МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет Энергетический

Кафедра Информатики и вычислительной техники, прикладной математики

**КУРСОВАЯ РАБОТА**

по программированию

(наименование дисциплины)

на тему «База данных клиентов банка»

Выполнил ст. гр. ИВТ-19-1

(группа)

Артемьев А.В.

(фамилия, инициалы)

Проверил доцент кафедры ИВТ и ПМ

Соловьёв В. А.

(должность, учёная 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

Кафедра информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ

на курсовую работу

По дисциплине: Программирование

Студенту: Артемьеву Артему Владимировичу

Специальности (направления подготовки): 09.03.01 Информатика и вычислительная техника

Тема курсовой работы: База данных клиентов банка;

Срок подачи студентом законченной работы: 26.06.2020

Исходные данные к работе: информация о вкладах клиентов банка.

Перечень подлежащих разработке в курсовой работе вопросов:

Создать программу обслуживания клиентов банка, в которой создаётся база данных, содержащая код клиента, лицевой счёт (величина вклада, проценты по вкладу, даты и суммы вложений в течение года).

Программа должна предоставлять возможность просматривать, добавлять, удалять, копировать и хранить данные. В программе использовать модули и объекты.

Дата выдачи задания: 09.02.2020

Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Соловьёв В. А./

(подпись, расшифровка подписи)

Задание принял к исполнению: 29.05.2020 г.

Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_ / Артемьев А.В.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет Энергетический

Кафедра Информатики и вычислительной техники, прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

По дисциплине: Программирование

На тему «База данных клиентов банка»

Цели курсовой работы:

1. применить простые и составные типы данных, изученных в течение двух семестров;
2. использовать процедуры и функции, модули, файлы (текстовые и типизированные), списки;
3. изучить и использовать динамические переменные;
4. обойтись без использования классов в модуле логики программы;
5. изучить работу с необходимыми компонентами Delphi;
6. создать простой оконный интерфейс на Delphi.

Выполнил: студент группы ИВТ-19-2 Артемьев Артем Владимирович

Руководитель работы: Соловьев Владимир Александрович

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

КАЛЕНДАРНЫЙ ГРАФИК  
выполнения курсовой работы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Этапы выполнения курсовой работы | Месяцы и недели | | | | | | | | | | | | | | | |
| Март | | | | Апрель | | | | Май | | | | Июнь | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. Патентный поиск |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. Анализ предметной области |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. Проектирование |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. Реализация |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. Защита работы |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

План выполнен: руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, расшифровка подписи)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc41816283)

[Глава 1. Типы данных и операции, реализуемые в КР 4](#_Toc41816284)

[1.1 Типы данных 4](#_Toc41816285)

[1.2 Операции 6](#_Toc41816286)

[Глава 2. Интерфейс приложения 16](#_Toc41816287)

[2.1 Описание компонентов 16](#_Toc41816288)

[2.2 Реализация обработчиков событий 18](#_Toc41816289)

[Глава 3. Тестирование созданного приложения, проверка полученных результатов 25](#_Toc41816290)

[ЗАКЛЮЧЕНИЕ 35](#_Toc41816291)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc41816292)

[ПРИЛОЖЕНИЕ А 39](#_Toc41816293)

# ВВЕДЕНИЕ

Тема является актуальной, так как значительно повысился уровень потребительских качеств систем управления базами данных: возможность сводить воедино информацию из самых разных источников, удобный для пользователя интерфейс, что особенно важно сейчас – для работы в сети.

Главная задача данной курсовой работы – создать простейшую базу данных (состоящую из одной таблицы) в виде приложения, которое создаётся с помощью интегрированной среды разработки ПО (программного обеспечения) на языке Delphi (ранее Object Pascal).

База данных (общее понятие) – это представленная в объективной форме совокупность самостоятельных материалов систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ) [1].

База данных (в информатике) – это совокупность массивов и файлов данных, организованная по определённым правилам, предусматривающим стандартные принципы описания, хранения и обработки данных независимо от их вида. [2]

Для выполнения данной курсовой работы были поставлены цели:

- применить простые и составные типы данных, изученных в течение двух семестров; использовать процедуры и функции, модули, файлы (текстовые и типизированные), списки;

- изучить и использовать динамические переменные; обойтись без использования классов в модуле логики программы; изучить работу с необходимыми компонентами и создать простой оконный интерфейс на Delphi.

Исходя из целей, был выдвинут ряд задач: 1) осуществить поиск информации; 2) изучить материал; 3) создать и протестировать базу данных.

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

## Типы данных

Для начала определимся с типами данных, которые будут использоваться в секции typeсоздаваемого логического модуля. Всего нужно использовать четыре главных элемента: запись с данными о клиенте, указатель на узел списка, узел списка и типизированный файл [3].

Типизированным называют файл, состоящий из однородных элементов, относящихся к одному и тому же типу данных. Каждый элемент такого файла имеет индекс, подобно элементам массива. В отличии от обычного массива, типизированный файл может длительное время сохраняться на жёстком диске компьютера или же на других электронных носителях. К каждому элементу такого файла возможен как прямой (по индексу элемента), так и последовательный доступ. Особенностью такого файла является то, что в отличии от текстового его нельзя просматривать с помощью обычного текстового редактора [16].

Теперь разберём структуру каждого элемента:

* + - 1. Запись с данными о клиенте (назовём client) содержит в себе семь элементов:
* Код клиента – int64 (ибо в integer не поместится, например, номер телефона);
* Величина вклада – double (т. к. ввод будет в рублях, а про копейки не стоит забывать);
* Проценты вклада – double (т. к. проценты могут быть не целыми)
* Дата пополнения вклада – string [10] (на описание полной даты хватает десяти символов);
* Сумма пополнения вклада – double;
* Дата изымания вклада – string [10];
* Сумма изымания вклада – double.
  + - 1. Типизированный указатель на узел списка(назовём PUzel) – тип узла списка.
      2. Узел списка (назовём ZpTwo) представляет собой запись из 3 элементов:
* Сведения о клиенте – тип client;
* Указатель на следующий узел – тип PUzel;
* Указатель на предыдущий узел -тип PUzel;

(т. е. список является двусвязным).

* + - 1. Типизированный файл (назовём FZap) – файловый тип из элементов типа client.

## Операции

Теперь приведём необходимый список основных операций, содержащейся в интерфейсной части создаваемого модуля:

1. Создание списка вручную
2. Создание списка из типизированного файла
3. Добавление нескольких элементов списка
4. Удаление конкретного узла
5. Полное удаление списка
6. Сохранение списка в файлы 2-х типов
7. Сортировка по коду клиента
8. Поиск записи в таблице по коду клиента

Далее для некоторых операций приводятся примеры реализации.

{Проверка на оригинальность кода клиента}

function CheckIndex(var f:PUzel; ind:int64):boolean;

var a:PUzel;

begin

result:=False;

a:=f;

while (a<>nil) do //пробег по списку

begin

if ind = a^.x.index then exit; //если нашлось совпадение

a:=a^.next; //движение дальше

end;

result:=True; //нет совпадений

end;

{Проверка на правильность даты}

function CheckData(s:string):boolean;

var two,four:integer;

c1,c2:char;

begin

result:=false;

if (Length(s)<>10) then exit; //длина даты не равна 10 символам

if TryStrToInt(s[1]+s[2],two) = False then exit; //если ДД не численный

if (two >31) or (two <=0) then exit; //если ДД вне возможного диапазона

if TryStrToInt(s[4]+s[5],two) = False then exit; //если ММ не численный

if (two >12) or (two<=0) then exit; //если ММ вне диапазона

if TryStrToInt(s[7]+s[8]+s[9]+s[10],four) = False then exit; //если ГГГГ не численный

c1:=s[3]; c2:=s[6];

if ((c1='.') and (c2='.')) or ((c1='/') and (c2='/')) then result:=true

else exit; //символы между числами неверны

end;

{Построить список; f - указатель на голову списка}

procedure BuildSpisok(var f: PUzel); {f - указатель на голову списка}

var

a,d :PUzel; //а – указатель на созд. узел; d - на последний узел списка

ch: char; //переменная для выхода из процедуры

i:integer; //кол-во клиентов

flag:boolean; //флажок для проверки ввода

S:string; //проверяемая строка даты

begin

f:=nil;

i:=1;

repeat

new(a); //выделяем память под узел а

with a^.x do//заполняем данные

begin

repeat//код клиента

flag:=TryStrToInt64(InputBox('('+IntToStr(i)+'-й Клиент) Введите номер клиента',' ',''),index);

if flag = false then ShowMessage ('Несовместивый тип ввода!')

else if (i>1) then

begin

flag:=CheckIndex (f,index);

if flag = False then//проверка на оригинальность кода клиента

ShowMessage ('Такой код клиента уже существует!');

end;

until (flag=True);

repeat //сумма вклада

flag:=TryStrToFloat(InputBox('Введите сумму вклада','руб. ',''),SumVK);

if flag = false then ShowMessage ('Несовместимый тип ввода!');

until (flag=True);

repeat //проценты

flag:=TryStrToFloat(InputBox('Введите проценты вклада','% ',''),ProcVK);

if flag = false then ShowMessage ('Несовместимый тип ввода!');

until (flag=True);

repeat //дата пополнения

s:=InputBox('Введите дату пополнения вклада','ДД.ММ.ГГГГ ','');

flag:=CheckData (s);

if flag=false then showmessage ('Неправильный ввод даты!')

else DataPlus:=s;

until (flag=true);

repeat //сумма пополнения

flag:=TryStrToFloat(InputBox('Введите сумму пополнения вклада','руб. ',''),SumPlus);

if flag = false then ShowMessage ('Несовместимый тип ввода!');

until (flag=True);

repeat //дата изымания

s:=InputBox('Введите дату изымания вклада','ДД.ММ.ГГГГ ','');

flag:=CheckData (s);

if flag=false then showmessage ('Неправильный ввод даты!')

else DataMinus:=s;

until (flag=true);

repeat //сумма изымания

flag:=TryStrToFloat(InputBox('Введите сумму изымания вклада','руб. ',''),SumMinus);

if flag = false then ShowMessage ('Несовместимый тип ввода!');

until (flag=True);

end;

if (f = nil) then

begin//если нет первого элемента

AddFirst (f,a); //вставляем узел а в голову списка f

d:= f; //запомнить указатель на последний узел списка

end

else

begin//если уже есть первый элемент

AddAfter (d,a); //добавляем узел в хвост списка

d := a; //запомнить указатель на последний узел списка

end;

ch:=InputBox ('Для завершения ввода нажмите Y',' ',' ')[1];

inc(i);

until (ch = 'Y') or (ch ='y');

end;

{сортировка с возрастанием кода клиента}

procedure SortDataUp(Var f:PUzel; row:integer);

var temp:int64;//карман для кода клиента

i:integer; //переменная для цикла

change: boolean; //флажок для фиксации изменений

ptr:PUzel;

begin

for I := 1 to (row-1) do //совершить row-1 проходов

begin

ptr:=f;

REPEAT

change := false;

while not(ptr=nil) do //обход списка

begin

if ptr^.next<>nil then//если существует следующий элемент

begin

if ptr^.x.index>ptr^.next^.x.index then//сравнение со следующим элементом

begin

{обмениваем}

temp:=ptr^.x.index;

ptr^.x.index:= ptr^.next^.x.index;

ptr^.next^.x.index:= temp;

change:=true; //произошло изменение

break;

end;

end;

ptr := ptr^.next; // двигаемся на следующий элемент

end;

UNTIL change = false;

end;

end;

{сортировка с убыванием кода клиента}

procedure SortDataDown(Var f:PUzel; row:integer);

var temp:int64;//карман для кода клиента

i:integer;//переменная для цикла

change: boolean;//флажок для фиксации изменений

ptr:PUzel;

begin

for I := 1 to (row-1) do //совершить row-1 проходов

begin

ptr:=f;

REPEAT

change := false;

while not(ptr=nil) do //обход списка

begin

if ptr^.next<>nil then//если существует следующий элемент

begin

if ptr^.x.index<ptr^.next^.x.index then//сравнение со следующим элементом

begin

{обмениваем}

temp:=ptr^.x.index;

ptr^.x.index:= ptr^.next^.x.index;

ptr^.next^.x.index:= temp;

change:=true; //произошло изменение

break;

end;

end;

ptr := ptr^.next; // двигаемся на следующий элемент

end;

UNTIL change = false;

end;

end;

{Вывод списка в текстовый файл}

procedure WriteSpText (var f: PUzel; var ftxt:Text);

var

p: PUzel;

i: integer;

begin

p:= f;

i:=0;

while not(p = nil) do //пробег по списку

begin

inc(i);

write(ftxt,i:6);

with p^.x do

begin

write(ftxt,index:13);

write(ftxt,SumVK:14:2);

write(ftxt,procVK:10:2);

write(ftxt,DataPlus:17);

write(ftxt,SumPlus:18:2);

write(ftxt,DataMinus:15);

write(ftxt,SumMinus:16:2);

end;

p:=p^.next;

end;

end;

{найти запись по коду клиента}

procedure FindZp(var a:PUzel; digit:int64);

var f:PUzel;

i:integer; //номер строки в таблице

begin

f:=a;

i:=0;

while (f<>Nil) do //пробег по списку

begin

inc(i);

if (f^.x.index=Digit) then //если нашлось совпадение

begin

showmessage ('Запись находится на '+IntToStr(i)+'-й строчке таблицы');

exit;

end;

f:=f^.next;

end;

ShowMessage ('Такой записи не существует');

end;

{Выделить из списка узел и удалить его}

procedure DelElement(row,num:integer;Digit : int64; var a: PUzel);

Var f, df :PUzel ;

begin

if num=1 then//если удаляется начальный элемент

begin

f := a; //Запомним адрес первого элемента списка

a := a^.Next; //Теперь а указывает на второй элемент списка

Dispose(f); //Освободим память, занятую переменной f^

end;

if num=row then// если удаляется конечный элемент

begin

//Найдем предпоследний элемент

f := a;

df :=a;

while f^.Next<>Nil do

Begin

df := f;

f := f^.Next;

End;

//Удаляем элемент f^ из списка и освобождаем занимаемую им память

df^.Next := Nil;

Dispose(f);

end;

if (num>1) and (num<row) then// если где-то в середине списка

begin

f := a; //Переменная f для хранения адреса удаляемого элемента

//Найдем адреса нужных элементов списка

while(f<>Nil) and(f^.x.index<>Digit) do

Begin

df := f;

f := f^.Next;

End;

df^.Next := f^.Next;

Dispose(f);

end;

end;

# Глава 2. Интерфейс приложения

## 2.1 Описание компонентов

Среда разработки Delphi ориентирована, прежде всего, на создание программ для Windows. При этом большое внимание уделяется возможности визуальной разработки приложений с помощью большого набора готовых компонентов Delphi, позволяющих избежать ручного кодирования. Компоненты Delphi охватывают практически все аспекты применения современных информационных технологий [2].

Конечно, для работы в Delphi прежде всего требуется изучить базовые компоненты Delphi, которые требуются при подготовке практически любого приложения. В этом параграфе разберём какие компоненты Delphi использовать и для чего, а также как их добавить. Начнём с последнего.

Добавить компонент на форму можно двумя способами:

А. Перетащить из паллеты иконку нужного компонента на любое место формы окна.

Б. Сделать щелчок ЛКМ на иконку нужного компонента, а потом сделать щелчок ЛКМ на любое место формы.

В Delphi паллета располагается ниже главного меню, в соответствии с рисунком 2.1 – выделена красным.

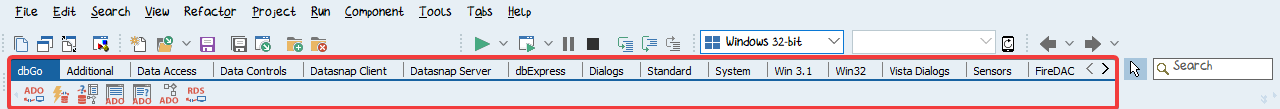


Рисунок 2.1

В данной курсовой работе используется четыре основных компонента: главное меню, таблица, строки состояния и вызов окна открытия файлов.

Теперь разберём каждый компонент:

1. Главное меню (тип TMainMenu). Компонент представляет из себя верхнюю строку из элементов типа TMenuItem, каждый из которых позволяет выполнить ту или иную операцию, в соответствии с рисунком 2.3

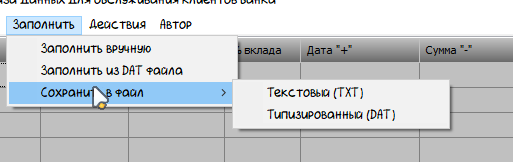


Рисунок 2.3

2. Таблица (тип TStringGrid). Его можно использовать для хранения и даже редактирования данных, однако в данной работе компонент используется только для отображения списка.

3. Строки состояния (тип TLabel). Представляет из себя строчку текста. В данной работе используется 3 таких компонента: первый – для подсчёта количества узлов списка, второй – для проверки доступности текстового файла, третий – как и второй, но для типизированного файла.

4. Вызов окна открытия файлов (тип TOpenDialog). Нужен для вызова окна, в котором выбирается открываемый файл. Визуально появляется при нажатии на кнопку меню Файл –> Открыть <…> файл.

## 2.2 Реализация обработчиков событий

Далее создаём обработчики событий. Большинство обработчиков требуется для элементов меню, поэтому достаточно сделать двойной щелчок ЛКМ по нужному элементу меню, и сразу же открывается редактор кода и вносится процедура. Вот уже готовые обработчики для кнопок меню:

{добавить узлы в хвост списка}

procedure TF\_Curs.NAddUzClick(Sender: TObject);

begin

AppUzels(PList);//основная процедура добавления узлов

row:=0;//обнуляем счётчик строк

RefreshTable;//обновляем таблицу

end;

{закрыть типизированный файл}

procedure TF\_Curs.NCloseTipClick(Sender: TObject);

begin

ifflagtip = Falsethen//проверка на доступность файла

begin

showmessage('Файл уже отключен');//вывод ответного сообщения

exit;

end;

CloseFile(ftip);//закрываем файл

ShowMessage('Типизированный файл закрыт');//вывод ответного сообщения

LTip.Caption:='Типизированный файл: Отключен';//обновляем строку состояния файла

flagtip:=False;//флажок о том, что файл отключен

end;

{закрыть текстовый файл}

procedure TF\_Curs.NCloseTxtClick(Sender: TObject);

begin

ifflagtxt = Falsethen//проверка на доступность файла

begin

showmessage('Файл уже отключен');

exit;

end;

CloseFile(ftxt);

ShowMessage('Текстовый файл закрыт');

LTxt.Caption:='Текстовый файл: Отключен';

flagtxt:=False;

end;

{полностью удалить список}

procedure TF\_Curs.NDelistClick (Sender: TObject);

begin

DelSpisok(PList);//основная процедура удаления всего списка

ShowMessage('Список удалён');//вывод ответного сообщения

ClearTable;//возвращаем исходный вид таблицы

end;

{удалить конкретный узел списка}

Procedure TF\_Curs.NDelElemClick(Sender: TObject);

var num:integer; //номер строки из таблицы

digit:int64;//переменная для кода клиента

begin

digit:=DelNum(num);//получаем код клиента из выбранной строки

DelElement(row,num,digit,PList);//удаляем элемент из списка

row:=0;

RefreshTable;

end;

{вызов справочной информации}

procedure TF\_Curs.NHelpClick (Sender: TObject);

begin

ShowMessage('Курсовая работа №2' +#10 + 'Выполнил: Артемьев А. группа: ИВТ-19-1');

end;

{создать список ручным способом}

procedure TF\_Curs.NListBuildClick(Sender: TObject);

begin

BuildSpisok(PList);//основная процедура

RefreshTable;

end;

{создать список из типизированного файла}

procedure TF\_Curs.NListBuildFFClick(Sender: TObject);

begin

if flagtip = False then//проверка на доступность файла

begin

showmessage('Файл не подключен');

exit;

end;

BuildSpisokFF(PList, ftip);//основная процедура создания списка

RefreshTable;

end;

{открыть тип. файл}

procedure TF\_Curs.NOpenTipClick(Sender: TObject);

var

s: string;

begin

if not(OpenDialog1.Execute) then exit;//если закрыто окно, то будет выход из процедуры

s := OpenDialog1.FileName;

AssignFile(ftip,s); //подключаем файл по пути FileName

reset(ftip);//открываем для чтения

LTip.Caption:='Типизированный файл: Подключен';//обновляем строку состояния

flagtip:=True;//флажок о том, что файл подключен

end;

{открыть тхт файл}

procedure TF\_Curs.NOpenTxtClick (Sender: TObject);

var

s: string;

begin

if not OpenDialog1.Execute then exit;

s := OpenDialog1.FileName;

AssignFile(ftxt,s);

Append(ftxt);

LTxt.Caption:='Текстовый файл: Подключен';

flagtxt:=True;

end;

{сохранить список в типизированный файл}

procedure TF\_Curs.NSaveTipClick (Sender: TObject);

begin

if flagtip = False then//проверка на доступность файла

begin

showmessage('Файл не подключен');

exit;

end;

WriteSpTip(PList,ftip);//основная процедура сохранения

ShowMessage('Сохранение в типизированный файл выполнено');

end;

{сохранить список в текстовый файл}

procedure TF\_Curs.NSaveTxtClick(Sender: TObject);

begin

if flagtxt = False then//проверка на доступность файла

begin

showmessage('Файл не подключен');

exit;

end;

writeln(ftxt,' # Код клиента Сумма вклада Проценты Дата пополнения Сумма пополнения Дата изымания Сумма изымания');

WriteSpText(PList,ftxt);

writeln(ftxt,' ');

ShowMessage('Сохранение в текстовый файл выполнено');

end;

{сортировка возрастания по коду клиента}

procedure TF\_Curs.NSortDownClick(Sender: TObject);

begin

SortDataDown(PList,row); //основнаяпроцедура

row:=0;

RefreshTable;

end;

{сортировка убывания по коду клиента}

procedure TF\_Curs.NSortUpClick (Sender: TObject);

begin

SortDataUp(PList,row); //основная процедура

row:=0;

RefreshTable;

end;

{поиск записи по коду клиента}

procedure TF\_Curs.NFindZpClick (Sender: TObject);

var digit:int64;

begin

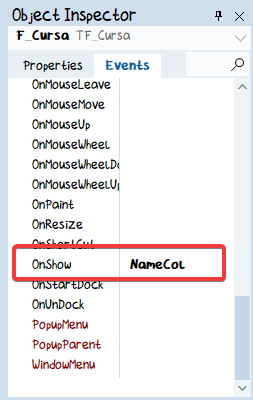
digit:=StrToInt(InputBox('Введите код клиента',' ',''));

FindZp(PList,digit);

end;

Это были все обработчики событий для кнопок меню. Но в данной работе есть один обработчик, который не используется в меню. В соответствии с рисунком 2. 4, это именование столбцов, который активируется при отображении формы (для этого в инспекторе объектов в разделе events в свойстве OnShow вписывается имя процедуры).

procedure TF\_Curs.NameCol(Sender: TObject);

begin

StrGrid.Cells[0,0]:='#';

StrGrid.Cells[1,0]:='Код клиента';

StrGrid.Cells[2,0]:='Сумма вклада';

StrGrid.Cells[3,0]:='Проценты';

StrGrid.Cells[4,0]:='Дата пополнения';

StrGrid.Cells[5,0]:='Сумма пополнения';

StrGrid.Cells[6,0]:='Дата изымания';

StrGrid.Cells[7,0]:='Сумма изымания';

end; Рисунок 2.4

# Глава 3. Тестирование созданного приложения, проверка полученных результатов

При тестировании созданного приложения необходимо выяснить, каковы его плюсы: простота и понятность интерфейса программы, что не требует дополнительного обучения или прохождения специальных курсов; также нетребовательность к системным ресурсам машины; легкость ввода и отслеживания, что немаловажно для такой программы [7].

Открываем готовое приложение и приступаем к тестированию:

3.1 Создание списка

Шаг 1. Согласно рисунку 3.1.1, в меню, в разделе «Заполнить» нажимаем на кнопку «Заполнить вручную».

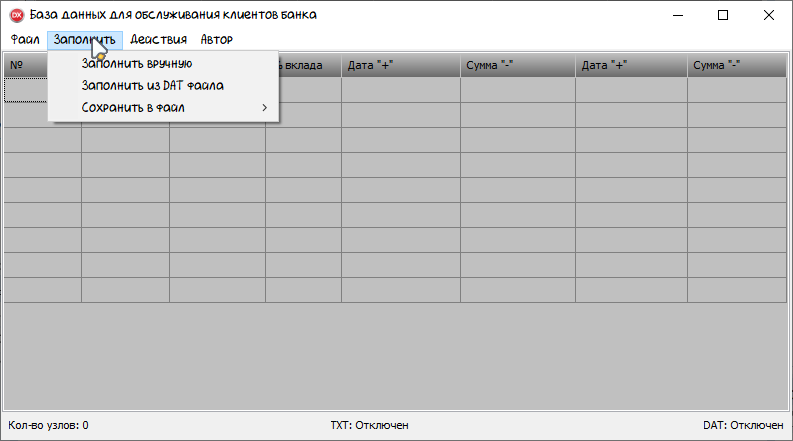


Рисунок 3.1.1

Шаг 2. Появляется InputBox и вводим данные, согласно рисунку 3.1.2. Проделываем это семь раз.

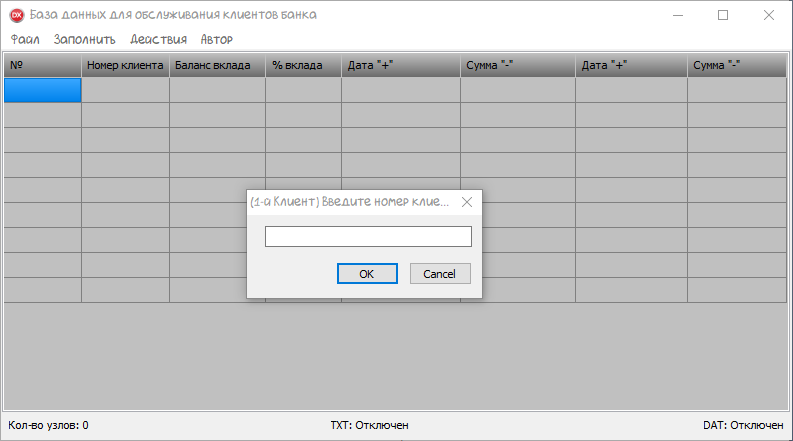


Рисунок 3.1.2

Шаг 3. Согласно рисунку 3.1.3, на восьмой раз появляется запрос. Вводим символ «Y» или «y», если хотим заполнить следующий узел, и возвращаемся к Шагу 2. Вводим любой символ, если хотим закончить заполнение списка.

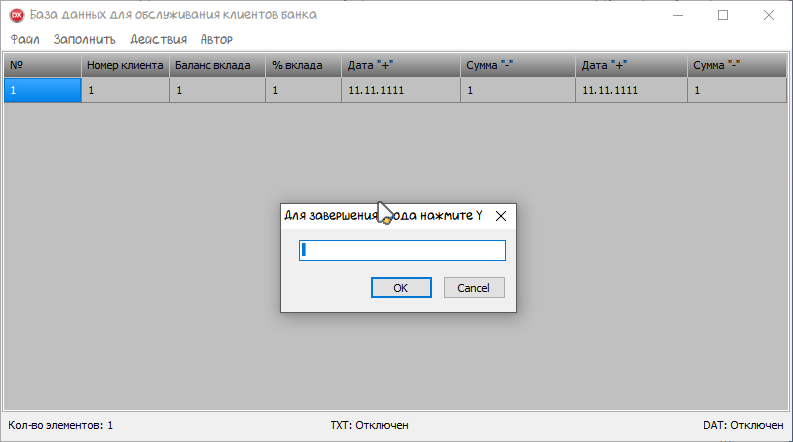
Дополнительно: с помощью команды «Добавить узлы в хвост» можно также создать список, если заранее известно кол-во узлов. 

Рисунок 3.1.3

3.2 Редактирование списка

Шаг 1. Чтобы добавить новые узлы, в разделе «Действия», согласно рисунку 3.2.1 выберите «Добавить узлы».

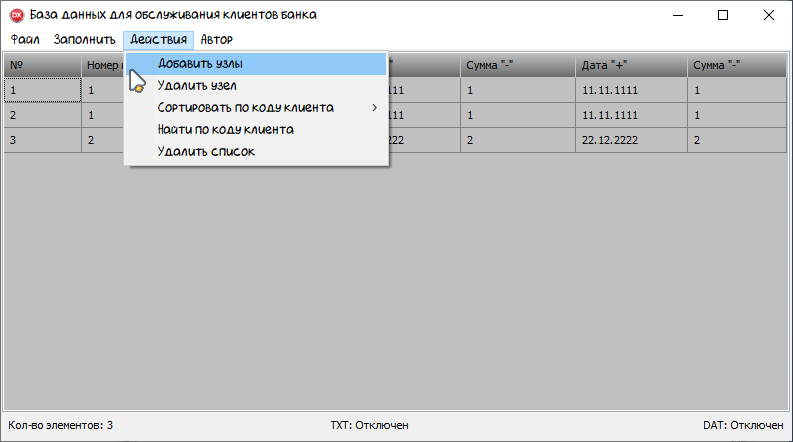


Рисунок 3.2.1

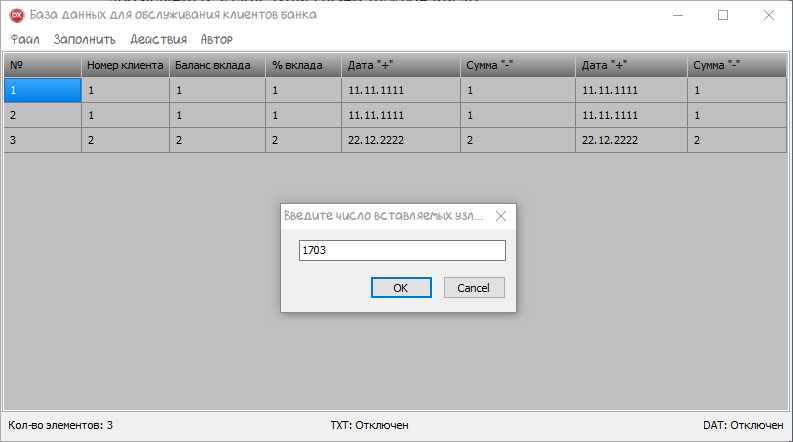
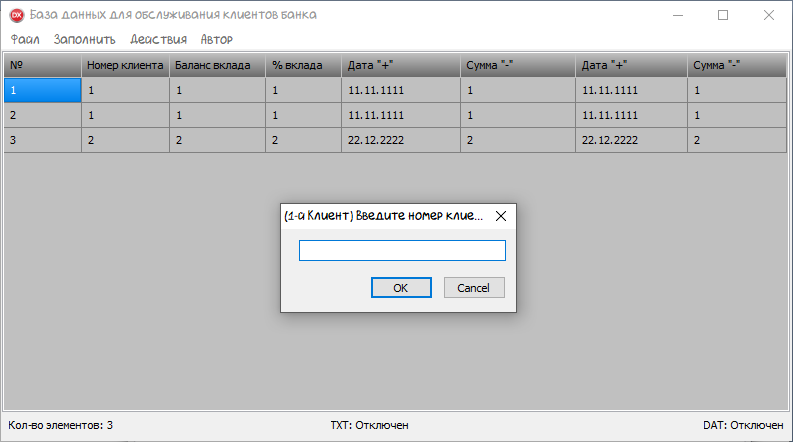
Шаг 2. Согласно рисунку 3.2.2, появляется запрос на количество добавляемых узлов. Вписываем нужное число.

Рисунок 3.2.2

Шаг 3. Согласно рисунку 3.2.3, появляется InputBox. Вводим данные.

Рисунок 3.2.3

3.3 Удаление конкретного узла

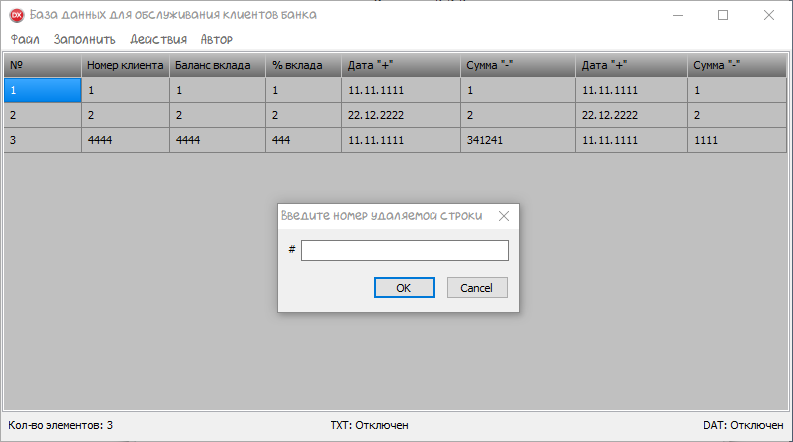
Для этого просто в разделе «Действия» выбираем «Удалить элемент», а затем, согласно рисунку 3.3. вводим номер строчки из таблицы. 

Рисунок 3.3.

3.4 Удаление списка

Для этого просто в разделе «Действия» выбираем «Удалить список». После этого таблица вернётся в начальный вид (рис. 3.4).

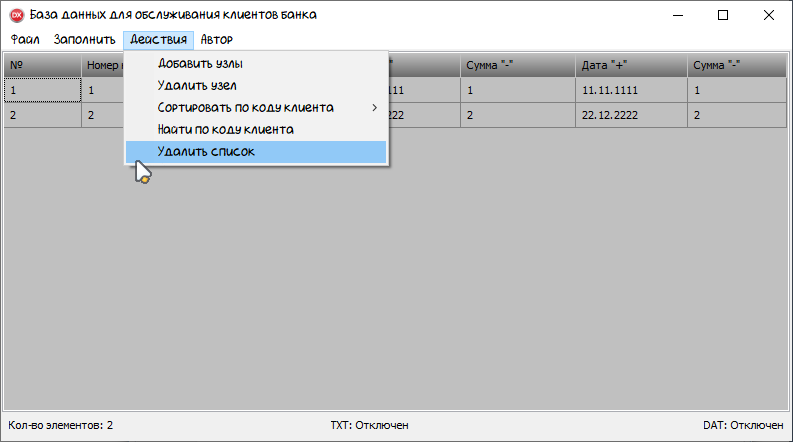


Рисунок 3.4.

3.5 Другие операции

Сортировка: в разделе «Действия», согласно рисунку 3.5.1, выберите «Сортировать коды клиента», а дальше на выбор.

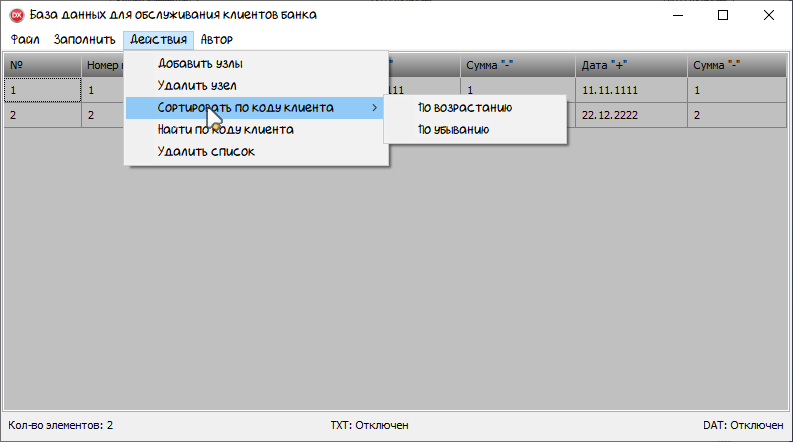


Рисунок 3.5.1

3.6 Поиск записи по коду клиента:в разделе «Действия» выберите «Найти по коду клиента». Далее появится запрос на код клиента (рис. 3.6.1):

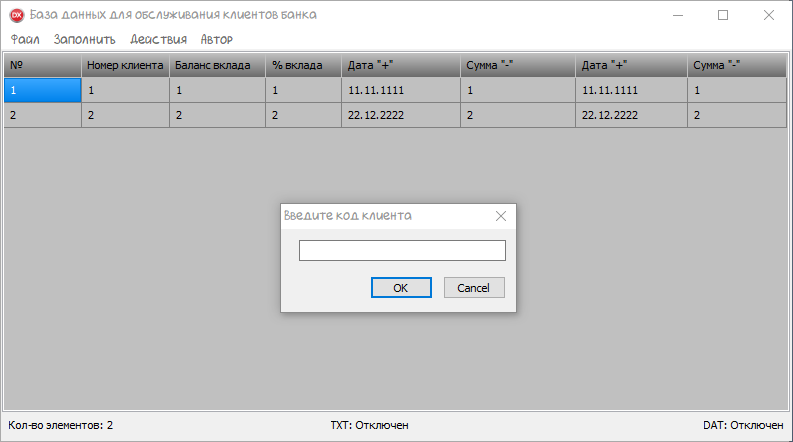


Рисунок 3.6.1

А потом, если такой номер существует, то появится сообщение

(рис.3.6.2):

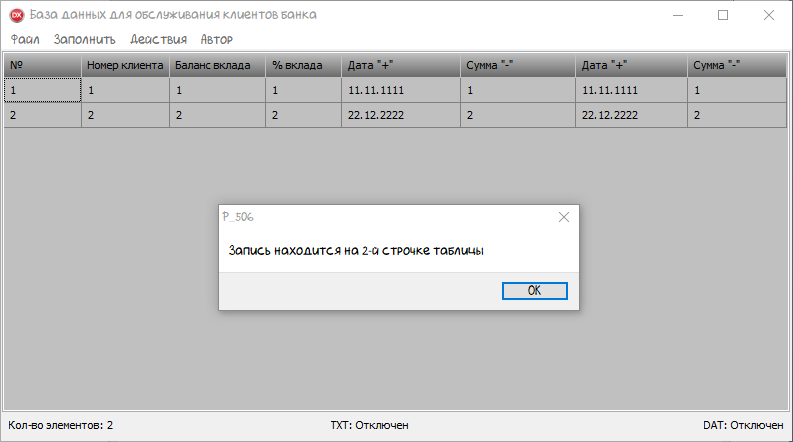


Рисунок 3.6.2

3.7 Использование файлов

Данная программа поддерживает текстовые (txt) и типизированные (dat) файлы. Txt используется для вывода таблицы, а dat – для сохранения ранее вводимых данных.

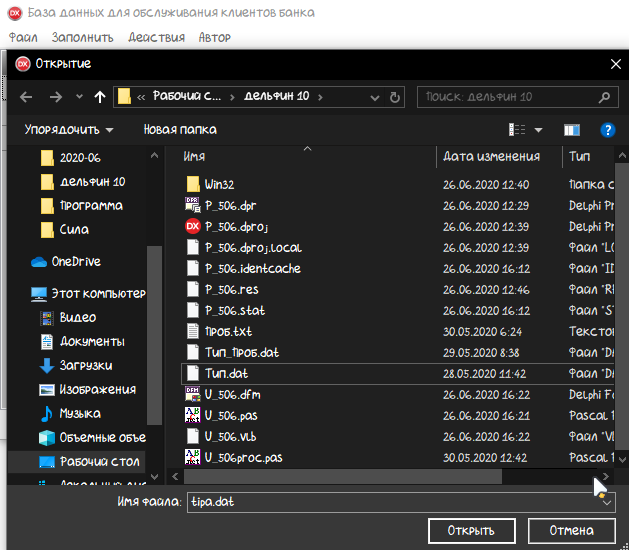
Открытие: В разделе «Файл» выбираем «Открыть <…>». После же откроется окно для выбора файла на диске (рис. 3.7.1):

Рисунок 3.7.1

Примечание: если нет заранее созданного файла, то его можно создать прямо в этом окне. Для этого нужно вызвать подменю в области списка файлов, затем выбрать «Создать» -> «Текстовый документ», а потом переименовать его с нужным именем и разрешением.

Сохранение: Выбираем в разделе «Список» команду «Сохранить в <…> файл».  
Если это будет типизированный, то сохранятся данные списка.  
Если это будет текстовый, то в файл будет выведена таблица (без линий).

Так примерно выглядит сохранённый текстовый файл (рисунок 3.7.2)

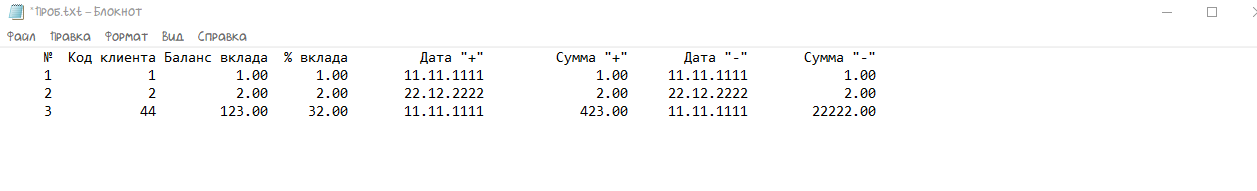


Рисунок 3.7.2

Импорт данных*:* Выбираем в разделе «Список» команду «Создать из тип. файла». Если в таблице были узлы, это операция дополнит список.

Примечание: Если dat-файл был открыт, а потом вы удалили список, то при повторном импорте уже не добавятся данные из файла. Однако в Проводнике открытый dat-файл не будет пуст. Поэтому если закрыть, а потом снова открыть файл, то данные при импорте уже появятся.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе были рассмотрены все поставленные задачи и обсуждены общие принципы создания баз данных и их реализации на языке программирования Паскаль

При создании приложения была использована книга А. Я. Архангельского «Программирование в Delphi 7», справка из программ «Delphi 10» и «PascalABC», а также выполненные задания по дисциплинам «Человеко-машинное взаимодействие» и «Программирование».

Самостоятельно были изучены компоненты типа TMainMenu, TStringGrid и TLabel.

Всего программа содержит пятнадцать операций с учётом вариаций:

- четыре в разделе «Файл»;

- четыре в разделе «Список»;

- шесть в разделе «Действия»;

- одна – кнопка меню «Справка».

Все операции работают устойчиво. Возможно, при работе с InputBox’и в некоторых операциях могут выйти стандартные ошибки компилятора, но последствия не вызывают критических сбоев программы.

Руководство составлено в целом понятно и подробно, так как использовались иллюстрации и примечания.

При написании данной курсовой работы были выполнены поставленные задачи по написанию базы данных клиентов банка:

- Поиск информации для написания курсовой в различных источниках;

- Изучение материала по Delphi;

- Изучение языка программирования Pascal;

- Написание программы;

- Тестирование программы.

Таким образом, поставленные цели достигнуты.

При написании и разработке курсового проекта также были закреплены знания по работе с Borland Pascal, получены навыки использования справки и других функций языка Pascal.

Проделанная работа позволяет любому пользователю с легкостью создавать большие объемы информации, обрабатывать их, редактировать, делать выборки по определенным критериям и т.д. Использование такой программы значительно облегчает деятельность человека.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамов, В.Г. Введение в язык Паскаль / В.Г. Абрамов, Н.П. Трифонов. - Москва: Наука, 2011. - 320 c.

2. Архангельский, А.Я. Программирование в Delphi 7 /А.Я. Архангельский - Москва: ООО «Бином-Пресс», 2018 . - 1152 с.

3. Белецкий, Я.Н. Турбо Паскаль для IBM / Я.Н. Белецкий – Москва: Машиностроение, 2001. – 762 с.

4. Епанешников, А.М. Программирование в среде Turbo Pascal 7.0 / А.М. Епанешников, В.А. Епанешников. - Москва: ДИАЛОГ-МИФИ; Издание 4-е, испр., 2014. - 367 c.

5. Культин, Н.Б. Turbo Pascal в задачах и примерах / Н.Б. Культин. - Москва: БХВ, 2011. - 256 c.

6. Касторнова, В.А. Структуры данных и алгоритмы их обработки на языке программирования Паскаль: учебное пособие / В.А. Касторнова. – Санкт-Петербург: BHV, 2016. — 304 c.

7. Павловская, Т.А. Паскаль. Программирование на языке высокого уровня / Т.А. Павловская. – Санкт-Петербург: Питер, 2012. - 393 c.

8. Поддубная, Л.М. Мне нравится Паскаль / Л.М. Поддубная, В.Ф. Шаньгин. - Москва: Радио и связь, 2015. - 160 c.

9. Семашко, Г.Л. Программирование на языке Паскаль / Г.Л. Семашко, А.И. Салтыков. - М.: Наука, 2011. - 128 c.

10. Фаронов, В.В. Turbo Pascal 7.0: учебное пособие / В.В. Фаронов. - Москва: Нолидж, 2012. - 211 c.

11. Форсайт, Р. Паскаль для всех / Р. Форсайт. - Москва: Машиностроение, 2016. - 288 c.

**ЭЛЕКТРОННЫЕ РЕСУРСЫ**

12. Множества в языке Pascal [Электронный ресурс] / Режим доступа: http://comp-science.narod.ru/Progr/mn.htm – Загл. с экрана.

13. Общие понятия о базах данных [Электронный ресурс] / Режим доступа: http://www.maksakov-sa.ru/BDDelphi/ObSvedBDDelphi/index.html - Загл. с экрана.

14. Система управления базами данных [Электронный ресурс] / Режим доступа: http://www.infosgs.narod.ru/31.htm - Загл. с экрана.

15. Турбо Паскаль 7.0 - Записи [Электронный ресурс] / Режим доступа: http://klax.tula.ru/~zet/frames/thm\_records.html - Загл. с экрана.

16. Типизированные файлы [Электронный ресурс] / Режим доступа: http://pas1.ru/typed-file - Загл. с экрана.

# ПРИЛОЖЕНИЕ

Код интерфейсной части модуля формы:

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, U\_506proc, Vcl.Menus, Vcl.StdCtrls,

Vcl.Grids;

type

TF\_Curs = class(TForm)

MainMenu1: TMainMenu;

OpenDialog1: TOpenDialog;

NFile: TMenuItem;

NOpenTxt: TMenuItem;

NCloseTxt: TMenuItem;

NList: TMenuItem;

NListBuild: TMenuItem;

NSaveTxt: TMenuItem;

NDelist: TMenuItem;

NSaveTip: TMenuItem;

NOpenTip: TMenuItem;

NCloseTip: TMenuItem;

NListBuildFF: TMenuItem;

NDelElem: TMenuItem;

NAddUz: TMenuItem;

Продолжение приложения А

NHelp: TMenuItem;

NAct: TMenuItem;

StrGrid: TStringGrid;

LTxt: TLabel;

LTip: TLabel;

LCount: TLabel;

NFindZp: TMenuItem;

NSort: TMenuItem;

NSortUp: TMenuItem;

NSortDown: TMenuItem;

procedure NOpenTxtClick(Sender: TObject);//открыть тхт файл

procedure NCloseTxtClick(Sender: TObject);//закрыть тхт файл

procedure NSaveTxtClick(Sender: TObject);//сохранить список в тхт файл

procedure NOpenTipClick(Sender: TObject);//открыть тип. файл

procedure NCloseTipClick(Sender: TObject);//закрыть тип. файл

procedure NSaveTipClick(Sender: TObject);//сохранить данные в тип. файл

procedure NListBuildClick(Sender: TObject);//составить список вручную

procedure NListBuildFFClick(Sender: TObject); //составить список из файла

procedure NDelElemClick(Sender: TObject); // удалить конкретный элемент

procedure NDelistClick(Sender: TObject); // удалить весь список

procedure NAddUzClick(Sender: TObject); // добавить узлы в хвост списка

procedure NHelpClick(Sender: TObject); // вывоз справочной информации

procedure NameCol(Sender: TObject); //называние столбцо втаблицы

procedureRefreshTable; //обновление таблицы после действий

procedureClearTable; //очищение таблицы после удаления списка

Продолжение приложения А

functionDelNum(varnum:integer):int64; //получение кода клиента

procedureNFindZpClick(Sender: TObject); //поиск записи по коду клиента

procedureNSortUpClick(Sender: TObject); //сортировка возрастания по коду клиентов

procedure NSortDownClick(Sender: TObject);//и убывания

private

{ Private declarations }

public

{ Public declarations }

end;

var

F\_Curs: TF\_Curs;

PList: PUzel; //указатель на список

ftip: FZap; //тип. файловая переменная

ftxt: textfile; //текст. файловая переменная

Row,DC: integer; //счётчик узлов и счётчик удалённых строк

flagtip,flagtxt:boolean; //индикаторы подключения файлов

Код интерфейсной части созданного логического модуля:

interface

usesSysUtils, Dialogs; //Модули для ввода/вывода данных

type

Client = record //Тип клиента банка

Index:int64; //Код клиента

Продолжение приложения

SumVK: double; //Величина вклада

ProcVK: double; //Проценты вклада

DataPlus: string[10]; //Дата и

SumPlus: double; //Сумма пополнения вклада

DataMinus: string[10]; //Дата и

SumMinus: double; //Сумма изыманий вклада

end;

PUzel = ^ZpTwo; //Тип указателя на узел

ZpTwo = record//Тип узла списка

x: client; //Сведения о клиенте

next: PUzel; //Указатель на следующий узел

pred: PUzel; //Указатель на предыдущий узел

end;

FZap = fileofClient; //Файловый тип для хранения данных о клиентах

{Операции для работы с двусвязанным списком }

Procedure AddFirst(varf: PUzel; a: PUzel);{Вставить узел a первым в список}

procedure AddAfter(var old:PUzel; a: PUzel);{Вставитьузел a после old}

function CheckIndex(var f:PUzel; ind:int64):boolean;{Проверка на оригинальность кода клиента}

function CheckData(s:string):boolean;{Проверка на правильность даты}

{Построить список; f - указатель на голову списка}

procedure BuildSpisok(var f: PUzel);

{Вывод списка в текстовый файл}

Продолжение приложения

procedure WriteSpText(var f: PUzel;varftxt:Text);

{Вывод списка в типизированный файл}

procedure WriteSpTip(var f: PUzel;varftip:FZap);

{Построить список из тип. файла}

procedure BuildSpisokFF(var f: PUzel; var ftip:Fzap);

{Выделить из списка первый узел и вернуть его пользователю}

procedure DelFirstElement(var f,a: PUzel);

{Выделить из списка узел, следующий за узлом old и вернуть его пользователю}

procedure DelElement(row,num:integer;digit:int64;var a: PUzel);

{Удалитьсписок}

procedure DelSpisok(var f: PUzel);

{добавить узлы в конец списка}

procedure AppUzels(var f: PUzel);

{найти запись по коду клиента}

procedure FindZp(var a:PUzel; digit:int64);

{сортировка с возрастанием даты пополнения}

procedure SortDataUp(Var f:PUzel;row:integer);

{сортировка с убыванием даты пополнения}

procedure SortDataDown(Var f:PUzel;row:integer);