***Преподаватель***
***Мельникова Татьяна Федоровна***

***Тема: Использование в проекте: списков, переключателей, строковых таблиц.***

## Создание главного меню программы.



Создай новое приложение. Положим на форму один компонент MainMenu . Теперь посмотрим какие есть свойства у этого компонента:

• AutoHotkeys – будут ли создаваться автоматически клавиши быстрого вызова.

Если ты выберешь maAutomatic, то Delphi будет автоматически создавать клавиши. При maManual придётся это делать вручную.

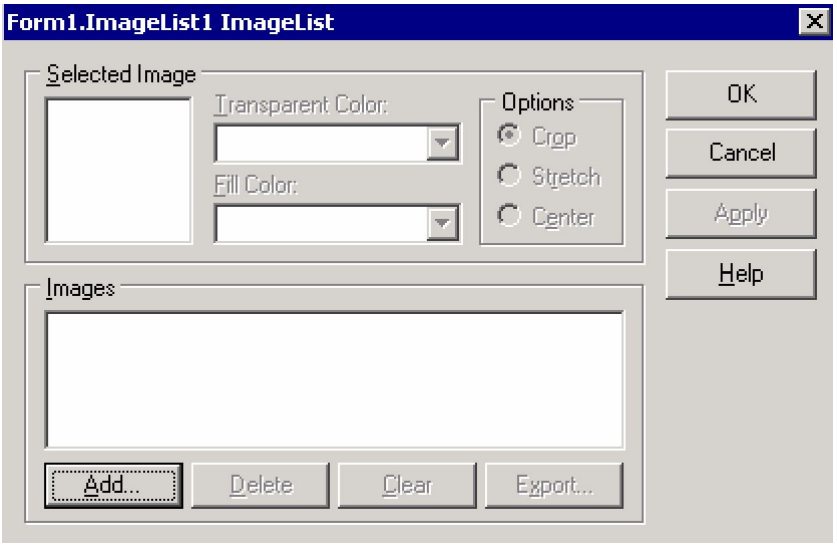
• AutoMegre – автоматическое слияние с дочерними окнами.

• Images – сюда можно подключать списки картинок, которые смогут  отображаться на пунктах меню.

• Items – здесь описываются пункты меню.

# Создание главного меню программы.

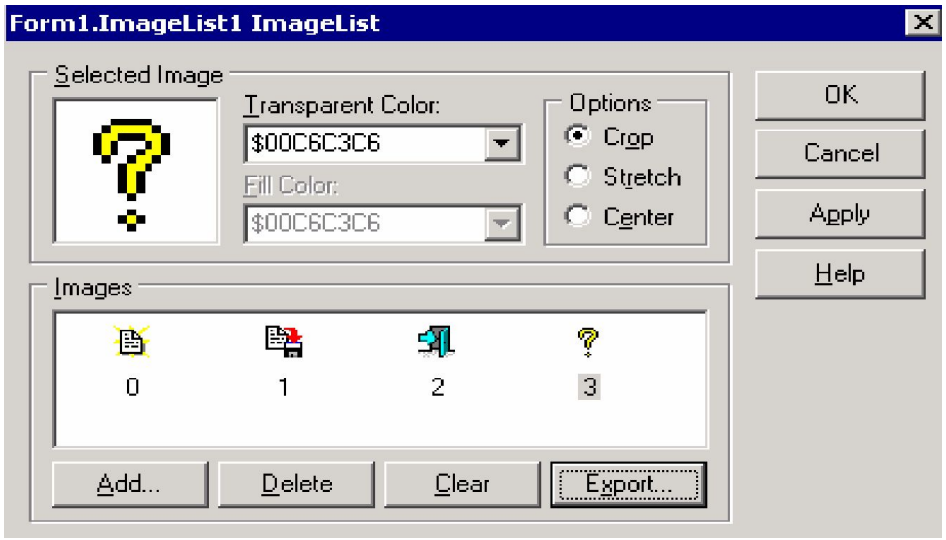
Подключение списка картинок. Брось на форму компонент ImageList с закладки Win32 . Теперь дважды щёлкни по нему и перед тобой откроется окно:




Здесь нажать кнопку Add чтобы добавить картинку. Откроется стандартное окно открытия файла. Открой какую-нибудь картинку, и она добавится в список. Желательно, чтобы размер - 16x16. Именно такие габариты используются по умолчанию.

# Создание главного меню программы.

Например,

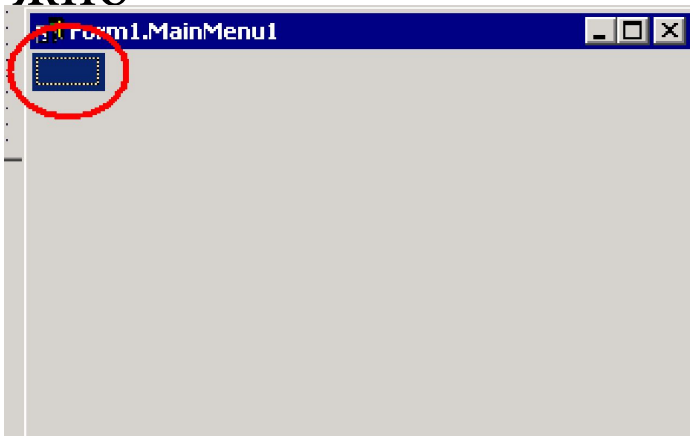



Теперь подключим наш список картинок к меню.
Выделим компонент MainMenu1 и у свойства Images, в выпадающем списке выбери пункт ImageList1.

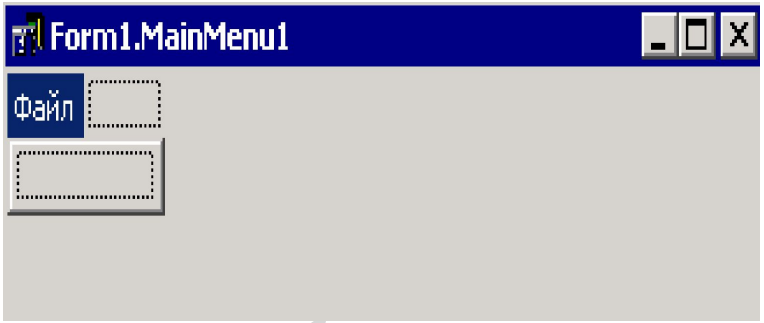
Теперь создадим само меню.

# Создание главного меню программы.

Создадим меню на форме. Для этого нужно дважды щёлкнуть по свойству Items и откроется редактор меню: Красным кругом выделен уже созданный пункт.



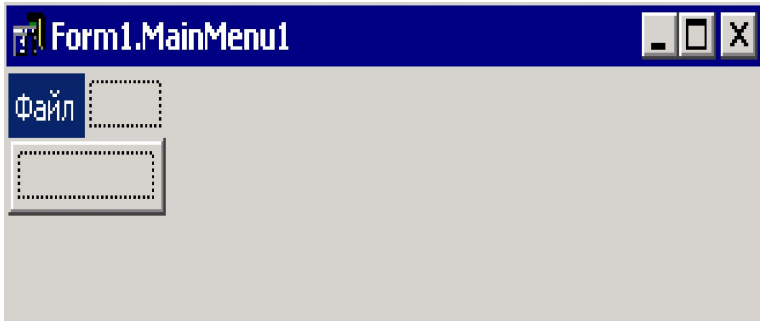

Этот же редактор можно вызвать, если дважды щёлкнуть по компоненту MainMenu1.
В инспекторе объектов набрать в свойстве Caption слово «Файл», кнопку Enter, будет создано меню «Файл»:



# Создание главного меню программы.

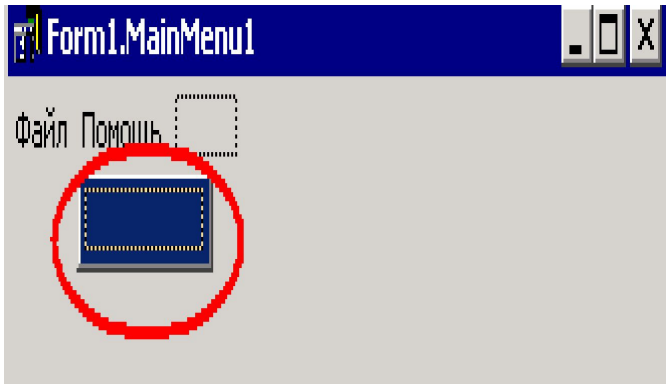
Этот же редактор можно вызвать, если дважды щёлкнуть по компоненту MainMenu1.

В инспекторе объектов набрать в свойстве Caption слово «Файл», кнопку Enter, будет создано меню «Файл»:



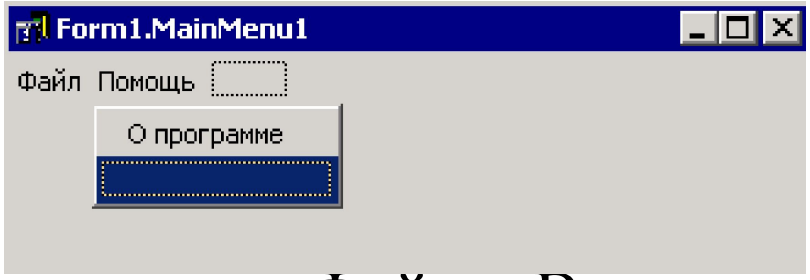

Создадим ещё и меню «Помощь».

Щёлкни справа от созданного меню (в рамочке обведённой пунктиром) и снова перейди в инспектор объектов в свойстве Caption слово «Помощь».

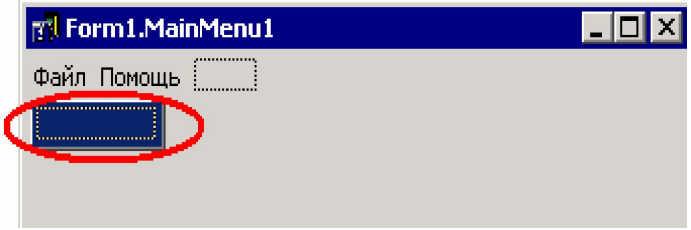


# Создание главного меню программы.

Здесь, в свойстве Caption мы введём слово «О программе», должно получится :




Теперь, таким же образом заполним меню «Файл». Выдели его. Теперь щёлкни в рамочке чуть ниже.

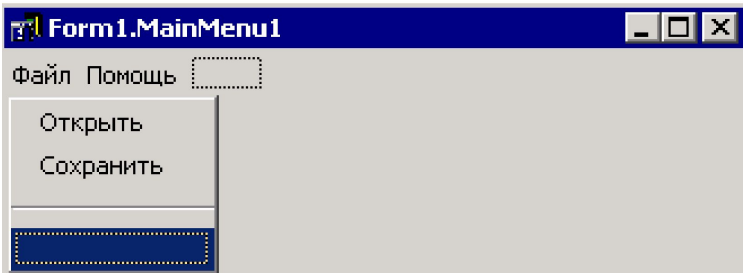



Напишем в свойстве Caption слово «Открыть»,нажмём Enter или перейдёшь на другой пункт меню в редакторе, создаться пункт «Открыть» и тут же немного ниже создаётся пустой пункт. Щёлкни по нему и введи в свойстве Caption слово «Сохранить».

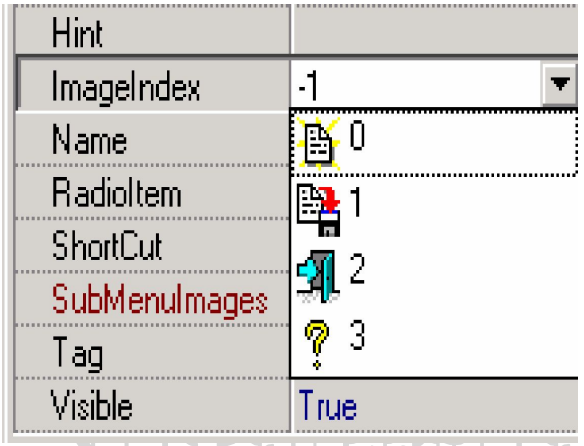
Теперь снова щёлкни на новом пункте меню и у него в

# Создание главного меню программы.

Теперь снова щёлкни на новом пункте меню и у него в свойстве Caption просто тире «-». Это заставит Delphi создать сепаратор:




Создадим последний пункт – «Выход». Теперь назначим каждому пункту меню картинки.
Выдели пункт «Открыть». Теперь в объектном инспекторе щёлкни по выпадающему списку свойства «ImageIndex».
Перед тобой откроется список всех картинок, которые мы подключили:



## Создание главного меню программы.


Выбери тот, что подходит, и картинка уже подключена.
Теперь создадим обработчик события нажатия по пункту меню. Для этого выбери в дизайнере меню пункт «Выход» и щёлкни по нему дважды или перейди на закладку Events и дважды щёлкни по событию OnClick.
Эти действия заставят Delphi создать обработчик события по нажатию меню. В этом обработчике напишем следующее:

```
procedure TForm1.N7Click(Sender: TObject);
begin
Close;
end;
```

## Создание главного меню программы.

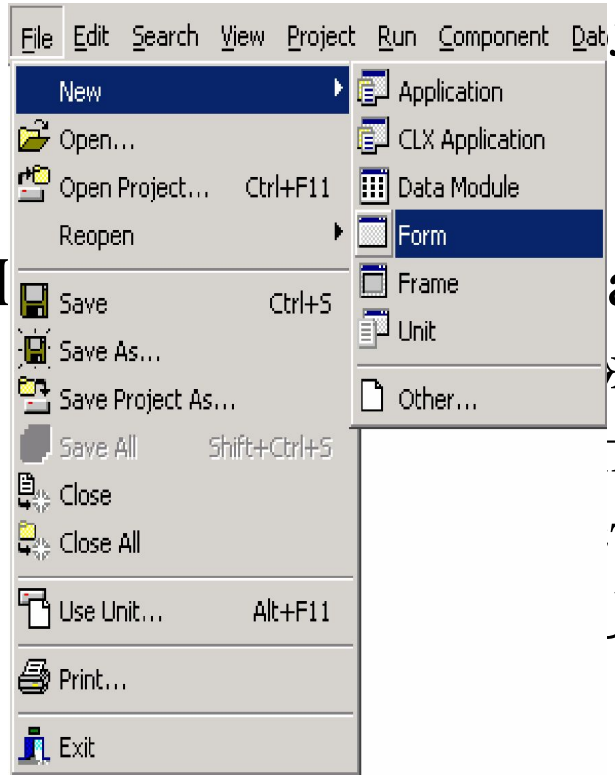


Здесь мы используем метод формы Close - этот метод закрывает форму.

Если мы закрываем главную форму, то закроется всё приложение.

# Создание дочерних окон

Создадим новую форму (дочернюю). Для этого, из меню File выбери пункт New, а затем выбери пункт Form, как показано на рисунке ниже.




лжен создать новую

чистую форму. Открой менеджер

(View ->Project

М а его

жимое и убедись, что

проекте Project1.exe

ть две формы:

Jnit2:

# Создание дочерних окон

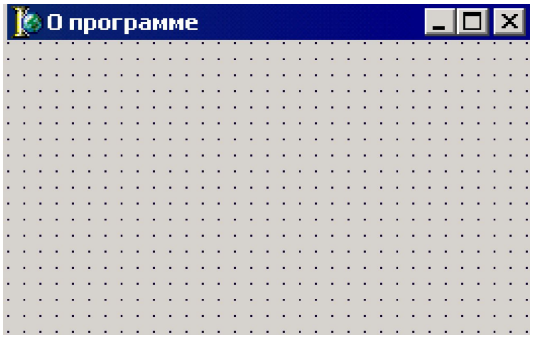


Сразу сохраним новую форму. Для этого при выделенной новой форме нажмем Ctrl+S. Появится окно для ввода имени формы.

Это окно у нас будет показывать информацию о программе, поэтому назовем его AboutUnit.pas.

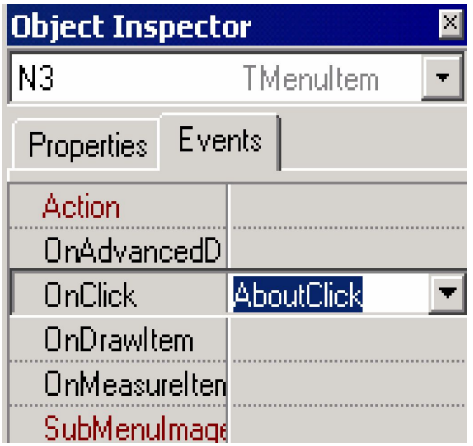
Модуль главной формы назовем - MainUnit.pas.

Сохранили. Теперь изменим заголовок формы на «О программе». Внутри формы можно написать любую информацию.



# Создание дочерних окон

Теперь нужно показать это окно.
Создадим обработчик события OnClick для пункта меню «О программе» у нашей главной формы.




```
procedure TForm1.AboutClick(Sender: TObject);
begin
AboutForm.ShowModal;
end;
```

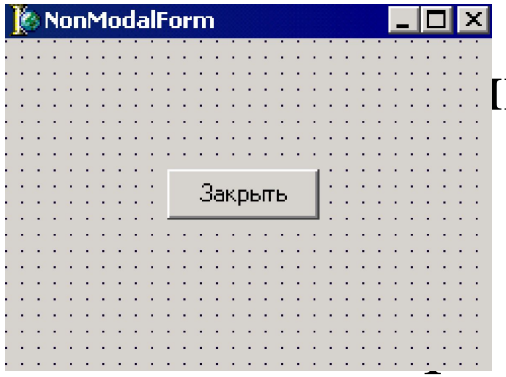
# Создание дочерних окон

В этом коде вызывается метод ShowModal окна AboutForm. Этот метод показывает форму в режиме Modal. В этом режиме окно получает полное управление, и пока оно не закроется, главная форма не будет работать.
Если сейчас откомпилировать код, то получим ошибку, что AboutForm не найдена.  Это потому, что эта форма описана в нашем модуле AboutUnit, а мы используем её в MainUnit. Чтобы MainUnit смог увидеть форму, описанную в AboutUnit, нужно её подключить. Для этого перейди в модуль MainUnit и запиши в раздел реализации

```
var
Form1: TForm1;
implementation
uses AboutUnit;
{$R *.dfm}
```

## Модальные и не модальные окна.

Создадим ещё одну форму. Сразу переименуем её свойство Name в NonModalForm. Положим на нее только одну кнопку, с помощью которой можно будет закрыть это окно:




им новую форму под именем NonModalUnit.pas.

вернёмся в главную форму и допишем в раздел uses имя модуля NonModalUnit:

**uses AboutUnit, NonModalUnit;**

**Модуль подключён, теперь можно его использовать.**

**Создадим обработчик события для пункта меню «Сохранить» и напишем в нём следующее:**

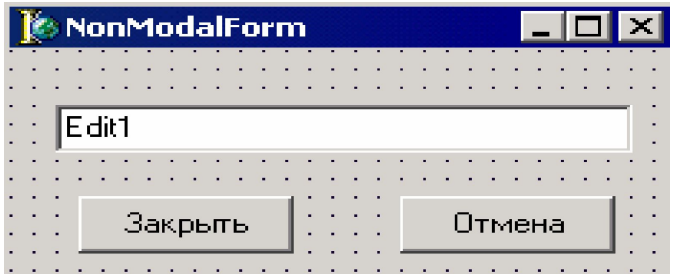
## Модальные и не модальные окна.

```
procedure TForm1.SaveClick(Sender: TObject);
begin
NonModalForm.Show;
end;
```

Здесь форма  NonModalForm как немодальное окно. Это значит, что если запустить программу и выбрать из меню пункт «Сохранить», то увидим окно новой формы и можно спокойно переключатся между главной формой и NonModalForm без каких-либо проблем.

# Обмен данными между формами.

мы создали немодальное окно для пункта меню «Сохранить», изменим это окно, добавив на него строку ввода:




Теперь посмотрим на свойство кнопки «Закрыть» - ModalResult. В этом свойстве мы можем задавать значение, возвращаемое при закрытии окна. Выберем здесь «mrOk». Теперь если мы покажем окно как модальное и потом закроем его кнопкой «Закрыть», то функция ShowModal вернёт нам значение mrOk.

## Обмен данными между формами.



Добавим на форму кнопку «Отмена», у которой свойство ModalResult установим в mrCancel. Нужно очистить обработчики событий OnClick, для кнопок. Когда мы указали в свойстве ModalResult возвращаемое значение, кнопка уже автоматически умеет закрывать окно и не нужно создавать для неё обработчик OnClick и в нём писать метод Close. В связи с этим, изменим обработчик события OnClick для пункта меню «Сохранить»:

```
procedure TForm1.SaveClick(Sender: TObject);
begin
if NonModalForm.ShowModal=mrOK then
Application.MessageBox(PChar(NonModalForm.Edit1.Text),
'Ты ввёл:', MB_OKCANCEL)
end;
```

## Обмен данными между формами.

В первой строке вызываем модальное окно и сразу проверяем возвращаемое значение. Если оно равно mrOK то выполняю следующее действие (if NonModalForm.ShowModal=mrOK then).
Вторая строка показывает стандартное окно диалога. MessageBox объекта Application. У этого метода три параметра:
1) Строка, которая будет показана внутри окна.
2) Строка заголовка окна.
3) Кнопки, которые будут на окне.
• MB_OK – кнопка «ОК».
• MB_OKCANCEL – кнопки «ОК» и «Отмена».

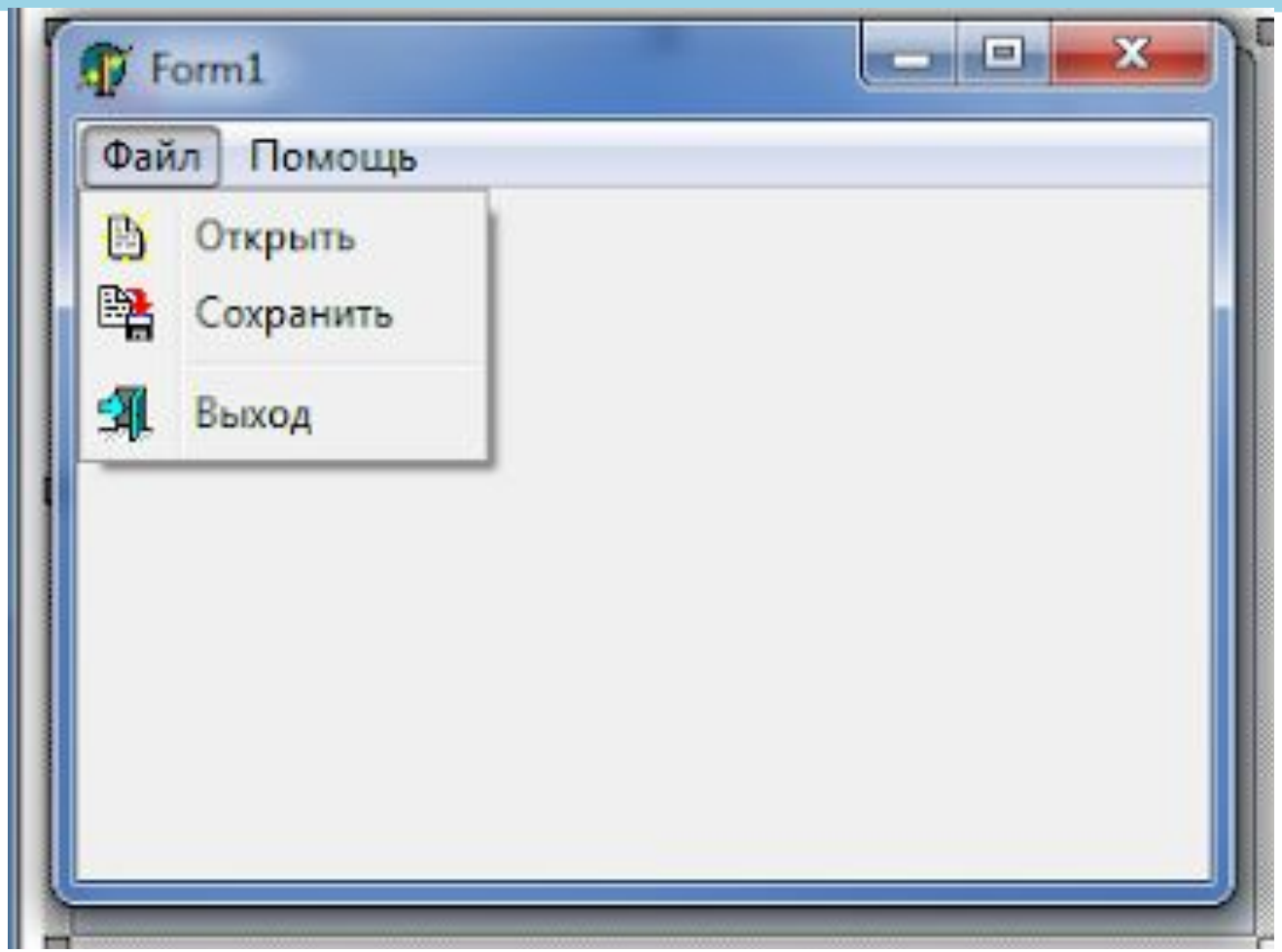
## Обмен данными между формами.

- MB_RETRYCANCEL – кнопки «Повторить» и «Отмена».
- MB_YESNO – кнопки «Да» и «Нет».
- MB_YESNOCANCEL – кнопки «Да», «Нет» и «Отмена».

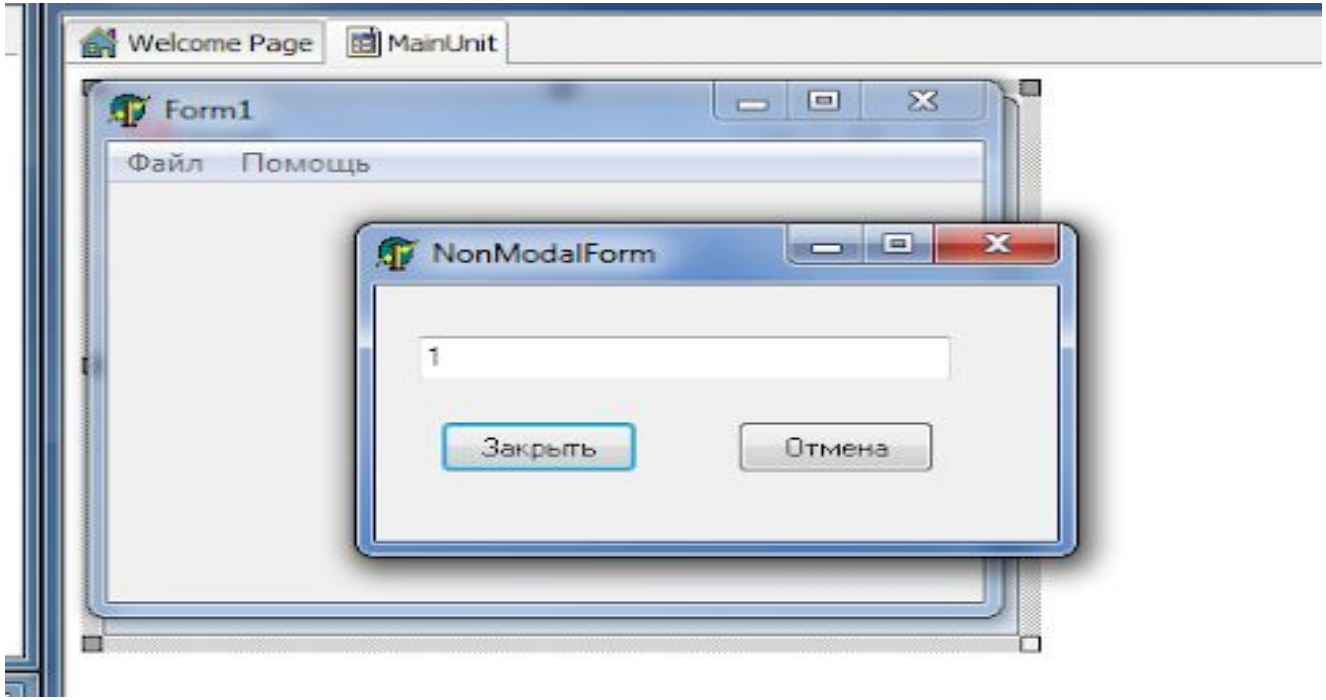
В качестве текста сообщения в окне выводится текст, введённый в строку ввода нашего модального окна (NonModalForm.Edit1.Text).

Теперь если пользователь нажмёт кнопку «Закрыть» в модальном окне, то появится окно с введённым текстом. Иначе ничего не произойдёт

окна.



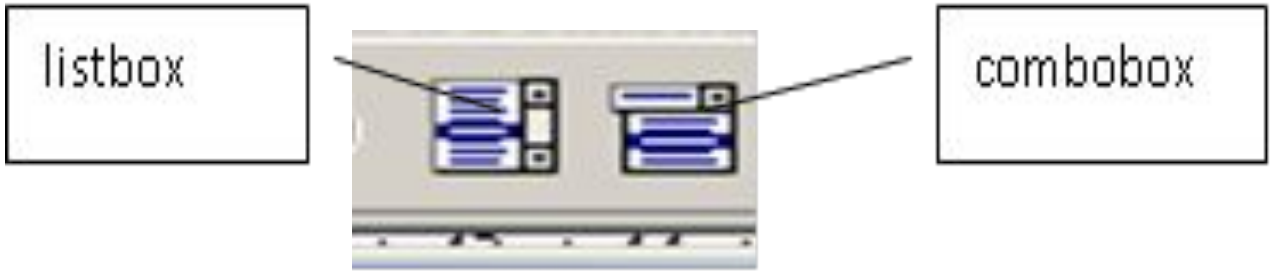
окна.


Welcome Page
MainUnit
Form1
Файл
Помощь
NonModalForm
1
Закрыть
Отмена

# Работа со списками

На вкладке **standart** панели компонент есть два компонента, соответствующих спискам.

**Listbox** – обычный список, **Combobox** – раскрывающийся список. Независимо от вида списка, принципы работы одинаковы. Рассмотрим основные свойства и режимы работы с ними.



## Работа со списками

Рассмотрим основные свойства и режимы работы с ними.

- **name** – имя, используемое в программе listbox1или combobox1 соответственно.
- **items** (элементы списка) – может использоваться как в программе, так и инспекторе объектов. Определяет значения элементов списка. При создании через инспектор объектов открывается дополнительное окно, в котором вводятся значения элементов, причем каждое значение с новой строки.
- **Itemindex** - номер выбранного элемента списка. Номер первого элемента списка равен 0, если не выбран ни один элемент номер равен –1. Может использоваться только в программе. Нумерация элементов списка начинается с нуля.

# Работа со списками

Например, Case listbox1.itemindex of

0: команда;

1: команда;

2: и т.д.;

end;

listbox1.items[n]:=<выражение>; //присваивает значение элементу с номером n.

listbox1.items.count //количество элементов в списке, может использоваться только в программе.

# Работа со списками

//добавляет элемент в список

**combobox1.items.add(‘строковое выражение’);**

// удаляет 5 по счету  строку

**combobox1.items.delete(4);**

**combobox1.items.clear;** //Очищает список

//загружает данные из текстового файла

**combobox1.items.loadfromfile(‘имя’);**

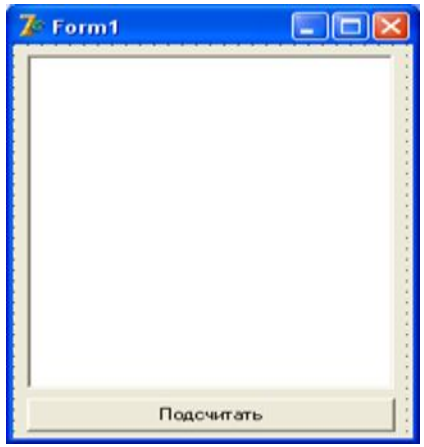
// сохраняет элементы списка в текстовом файле

**combobox1.items.savetofile(‘имя’);**

**combobox1.text** // значение элемента, который выбран в списке.

# пример

***Разделить число 10 на число от -3 до 3 включительно, и результат вывести в ListBox.***




```
procedure TForm1.Button1Click(Sender: TObject);
var    i, r : Integer;
Begin
  for i := -3 to 3 do begin
    if  i = 0 then begin
      ListBox1.Items.Add('На ноль делить нельзя!');
      Continue;
    end;
   r := Round(10/i);
   ListBox1.Items.Add('10/  '+ IntToStr(i) +'= '+ IntToStr(r));
  end;
end;
```

# Компонент – memo


Вкладка **standart**. Основные свойства и методы.

- **Name** – имя, используемое в программе (**memo**).
- **Text** – все содержимое компонента независимо от количества заполненных строк. Может использоваться только в программе.
- **Lines** – отдельные строки компонента. Свойство можно задавать через инспектор объектов или через программу. Все строки нумеруются, начиная с нуля.

**Memo1.lines[3]:=<выражение>;** //присваивает значение четвертому по счету элементу.

**Memo1.lines.clear;** //очищает весь компонент.

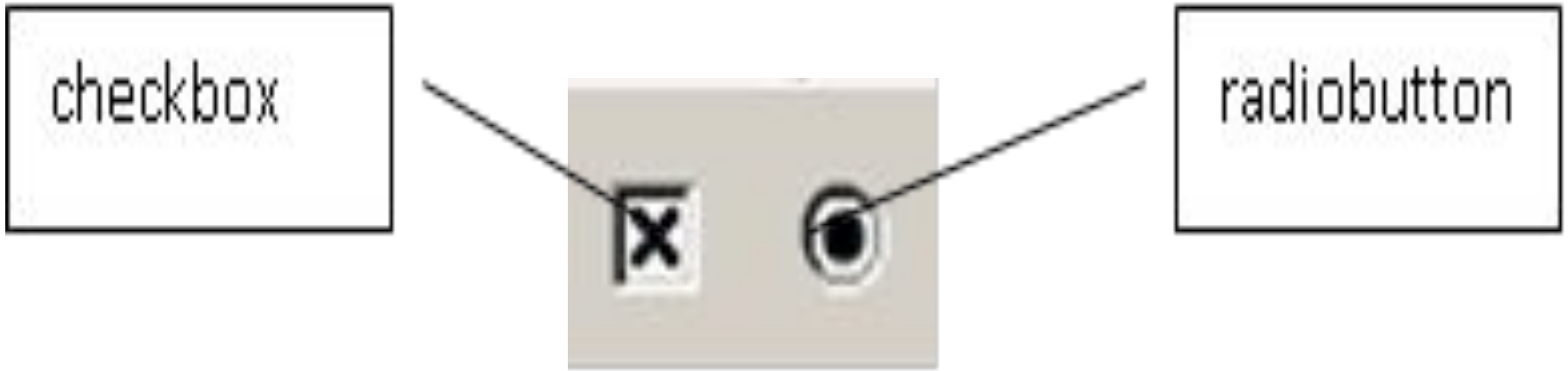
**Memo1.lines.add('строка');** //добавляет новую строку в компонент.

**Memo1.lines.loadfromfile('имя');** // загружает данные из файла.

**Memo1.lines.savetofile('имя');** //сохраняет данные из компонента в текстовом файле.

# Работа с переключателями

На вкладке **standart** есть два компонента, соответствующие переключателям.



# Основные свойства компонента -флажок (Checkbox)

| Свойство | Обозначение | Значение |
|---|---|---|
| Имя компонента. Используется для доступа к свойствам компонента. | Name | Checkbox1 |
| Текст, поясняющий назначение переключателя. | Caption | Выводить протокол |
| Определяет состояние, внешний вид переключателя. Если переключатель выбран (в квадратике, изображающем переключатель, находится «галочка»), то checked=true.Если переключатель не выбран, то checked=false. | Checked | True |

# Основные свойства компонента -флажок (checkbox)

| Свойство | Обозначение | Значение |
|---|---|---|
| Определяет состояние переключателя. В отличие от свойства **checked** позволяет различать выбранное, невыбранное и промежуточное состояния. Состояние определяют константы: cbchecked (выбран), cbgrayed (серый, неопределенное состояние) и cbunchecked (не выбран). | State | |
| Определяет, может ли переключатель быть в промежуточном, неопределенном состоянии. Если allowgrayed=false, то переключатель может быть только выбранным или невыбранным. Если allowgrayed=true, то промежуточное состояние допустимо. | Allowgrayed | False |

Например, при включенном переключатели, вывести максимальное значение элемента массива.

```
If checkbox1.checked then
   label2.caption:=’ максимальный элемент ‘ +inttostr(max);
```

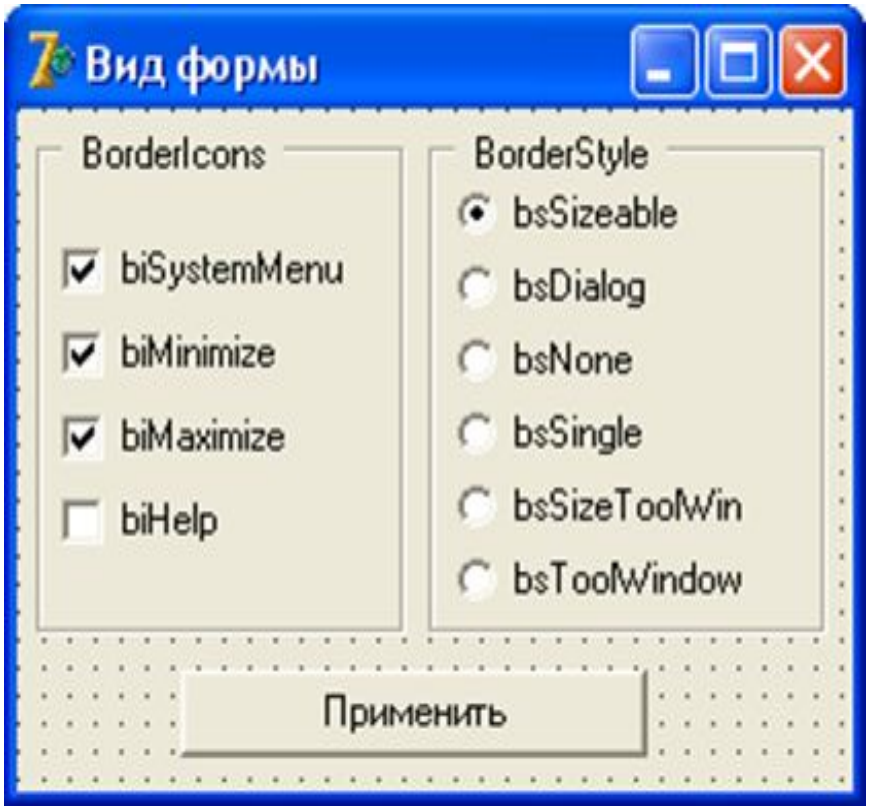
Можно вместо одного переключателя, использовать группу переключателей – **Radiogroup**.

# Основные свойства компонента Radiogroup



| Свойство | Обозначение |
|---|---|
| Имя компонента. Используется в программе для доступа к свойствам компонента. | **Name** |
| Текст, поясняющий назначение переключателя. | **Caption** |
| Количество колонок, на которые разбит вывод переключателей. | **columns** |
| Перечисление значений, которые выбираем. Нумерация пунктов начинается с нуля. | **items** |
| Какой пункт выбран | **itemindex** |

# Форма проекта



## Обработчик события



```
procedure TfMain.Button1Click(Sender: TObject);
begin      //обрабатываем компонент BorderIcons
  if CheckBox1.Checked then fMain.BorderIcons :=
fMain.BorderIcons + [biSystemMenu]
  else fMain.BorderIcons := fMain.BorderIcons - [biSystemMenu];

  if CheckBox2.Checked then fMain.BorderIcons :=
fMain.BorderIcons + [biMinimize]
  else fMain.BorderIcons := fMain.BorderIcons - [biMinimize];

  if CheckBox3.Checked then fMain.BorderIcons :=
fMain.BorderIcons + [biMaximize]
  else fMain.BorderIcons := fMain.BorderIcons - [biMaximize];
```

## продолжение

```
if CheckBox4.Checked then fMain.BorderIcons := fMain.BorderIcons
+ [biHelp]
  else fMain.BorderIcons := fMain.BorderIcons - [biHelp];
  //обрабатываем компонент BorderStyle
  case RadioGroup1.ItemIndex of
    0 : fMain.BorderStyle := bsSizeable;
    1 : fMain.BorderStyle := bsDialog;
    2 : fMain.BorderStyle := bsNone;
    3 : fMain.BorderStyle := bsSingle;
    4 : fMain.BorderStyle := bsSizeToolWin;
    5 : fMain.BorderStyle := bsToolWindow;
  end;
end;
```

# Работа со строковыми таблицами


Элемент **StringGrid** (строковая таблица) находится на дополнительной (Additional) странице палитры компонентов.

Этот компонент представляет собой ***двумерный строковый массив***.

# Основные свойства объекта:StringGrid

| Свойство | Назначение |
|---|---|
| **ColCount** | Определяет число колонок таблицы |
| **RowCount** | Определяет число строк таблицы |
| **Col** | Колонка активной ячейки |
| **Row** | Строка активной ячейки |
| **ColWidths** | Изменяет ширину текущей колонки |
| **RowHidghts** | Изменяет высоту текущей строки |
| **DefaultColWidths** | Определяет ширину всех колонок |
| **DefaultRowHidghts** | Определяет высоту всех строк |
| **DefaultDrawing** | Определяет, будет ли автоматически прорисовываться рамка (TRUE) |
| **EditorMode** | Определяет, возможно ли редактирование содержимого (TRUE) |

# Свойства StringGrid

| Свойство | Назначение | | |
|---|---|---|---|
| Options | Определяет внешний вид таблицы | | |
| | goFixedVerLine | True | Отображаются вертикальные линии между колонками в фиксированных блоках |
| | goFixedHorLine | True | То же горизонтальные |
| | goVerLine | True | Отображаются вертикальные линии между колонками |
| | goHorLine | True | То же горизонтальные |
| | goRangeSelect | True | Можно выделить блок ячеек (если нет goEditing) |
| | goDrawFocusSelected | True | Активная ячейка не меняет цвет |
| | goRowSizing | True | Размеры строк можно менять при работе приложения (за исключением фиксированных) |

# StringGrid

| Свойство | Назначение | | |
|---|---|---|---|
| Options | Определяет внешний вид таблицы | | |
| | goColSizing | True | То же для колонок |
| | goRowMoving | True | Можно переместить строку на новое место с помощью мыши |
| | goColMoving | True | То же для колонок |
| | goEditing | True | Можно редактировать текст (но не выделять блок!) |
| | goTabs | True | Перемещаться по ячейкам можно через «TAB» |
| | goRowSelect | True | Можно выделять отдельные строки |
| | goAlwaysShowEditor | True | Таблица автоматически редактируется (иначе f2 или Enter) |
| | goThumboTracking | True | Видимая часть таблицы прокручивается синхронно бегунку |

# Свойства StringGrid

| | |
|---|---|
| FixedColor | Определяет цвет фиксированных элементов таблицы |
| FixedCols | Определяет число фиксированных колонок |
| FixedRows | То же для строк |
| TopRow | Определяет, какая строка первая сверху (если нет фиксированных строк) |
| LeftCol | То же для колонок |
| GridHeight | Высота таблицы в пикселах |
| GridWidth | То же для ширины |
| GridLineWidth | Определяет толщину разделительной линии |
| Selection | Описывает прямоугольник, содержащий выделенный блок таблицы |
| VisibleColCount | Число видимых колонок (кроме фиксированных) |
| VisibleRowCount | То же для строк |
| Objects | Массив объектов, соответствующий ячейкам таблицы StringGrid.Objects[10,3]:=333 (запись числа в ячейку) |
| Cells | Доступ к тексту из заданной ячейки. Строки и столбцы таблицы нумеруются с нуля. Ячейка **cells[j,i]** находится на пересечении i-й строки и j-го столбца. |

# Примеры программ:



Добавление новой строки вниз таблицы.

```
Begin
  Sg.rowcount:=sg.rowcount+1;
  Sg.rows[sg.rowcount-1].clear;  //очистка добавленной строки
End;
```

## пример

Удаление текущей строки (на которой стоит курсор) из таблицы. В результате выполнения процедуры все строки, начиная с текущей сдвигаются вверх, последняя строка удаляется. Удаление выполняется, если в таблице более двух строк.

```
var n: integer;
Begin   if sg.rowcount=2 then exit;
 for n:=sg.row to sg.rowcount-2 do
sg.rows[n]:=sg.rows[n+1];  sg.rowcount:=sg.rowcount-1;
end;
```

sg.row – текущая строка.

sg.rows[n] – строка с номером n.

```
Listbox1.items.assign(sg.cols[3]); //запись данных из
```

столбца с номером 3 в список.

## пример

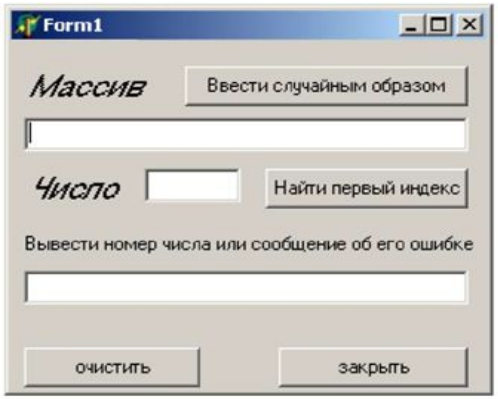
Фрагмент процедуры, которая суммирует значения третьего столбца таблицы. Sg – значение свойства name таблицы Stringgrid.

```
Var
   i,s:integer;
begin
  s:=0;
for i:=1 to sg.rowcount-1 do
   s:=s+strtoint(sg.cells[3,i]);
end;
```

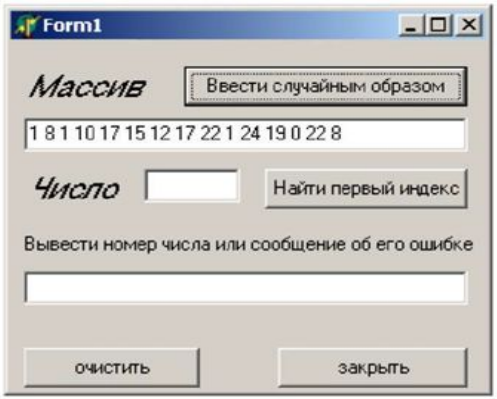
# Нахождение индекса в массиве случайных чисел

Вид проекта по действиям:

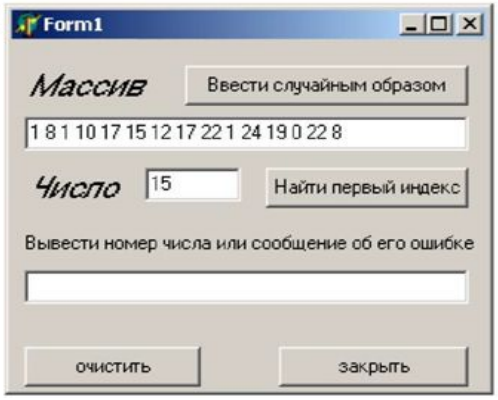
1)



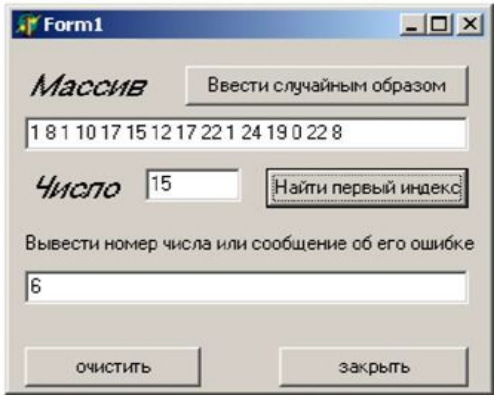

2)




3)




4)



# Нахождение индекса в массиве случайных чисел


Описание плана разработки программы

1. Открыть новый проект.

2. Разместить на форме экземпляры компонентов: Button, Edit, Label.

3. Выполнить следующие действия:

| Выделенный объект | Вкладка окна Object Inspector | Имя свойства/ имя события | Действие |
|---|---|---|---|
| **Forml** | Properties | Caption | Установка имени формы "Новый проект" |
| | Events | OnCreate | Очистить значения свойств Text текстовых полей |
| **Buttonl** | Properties | Caption | Введите название "Очистить" |
| | Events | Onclick | Очистить значения свойств Text текстовых полей |

## пример

| Выделенный объект | Вкладка окна Object Inspector | Имя свойства/ имя события | Действие |
|---|---|---|---|
| **Button2** | Properties | Caption | Введите название "Закрыть" |
| | Events | Onclick | Обработка события закрытия формы |
| **Button3** | Properties | Caption | Введите название "Найти первый индекс" |
| | Events | Onclick | Обработка события нахождения индекса введенного числа |
| **Button4** | Properties | Caption | Введите название "Ввести случайным образом" |
| | Events | Onclick | Ввод массива случайным образом |
| **Editl** | Properties | Caption | Очистить значение свойства Text |
| **Edit2** | Properties | Caption | Очистить значение свойства Text |
| **Edit3** | Properties | Caption | Очистить значение свойства Text |

## пример

4. Опишите переменные ik, k, i : integer;

s : string; a : array [1..15] of integer.

Листинг программы

```
procedure Tform1.FormCreate(Sender: Tobject);
begin Edit1.Text := ''; Edit2.Text := ''; Edit3.Text := ''; end;
procedure Tform1.Button1Click(Sender: Tobject);
 begin Edit1.Text := ''; Edit2.Text := ''; Edit3.Text := '';
end;
procedure Tform1.Button2Click(Sender: Tobject);
 begin close; end;
```

## пример

```
procedure Tform1.Button3Click(Sender: Tobject);
 begin  k := StrToInt (Edit2.Text);
 For i:= 1 to 15 do
 if  k = a[i] then ik := i;
 if ik = 0 then Edit3.Text := 'number ubsent'
              else Edit3.Text := IntToStr (ik);
end;
procedure Tform1.Button4Click(Sender: Tobject);
Begin  randomize; s := '';
 For i := 1 to 15 do begin  a[i] := random (26);
 s := concat (s, IntToStr (a[i]), #32); end; Edit1.Text := s;
 end; end.
```

# Спасибо за внимание!