeFile: TMenuItem;
 N4: TMenuItem;
 DaleteSp: TMenuItem;
 DeleteEl: TMenuItem;
 Search: TMenuItem;
 SearchNumber: TMenuItem;
 N5: TMenuItem;
 N6: TMenuItem;
 N7: TMenuItem;
 Help1: TMenuItem;
 procedure FormCreate(Sender: TObject);
 procedure N2Click(Sender: TObject);
 procedure SaveTxtFileClick(Sender: TObject);
 procedure SaveTypeFileClick(Sender: TObject);
 procedure OpenTypeFileClick(Sender: TObject);


```

procedure DaleteSpClick(Sender: TObject);
procedure DeleteElClick(Sender: TObject);
procedure SearchNumberClick(Sender: TObject);
procedure N5Click(Sender: TObject);
procedure N7Click(Sender: TObject);
procedure Help1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Oko: TOko;
    PList: PUzel;
    ftxt: text; //файловая переменная для текстового файла
    flag:boolean; //переменная для проверки заполненности списка
    ftype: FZap; //файловая переменная для типизированного файла

```

Интерфейсная часть созданного модуля

```

interface
uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics,
    Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids;

type
    Date = record
        DT:TDateTime;
        SumVlojIzim:integer;//Сумма вложений и изыманий
    end;

    Client = record
        {Фамилия клиента,Имя клиента,Отчество клиента}
        Surname,Name,Patronymic:string[20];
        Number:string[5];//Код клиента
        PersonalAccount:string[16]; // лицевой счёт
        Contribution:integer; // Величина вклада
        Percent:1..100;// Проценты по вкладу
        Actions : array [1..100] of Date;// действия клиента
    end;

```

```

    end;

    PUzel = ^Zp;
Zp=record
    x:Client;
    next:PUzel;
    pred:PUzel;
end;

FZap = file of Client;  //Файловый тип для хранения данных

procedure AddFirst(var f: PUzel; a: PUzel);//процедура добавления первого
узла
procedure DelElemSp(var f,u: PUzel);//процедура удаления узла
procedure AddAfter(var old:PUzel; a: PUzel);//процедура добавления узла в
конец
Procedure AddLast(var old:PUzel; a: PUzel);//процедура добавления узла
Procedure ZapSpisok(var f:PUzel);// процедура для заполнения списка и
вывода данных в TSG
procedure DelSpisok(var f: PUzel);//процедура удаления всего списка
procedure DelElement(var old,u: PUzel);//процедура удаления отдельного узла
procedure DelSp(var f: PUzel; var n:integer); //процедура удаления отдельного
элемента списка
procedure WriteSpText(var f: PUzel; var ftxt:Text);
procedure WriteSpTip(var f: PUzel; var ftip:Fzap); // процедура записи списка
в типизированный файл
procedure BuildTip (var ftip:Fzap); // процедура добавления записей в список
из типизированного файла
procedure SearchNumberProc (var a:PUzel; var s:string);//процедура поиска по
коду клиента
procedure SearchPersonalAccount(var a:PUzel; var s:string);//поиск по
лицевому счёту
Procedure AddData(var u:PUzel; var s:string); //добавление операций
procedure AddDataa(var u:PUzel; var s:string);//процедура добавления доп
данных в список

```

Интерфейсная часть второй формы

```

interface

```

```

uses

```

```

    Winapi.Windows,    Winapi.Messages,    System.SysUtils,    System.Variants,

```

System.Classes, Vcl.Graphics,
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.Grids,logic;

type

```
TOko5 = class(TForm)
    Button1: TButton;
    Button3: TButton;
    TSG3: TStringGrid;
    TSG4: TStringGrid;
    MaskEdit1: TMaskEdit;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;
```

var

```
Oko5: TOko5;
```

Интерфейсная часть третьей формы

interface

uses

```
Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids, Logic,
UClientBanck;
```

type

```
TOko2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label5: TLabel;
    Label4: TLabel;
    Edit6: TEdit;
    Label6: TLabel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
private
    { Private declarations }
public
    { Public declarations }
end;
```

```
var
    Oko2: TOko2;
```