Московский государственный технический университет
имени Н. Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети

«УТВЕРЖДАЮ»
Заведующий кафедрой ИУ-6
_____ Сюзев В.В.

Г. С. Иванова, Т.Н. Ничушкина

# Программирование под Windows в среде Turbo DELPHI 2006

Методические указания по выполнению лабораторной работы № 10
по дисциплине Основы программирования

Москва 2012

**Цель работы:** изучение процесса создания приложений с графическим интерфейсом в среде программирования Turbo Delphi

**Объем работы:** 6 часов

# Теоретическая часть

## 1. Введение

Интегрированная среда программирования Turbo Delphi предназначена для создания $32^x$ разрядных приложений WINDOWS. Эта среда является частью профессиональной среды программирования Delphi Studio (2006 г.) и относится к классу визуальных, в которых разработчику предоставляется возможность прямо на экране формировать интерфейс разрабатываемого программного продукта из стандартных элементов управления.

Языком программирования для среды **Turbo Delphi** является язык **Object Pascal**, являющийся дальнейшим развитием Borland/Turbo Pascal.

### *1.1. Типы файлов, используемые Turbo Delphi*

Среда **Turbo Delphi** предназначена для создания больших программ, элементы которых размещаются в разных файлах. Основной частью программы является *проект*.

Среда создает два файла программы, содержащие проект, которые имеют расширение ***.bdsproj*** (Borland Developer Studio Project File) и ***.dpr*** (Delphi Project File). Файл с расширением ***.dpr*** создается в формате, совместимом с ранними версиями **Delphi** (в частности, с **Delphi 7**) и содержит текст основной программы. Файл с расширением ***.bdsproj*** формируется в формате **Delphi Studio** и содержит информацию о проекте. Запуск любого из этих файлов в среде **Turbo Delphi** вызовет открытие проекта. Если сформированный в среде **Turbo Delphi** проект необходимо запустить в среде **Delphi 7**, следует работать с файлом ***.dpr.***

Как правило, основная программа при программировании «под Windows» формируется самой средой, но разработчик может ее изменять.

Помимо файла проекта программа может включать различные *модули* (**Unit)**, которые содержатся в файлах с расширением ***.pas***. Часть модулей стандартны и содержат процедуры и функции, выполняющие операции ввода-вывода и т.п., а остальные – добавляются разработчиком при написании программы.

Среди добавляемых модулей принято различать модули, содержащие информацию о *формах*, и модули, содержащие процедуры и функции, непосредственно связанные с решением задачи.

Кроме указанных компонент программа может использовать динамические библиотеки **DLL**, файлы которых имеют расширение ***.dll***.

При создании программы используется также библиотека стандартных компонентов **DCL** (файлы которых имеют расширение ***.dcl***), содержащая особым образом подготовленные классы.

Среда **Turbo Delphi** позволяет создавать проекты, модули форм, модули разработчика, библиотеки **DLL**, а также текстовые файлы.

После успешной компиляции программы создается исполняемый файл с именем, совпадающим с именем проекта, и расширением ***.exe***, а также файлы – результат компиляции модулей с расширением ***.dcu***.

Помимо указанных файлов при работе в **Turbo Delphi** формируются файлы ресурсов с расширением ***.res***, конфигурации с расширением ***.cfg*** – для проекта и с расширением ***.dfm*** – для модулей форм. В эти файлы помещаются параметры проекта и его компонентов, графические изображения, созданные в результате визуального программирования. Если в процессе разработки эти файлы случайно потеряются, то файл ***.res*** система предложит пересоздать, файл ***.cfg*** пересоздаст автоматически. Отсутствие же файла ***.dfm*** приведет к невозможности дальнейшей работы с проектом.

Кроме того, в директории проекта присутствуют файл с расширением ***.identcache*** и файл с расширением ***.bdsproject.local***. В них содержится информация о некоторых характеристиках проекта. При отсутствии этих файлов среда пересоздаст их.

### *1.2. Основные принципы событийного программирования*

*Событийным* называется программирование, при котором программа представляет собой набор обработчиков некоторых *событий*. В качестве событий при этом могут интерпретироваться как нажатие какой-либо “кнопки” в окне программы, так и некоторые ситуации в самой программе (например, вызов формы). Таким образом, основной цикл работы программы представляет собой ожидание какого-либо события, вызов соответствующего обработчика для обработки этого события, после чего вновь следует ожидание события, и цикл повторяется.

Такая программа не имеет алгоритма в традиционном смысле, так как связь между отдельными частями не задана жестко, а зависит от последовательности наступления тех или иных событий.

Задача разработчика в этом случае - определить множество событий для программируемой задачи и написать соответствующие обработчики. Причем **Turbo Delphi** предоставляет как стандартные обработчики некоторых событий, так и заготовки для новых, добавляемых, обработчиков.

# Последовательность выполнения работы

## Часть 1. Создание приложения Калькулятор

При вызове интегрированной среды **Turbo Delphi** на экране появляется окно, вид которого представлен на рисунке 1.

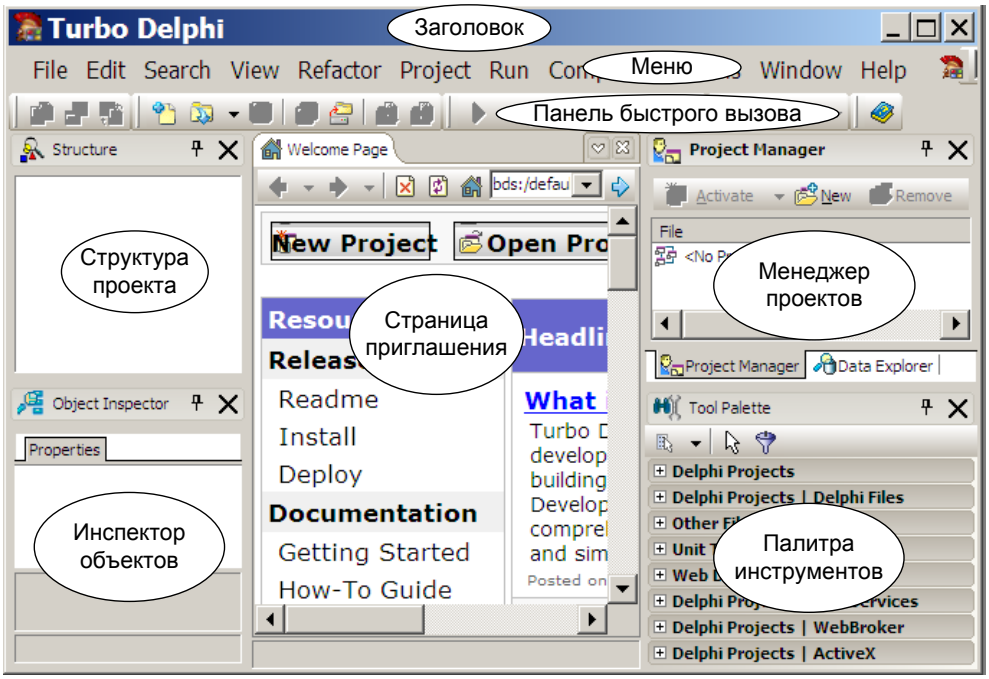


**Рисунок 1** – Вид окна Turbo Delphi при входе в среду

Последовательность действий при создании программы рассмотрим на конкретном примере.

**Задание.** Разработать программу – калькулятор, выполняющий основные арифметические действия.

Разработка программы начинается с создания нового проекта. Это можно сделать двумя способами:

- нажав закладку **New Project** на странице приглашения. После этого на экране появится окно диалога выбора типа проекта (см. рисунок 2), в котором надо выбрать значок «VCL Forms Application» и нажать его;

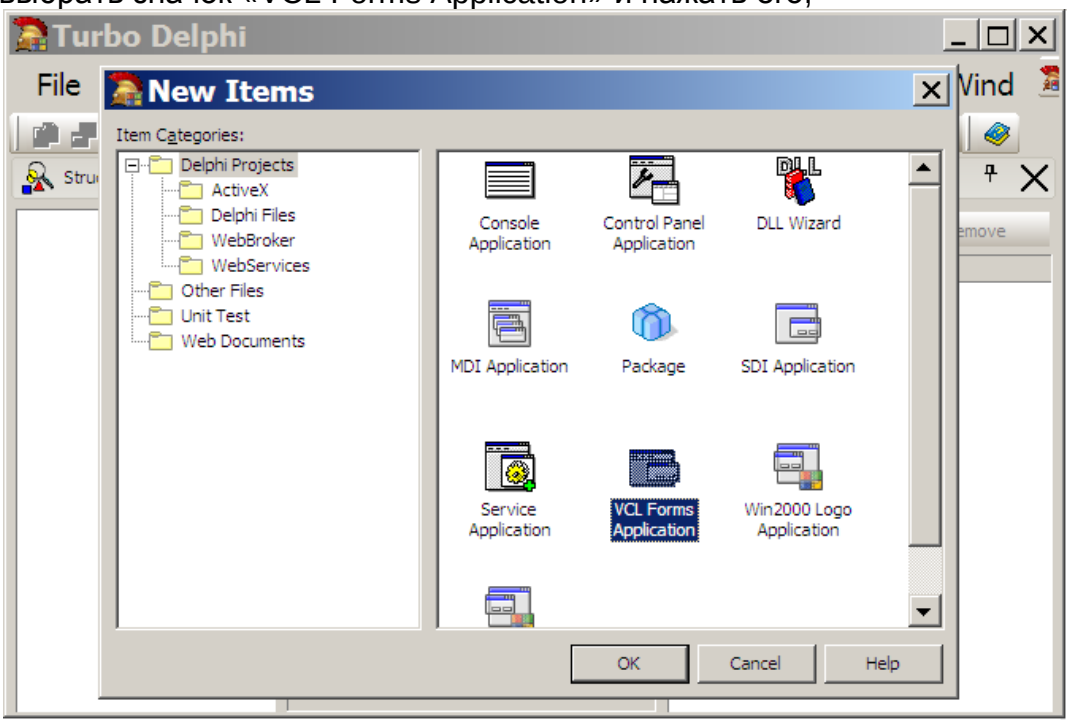


**Рисунок 2** – Вид окна выбора типа приложения

- выбрав пункт меню **File\New Project\VCL Form Application** (см. рисунок 3).

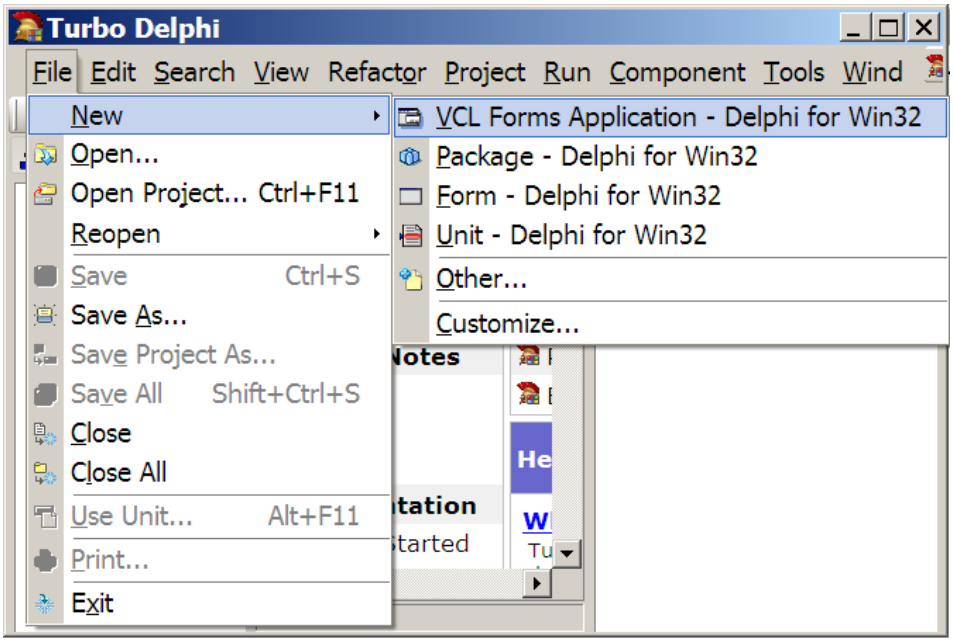


**Рисунок 3** – Вид окна при выборе типа проекта

После этого на экране появится заготовка формы проекта (см. рисунок 4).

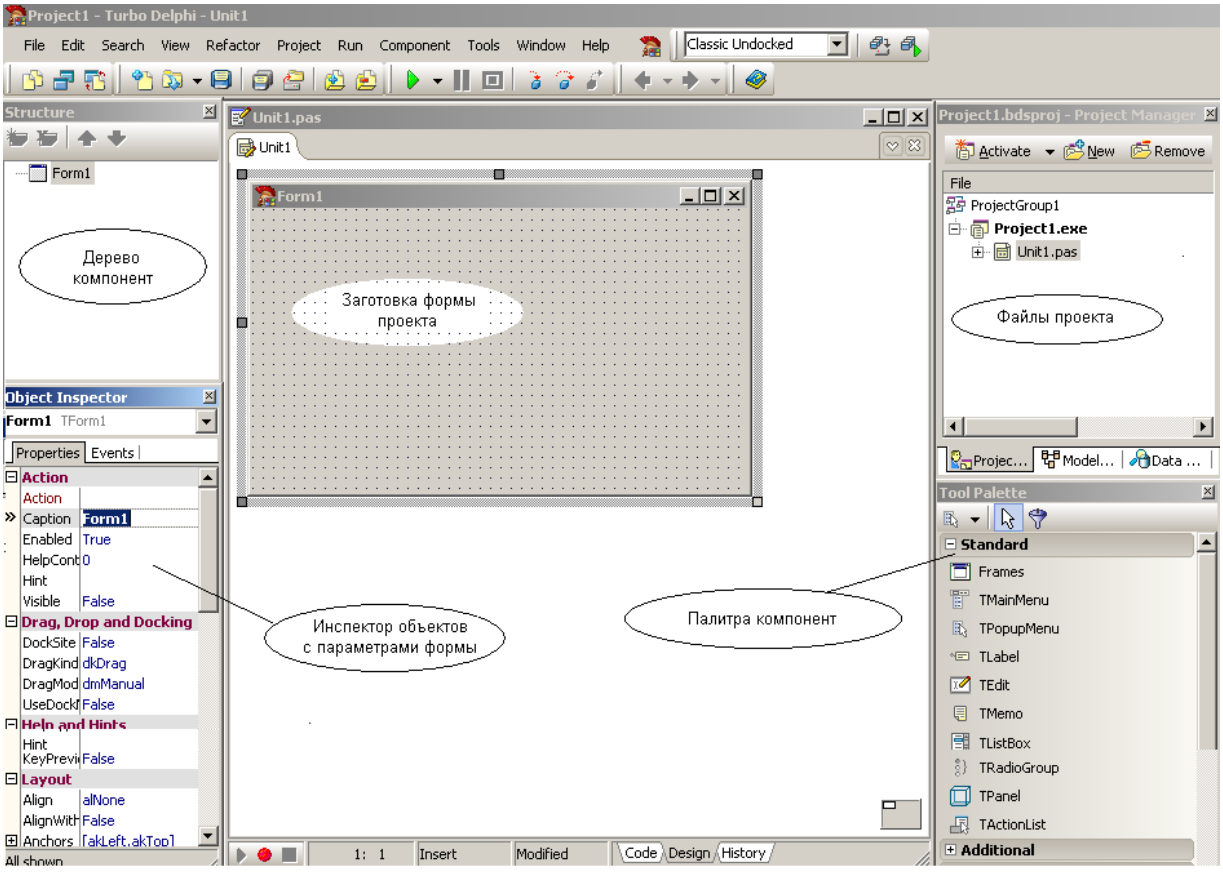


**Рисунок 4** – Вид окна приложения с заготовкой формы проекта

Продолжить создание программы необходимо с задания имен проекта и модуля. При определении имени модуля следует сначала щелкнуть либо по первой форме (**Form1**), либо по странице (**Unit1**) текстового редактора, а за-

тем определить новое имя через меню **File\Save As...** В появившемся окне создайте новую папку и введите имя модуля – **C_unit.pas**. Имя проекта определяется через меню **File\Save Project As...** (сохраните его в той же папке с именем **Culc.dpr**).

2. Измените заголовок формы **Form1** на заголовок **Калькулятор**. Для этого, предварительно выделив щелчком форму, на странице **Properties** инспектора объектов щелчком выделите свойство **Caption** (Заголовок) и введите имя **Калькулятор**.

3. Разместите на форме окно ввода чисел. Для этого на странице **Standard** палитры компонент найдите кнопку **Edit,** щелкните на ней левой кнопкой мыши. Затем щелкните левой кнопкой мыши на нужном месте формы (см. Рисунок 4). После этого измените размер компонента (тащите за черные квадратики в нужную сторону). Теперь удалите текст из окна компонента. Для этого на странице **Properties** инспектора объектов выделите свойство **Text** и удалите информацию из этого поля.

4. Разместите кнопки операций на форме. Для этого на странице **Standard** палитры компонент найдите кнопку **Button**. Для того, чтобы не перетаскивать каждый компонент отдельно, перед выбором мышкой компонента нажмите клавишу **Shift**. Теперь щелкая мышью в нужных местах можно установить сразу все 7 кнопок. Для отмены работы с кнопкой щелкните мышью по стрелке под словом **Standart** палитры компонент. Затем, последовательно щелкая мышью по установленным кнопкам, измените заголовки кнопок (свойство **Caption** на странице **Propeties** инспектора объектов) соответственно на **C**, **=**, **+**, **-**, **x**, **/** и **Quit** (см. рисунок 5).

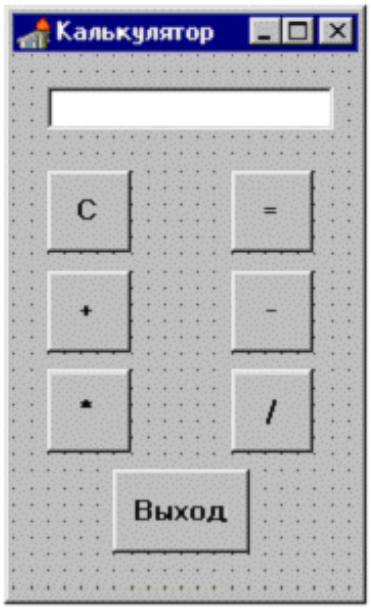


**Рисунок 5** – Форма «Калькулятор»

Одновременно с построением формы Turbo Delphi строит новый класс с именем TForm1 (потомок класса стандартного класса **TForm**) и создает объект данного класса с именем **Form1** в тексте модуля **C_unit.pas** (см. рисунок 6).

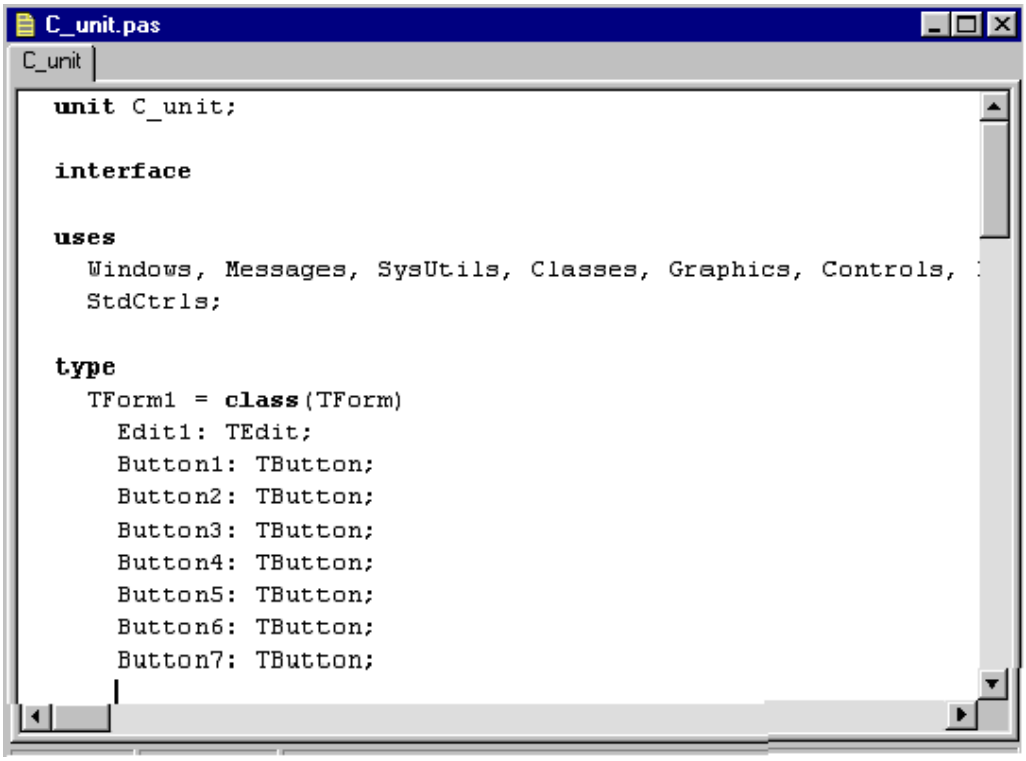


**Рисунок 6** – Фрагмент текста модуля.

Компиляцию проекта выполняют, используя Ctrl-F9 или меню Project\Compile.

***Рекомендации***. Чаще **сохраняйте** создаваемый проект. Для более быстрого поиска ошибок регулярно выполняйте компиляцию

5. Добавьте в модуль после служебного слова ***implementation*** объявление переменных, которые будут использоваться для создания проекта:

```
var     Sum:real;
        operation:char='@';
```

**6.** Создайте процедуру **Operate**, которая непосредственно выполняет вычисления**:**

```
procedure operate;
  var   s:string;
        code:integer;
        n:real;
  begin
        s:=Form1.Edit1.text;  { читаем сроку из параметра text Edit1}
       Form1.Edit1.clear;   { очищаем Edit1}
       val(s,n,code);  { преобразуем строку в число}
      case operation of       { выполняем операцию}
            '@': sum:=n;
```

```
        '+': sum:=sum+n;
        '-': sum:=sum-n;
        '*': sum:=sum*n;
        '/': sum:=sum/n;
    end;
    end;
```

7. Теперь «научите» форму обрабатывать нажатия на кнопки. Для этого щелкните мышью по кнопке **C** и перейдите на страницу **Events** инспектора объектов. На этой странице приведены все события, на которые может реагировать компонент **Button**. Щелкнув по строке **OnClick**, вы выберете событие «щелчок мыши по компоненту». Двойным щелчком по той же строке вы вставите заготовку обработчика данного события в текст модуля, определяющего реакции формы:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
end;
```

Между ***begin*** и ***end*** необходимо ввести текст процедуры:

```
Edit1.Clear;        { очистить окно компонента Edit1}
operation:='@';  { установить состояние “первая операция “}
Edit1.setfocus;   { установить активным окно компонента Edit1}
```

Аналогично введите процедуры обработки нажатий на другие клавиши:

Для кнопки **=** (***Button2***):

```
procedure TForm1.Button2Click(Sender: TObject);
var s:string;
begin
        operate;     { выполнить предыдущую операцию}
        str(sum:6:3,s);  { преобразовать результат в строку}
        Edit1.text:=s;     { вывести строку в окно компонента Edit1}
       operation:='@';
        Edit1.setfocus;   { установить курсор на кнопку Button1}
end;
```

Для кнопки **+** (***Button3***):

```
procedure TForm1.Button3Click(Sender: TObject);
begin
      operate; { выполнить предыдущую операцию}
       operation:='+'; { установить состояние "операция +"}
       Edit1.setfocus; { установить активным окно компонента Edit1}
```

```
end;
```

Для кнопки **-** (***Button4***):

```
procedure TForm1.Button4Click(Sender: TObject);
begin
    operate; { выполнить предыдущую операцию}
    operation:='-'; { установить состояние "операция +"}
    Edit1.setfocus; { установить активным окно компонента Edit1}
end;
```

Для кнопки ***** (***Button5***):

```
procedure TForm1.Button5Click(Sender: TObject);
    operate;
      operation:='*';
      Edit1.setfocus;
end;
```

Для кнопки **/** (***Button6***):

```
procedure TForm1.Button6Click(Sender: TObject);
begin
      operate;
      operation:='/';
      Edit1.setfocus;
end;
```

Для кнопки **Выход** (***Button7***):

```
procedure TForm1.Button7Click(Sender: TObject);
begin
     Close;  { Завершение работы приложения}
end;
```

8. Запустите программу на выполнение, используя либо **F9**, либо меню **Run/Run**, либо кнопку **Run** на панели быстрого доступа.

***Примечание.*** Обратите внимание, что основные интерфейсные элементы, связанные с окном программы (такие как кнопки вызова системного меню, свертывания, развертывания, завершения программы и т.д.), при использовании среды **Turbo Delphi** программируются автоматически.

## 2. Структура программы.

Созданная в предыдущем разделе программа состоит из следующих элементов:

1. Проект программы. Он был создан средой автоматически. Для просмотра (а при необходимости и изменения) проекта необходимо использовать меню **Project /View Source:**

```
program Culc;
uses   { используемые модули}
      Forms,
      C_unit in 'C_unit.PAS' {Form1};
{$R *.RES}
begin
       Application.Initialize;
       Application.CreateForm(TForm1, Form1); { создание формы}
       Application.Run;          {  основной цикл программы }
end.
```

2. Модуль **C_unit.PAS**:

```
unit Culc;
interface
uses SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
      Controls, Forms, Dialogs, StdCtrls;
type
 TForm1 = class(TForm)
       Edit1: TEdit;
       Button1: TButton;
       Button2: TButton;
       Button3: TButton;                    поля-объекты
       Button4: TButton;
       Button5: TButton;
       Button6: TButton;
       Button7: TButton;
       procedure Button1Click(Sender: TObject);
       procedure Button2Click(Sender: TObject);
       procedure Button4Click(Sender: TObject);
       procedure Button5Click(Sender: TObject);                  методы
       procedure Button3Click(Sender: TObject);
       procedure Button6Click(Sender: TObject);
       procedure Button7Click(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var   Form1: TForm1;
implementation
     var   Sum:real;
           operation:char='@';
{$R *.DFM}
procedure operate;
  var  …
  begin
   . . .
  end;
procedure TForm1.Button1Click(Sender: TObject);
begin
. . . .
end;
procedure TForm1.Button2Click(Sender: TObject);
var s:string;
begin
  . . .
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
 . . .
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
 . . .
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
 . . .
```

```
end;
procedure TForm1.Button6Click(Sender: TObject);
begin
. . .
end;
procedure TForm1.Button7Click(Sender: TObject);
begin
. . .
end;
end.
```

Кроме этого, были созданы файлы ресурсов, с которыми **Turbo Delphi** работает самостоятельно.

### Контрольные вопросы

1. Как создать проект приложения с графическим интерфейсом?

2. Поясните вид экрана при высвечивании заготовки приложения. Какую информацию на экране можно менять непосредственно, и какую через функции среды? Почему?

3. Как создается окно приложения?

4. Как перейти в режим ввода текста программы?

5. Какой принцип используется при создании приложений и почему?

6. Какова структура программы приложения? Почему файл проекта строится автоматически и практически не изменяется для различных приложений?

**Часть 2. Создание приложения Записная книжка**

**Задание 2.** Разработать приложение Записная книжка.

Примерный вид форм представлен на рисунках 7 и 8.


**Рисунок 7** – Основная форма (Form1).

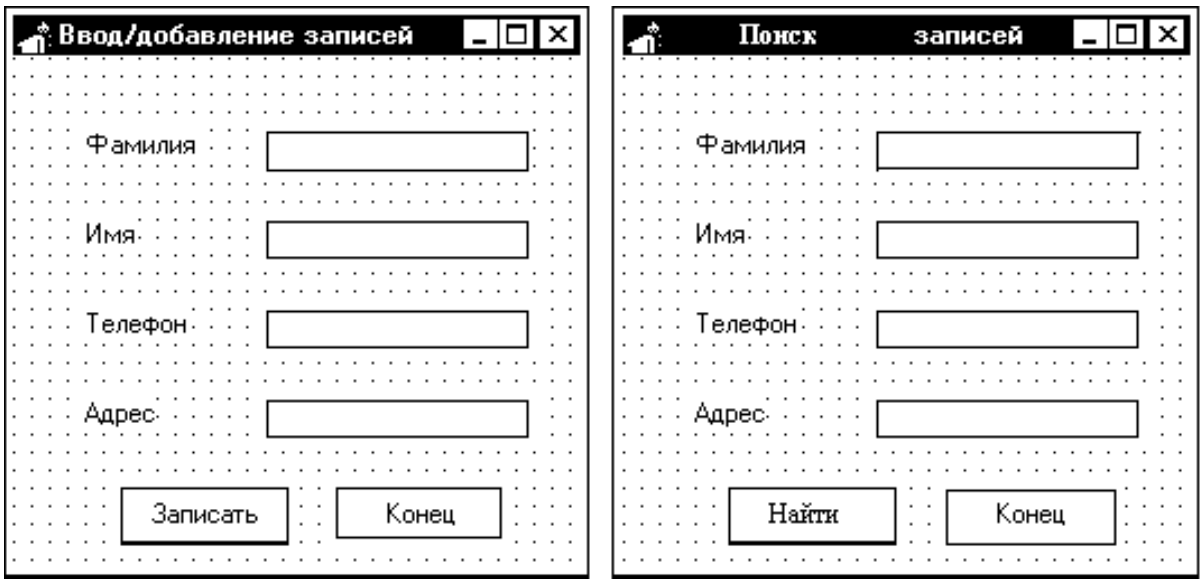


**Рисунок 8** – Дополнительные формы (Form1 и Form2).

Самостоятельно реализуйте данное приложение Windows. В случае возникновения трудностей обратитесь к приложению А, которое содержит возможный текст процедур предложенного проекта.

Выполните отладку приложения. Выполните анализ допущенных вами ошибок.

**Контрольные вопросы к части 2**

1. Какие визуальные компоненты следует использовать для построения заданного интерфейса?

2. Как добавить в приложение новые формы и модули?

3. Как обеспечить видимость описания структуры записи файла в других модулях?

4. Опишите структуру полученной программы. Чем структура приложения отличается от структуры консольной программы?

5. Как обеспечить размещение всех модулей и форм программы в одной папке?

## Требования к отчету

Отчет по лабораторной работе должен содержать:

1) текст задания 1;

2) рисунок формы интерфейса Калькулятора;

3) диаграмму состояний интерфейса Калькулятора;

4) диаграмму интерфейсных классов Калькулятора;

5) текст задания 2;

6) рисунки форм интерфейса Записной книжки;

7) диаграмму состояний интерфейса Записной книжки;

8) диаграмму интерфейсных классов Записной книжки;

9) при наличии перечень допущенных ошибок (2-3) и пояснения к сделанным исправлениям;

10) выводы по работе.

**Приложение А**

**Текст процедур для создания проекта «Записная книжка»**

I. Создание главной формы Form1 – «Записная книжка»

1. Определение типа записей, из которых состоит файл, и описание некоторых переменных:

```
type zap=record
    fam:string[22];          {Фамилия}
    name:string[22];              {Имя}
    fon:string[22];          {Телефон}
    adr:string[22];          {Адрес}
     end;
var
        f:file of zap;
        z:zap;
```

2. Обработчик события – нажать на кнопку «Ввод и добавление записей»

```
var size:integer;
begin
  AssignFile(f,'telefon.dat');
  {$I-} Reset(F); {$I+}
  if ioresult=0 then
    begin size := FileSize(f);
          seek(f,size);
    end
  else   rewrite(f);
  Form2.Show;
  Form2.edit1.setfocus;
end;
```

3. Обработчик события – нажать на кнопку «Поиск по фамилии».

```
begin
  AssignFile(f,'telefon.dat');
  reset(f);
  form3.show;
  form3.edit1.setfocus;
end;
```

4. Обработчик события – нажать на кнопку «Завершение работы».

```
begin
  Close;
end;
```

II. Создание Form2 – «Ввод и добавление записей»

1. Обработчик события – нажать на кнопку «Записать».

```
begin
 z.fam:=edit1.text;
 z.name:=edit2.text;
 z.fon:=edit3.text;
 z.adr:=edit4.text;
 edit1.clear;
 edit2.clear;
 edit3.clear;
 edit4.clear;
 write(f,z);
 edit1.setfocus;
end;
```

2. Обработчик события – нажать на кнопку «Конец».

```
begin
  closefile(f);
  self.hide;
end;
```

III. Создание Form2 – «Поиск записей».

1. Обработчик события – нажать на кнопку «Найти».

```
var c:integer;
   fam:string[22];
begin
  c:=1;
  fam:=edit1.text;
  while not eof(f) do
    begin
      read(f,z);
      if fam=z.fam then
        begin
```

```
            c:=0;
            edit2.text:=z.name;
            edit3.text:=z.fon;
            edit4.text:=z.adr;
            break;
          end;
      end;
    if c<>0 then
      begin
        edit3.text:='Нет данных';
      end;
    reset(f);
    end;
```

2. Обработчик события – нажать на поле ввода «Фамилия» (Edit1).

```
begin
  edit1.clear;
  edit2.clear;
  edit3.clear;
  edit4.clear;
end;
```

3. Обработчик события – нажать на кнопку «Конец».

```
begin
  closefile(f);
  self.hide;
end;
```