**Цикл презентаций «ООП на Delphi» посвящен объектно – ориентированному программированию с использованием одной из самых распространенных систем быстрой разработки приложений – Delphi**

**Используя данный учебный курс, можно самостоятельно овладеть основами объектно – ориентированного программирования на Delphi. Для расширения Ваших знаний к курсу приложен ряд учебных пособий и справочников по Delphi**

## Цикл содержит 13 презентаций:

ООП на Delphi – 1: Знакомство с системой программирования Borland Delphi. Объекты (компоненты) и их свойства и методы

ООП на Delphi – 2: Первая программа на Delphi, сохранение и компиляция

ООП на Delphi – 3: Программное изменение свойств объектов

**ООП на Delphi – 4: Условия в Delphi. Создание простого теста**

ООП на Delphi – 5: Элементы ввода и вывода информации

ООП на Delphi – 6: Заставка программы и элемент таймер

ООП на Delphi – 7: Программируем свою игрушку

ООП на Delphi – 8: Меню программы, диалоги

ООП на Delphi – 9: Создаем свой текстовый редактор

ООП на Delphi – 10: Базы данных на Delphi

ООП на Delphi – 11: Калькулятор на Delphi

ООП на Delphi – 12: Создаем обучающе - тестирующую систему

ООП на Delphi – 13: Графика на Delphi

Delphi использует язык программирования Объект Паскаль, поэтому лучше сначала изучить обычный Паскаль и поработать в ТурбоПаскале, а затем и переходить к Delphi – перейти будет очень просто, т.к синтаксис языка остается неизменным.

Изучение ООП на Delphi желательно проводить в старших профильных классах – количество часов, отводимое на информатику там вполне достаточно для освоения основ ООП на Delphi

# Создание интерфейса оконного приложения

Borland®

# DELPHI - 4

## На этом уроке:

**Мы научимся создавать и проверять условия, пользоваться компонентами Radio Button и Checkbox и составим простейшую тестирующую программу**

## Вопросы:

1. Компоненты Delphi **Radio Button и Checkbox**
2. Создание простейшей тестирующей программы

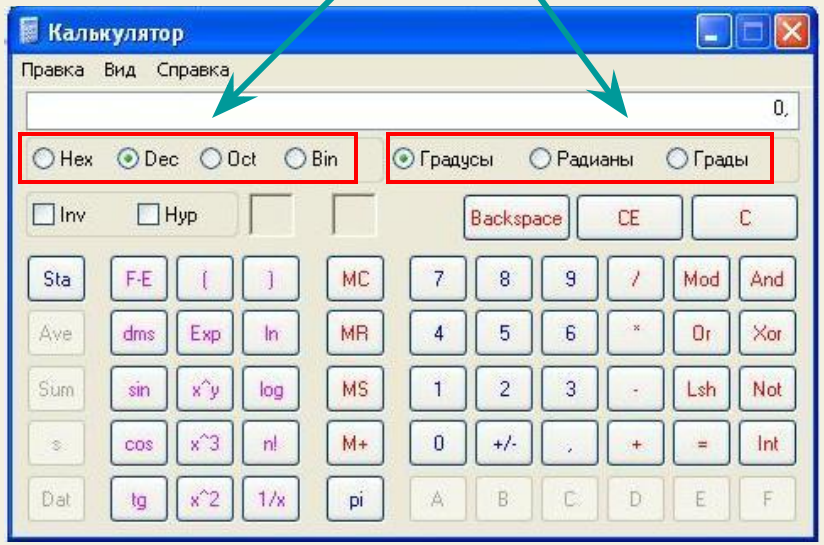
Borland®

# 1. Компоненты Delphi
# Radio Button и Checkbox

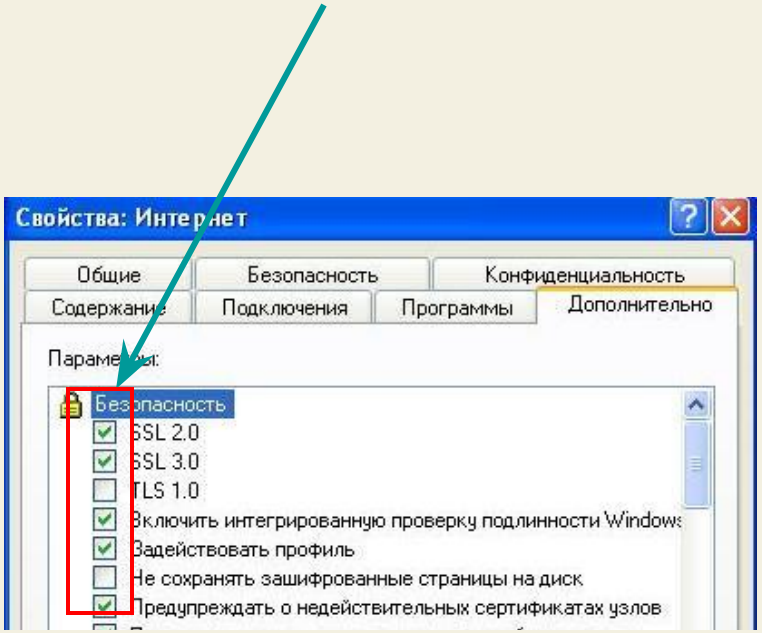
Компоненты Radio Button и Check Box позволяют сделать выбор из нескольких условий, причем Radio Button позволяет сделать единственный выбор из многих условий, а Check Box – множественный выбор

С этими компонентами мы постоянно встречаемся, работая в операционной системе Windows:

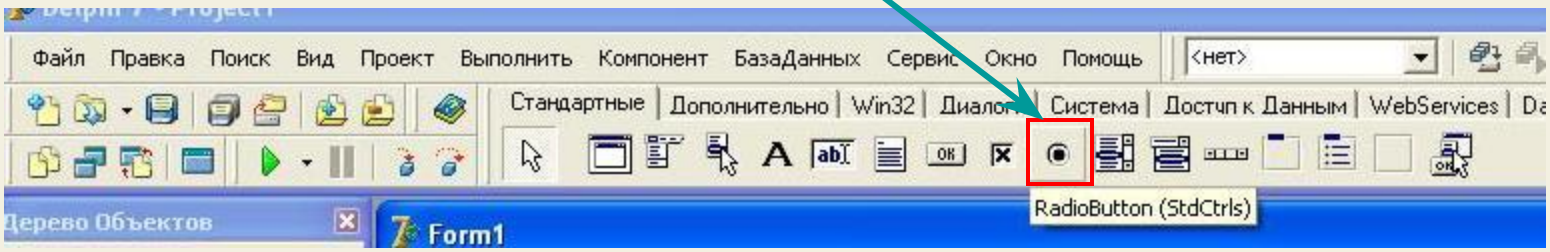
Radio Button - ы для выбора одной из нескольких систем исчисления, а также одной из мер измерения угла

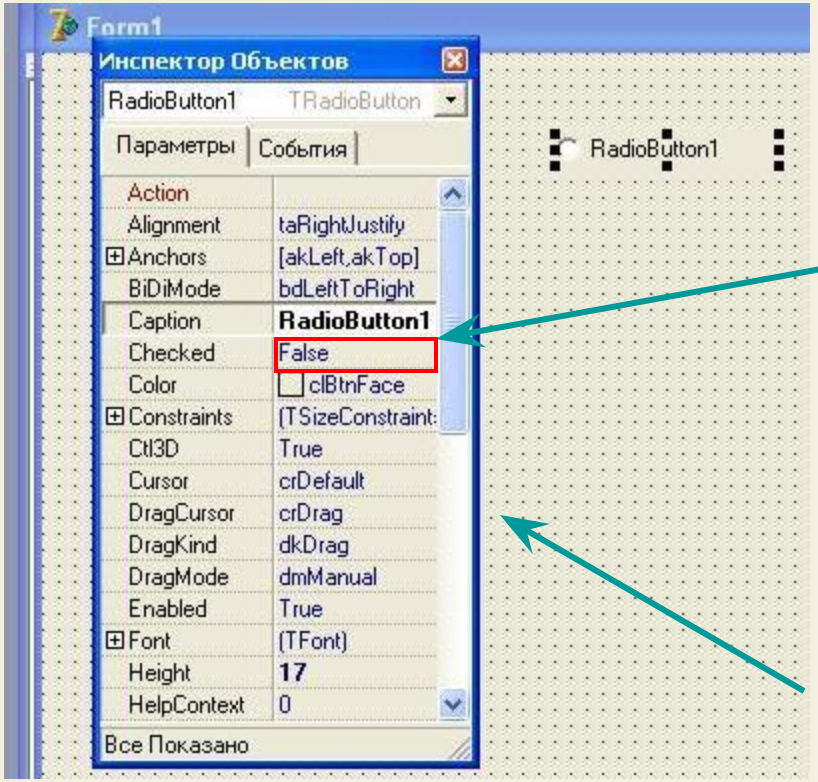



Check Box - ы для выбора сразу нескольких условий из многих



# Компонент Radio Button




Radio Button может иметь только 2 значения:

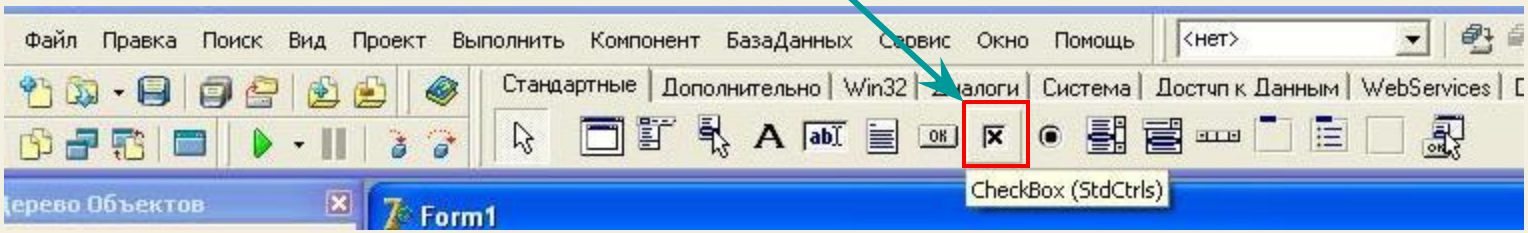
False (ложно)

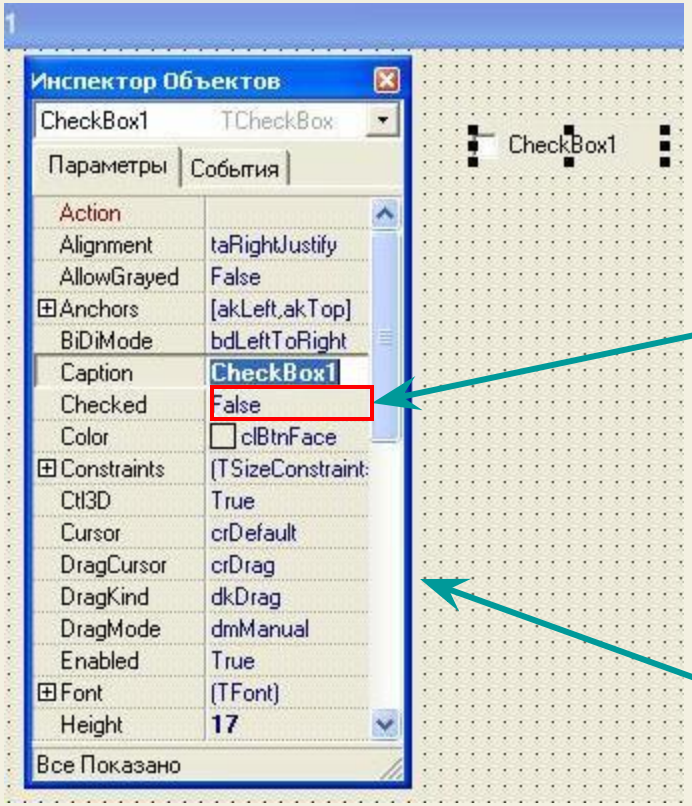
True (истинно)

Кроме этого у радиокнопки есть множество других свойств: можно изменить надпись на кнопке, ее цвет, положение, доступность ...

Посмотрите внимательно набор свойств компонента Radio Button в инспекторе объектов

# Компонент Check Box




Check Box

может иметь только 2 значения:

False (ложно)

True (истинно)

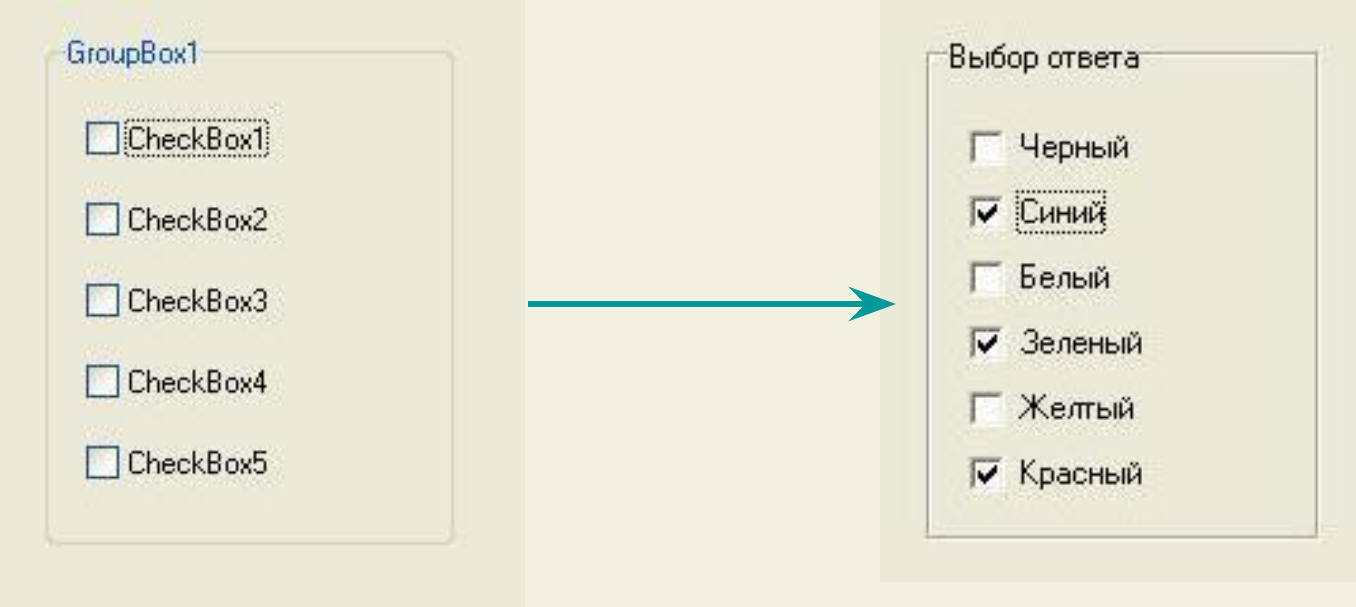
Кроме этого у Check Box-а есть множество других свойств: можно изменить надпись, цвет, положение, доступность ...

Посмотрите внимательно набор свойств компонента Check Box в инспекторе объектов

Для объединения Radio Button и Check Box при размещении на форме удобно и красиво применять компоненты Group Box и Radio Group

**Использование Group Box:**

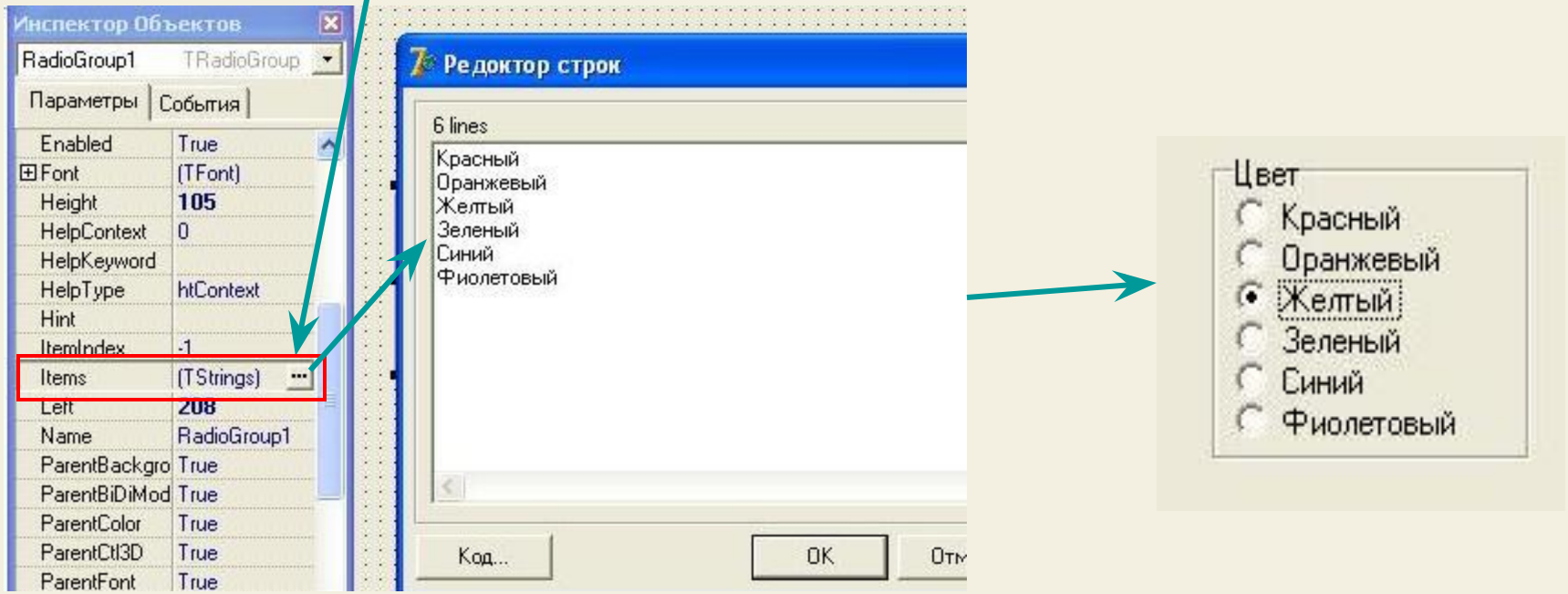
- Помещаем компонент Group Box на форму
- Помещаем на него необходимое количество компонент (например Check Box, а можно и других)
- В свойстве Caption изменяем надписи




В результате мы получаем группу независимых переключателей, где значение True (или False) может иметь сразу несколько переключателей – т.е. мы можем осуществить **множественный выбор**

## Использование Radio Group:

- Помещаем компонент Radio Group на форму
- Раскрываем свойство Items этого компонента и заполняем строки для будущих зависимых переключателей, нажимая Enter после каждой строки
- В свойстве Caption компонента изменяем заголовок




В результате мы получаем группу зависимых переключателей, где значение True может иметь только один из всех переключателей, т.е. мы можем осуществить **единственный выбор**

Borland®

# 2. Создание простого теста

**А сейчас давайте попробуем, используя компоненты Radio Button и Check Box создать простой тест с минимальными требованиями к программе (обойдемся пока приложением из одной формы, без регистрации, отдельных файлов тестов, картинок и пр.) – сделаем это по шагам**

**ШАГ 1**

Конечно, сначала нужно составить тест на бумаге. Для примера возьмем тест из 5 вопросов, причем первые три вопроса с единственным выбором, а последние два – с множественным:

## с единственным выбором

| | | |
|---|---|---|
| 1 | Верно ли, что вся информация в компьютере хранится в двоичном коде | Только числовая |
| | | Только числовая и текстовая |
| | | Любая информация |
| | | Вся информация за исключением видео |
| 2 | Наименьшая единица информации называется | байт |
| | | бит |
| | | бод |
| | | бит/с |
| 3 | Элементной базой первых компьютеров являлись | микросхемы |
| | | дискретные полупроводниковые элементы |
| | | радиолампы |
| | | транзисторы |

## с множественным выбором

| | | |
|---|---|---|
| 4 | Назовите устройства вывода информации | Монитор |
| | | мышь |
| | | клавиатура |
| | | принтер |
| 5 | Назовите системы программирования | Delphi |
| | | Visual Basic |
| | | Microsoft |
| | | Turbo Pascal |

**ШАГ 2**

Сейчас давайте определим требования к программе :

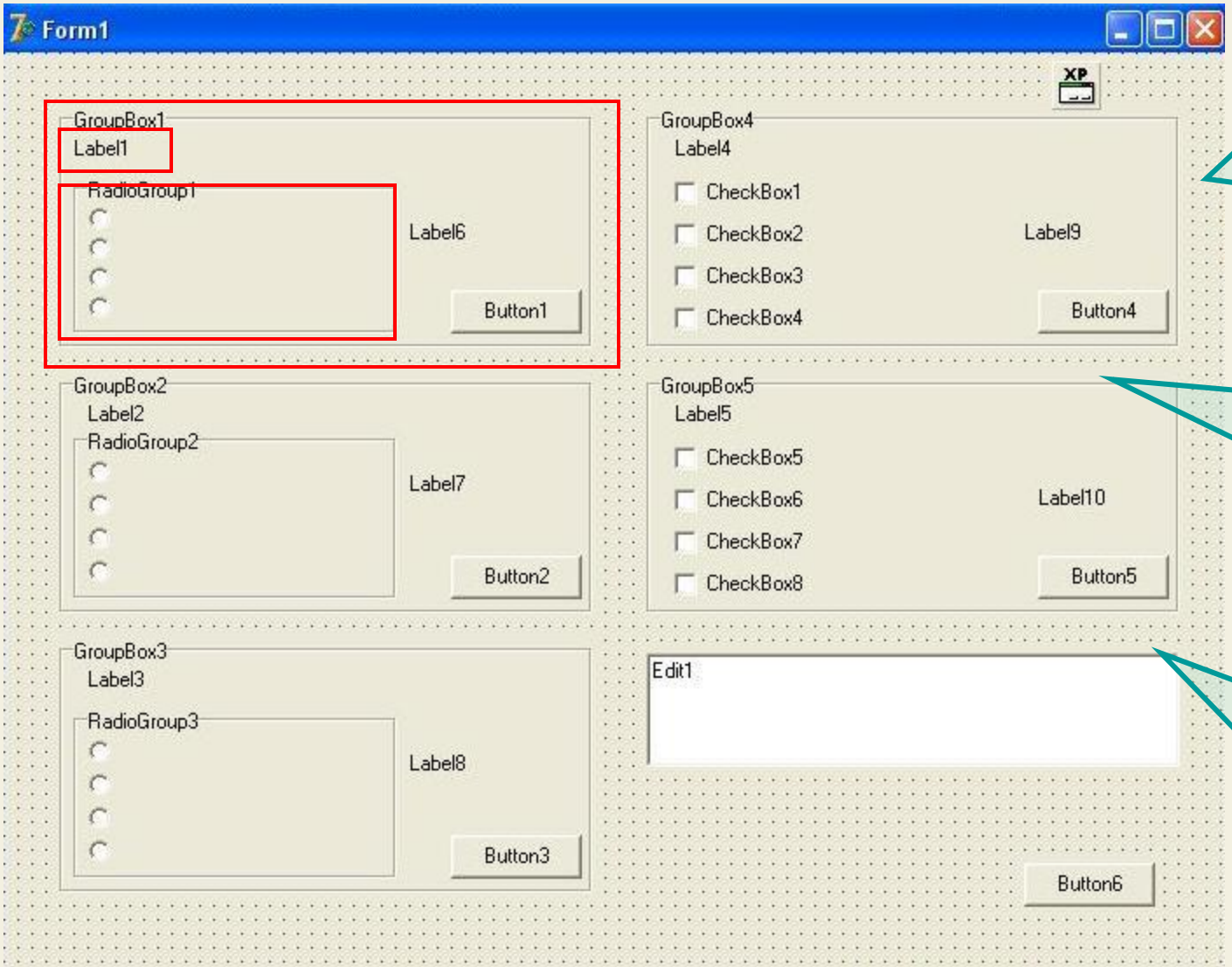
Для первой тестирующей программы мы ограничимся только следующими функциями:

- Подсчет числа верных ответов
- Подсчет % верных ответов
- Вывод результата после окончания теста
- Блокировка возврата к предыдущим вопросам
- Использование 4 вариантов выбора ответа
- Использование заданий как с единственным, так и множественным выбором ответа
- После каждого ответа программа должна оповещать нас о правильности нашего выбора

И на этом функциональность программы пока ограничим

**ШАГ 3**

Следующим шагом будет разработка внешнего вида приложения и определение компонент, которых мы будем использовать для ввода и вывода информации




Размещаем для каждого вопроса свой компонент Group Box

Размещаем Label для вывода текста вопроса

Размещаем Radio Group на 4 варианта для выбора ответов

**ШАГ 3**

Следующим шагом будет разработка внешнего вида приложения и определение компонент, которых мы будем использовать для ввода и вывода информации

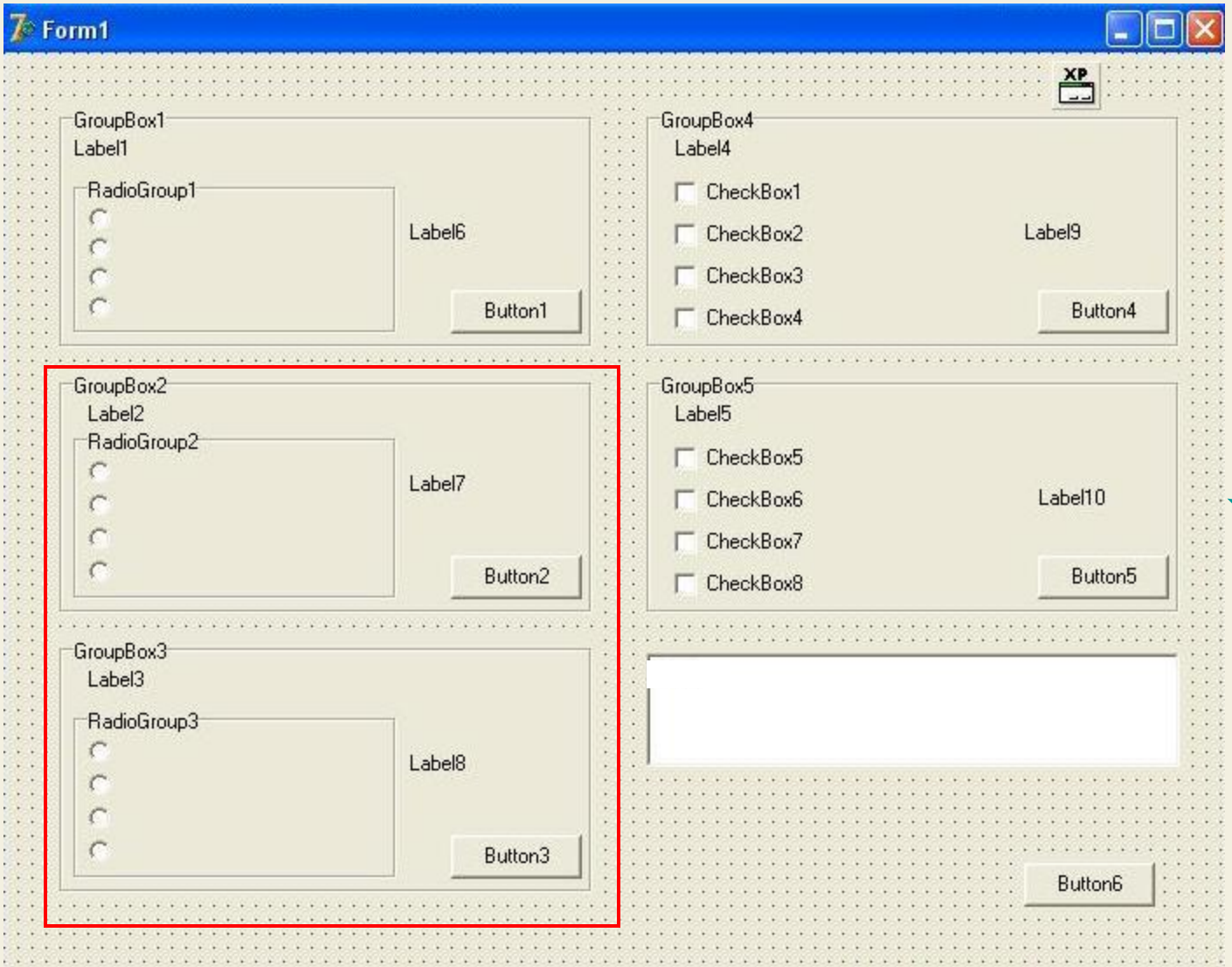



Здесь помещаем Label для визуального отображения (верно/ неверно мы ответили)

Ставим кнопку для проверки выбранного ответа

**ШАГ 3**

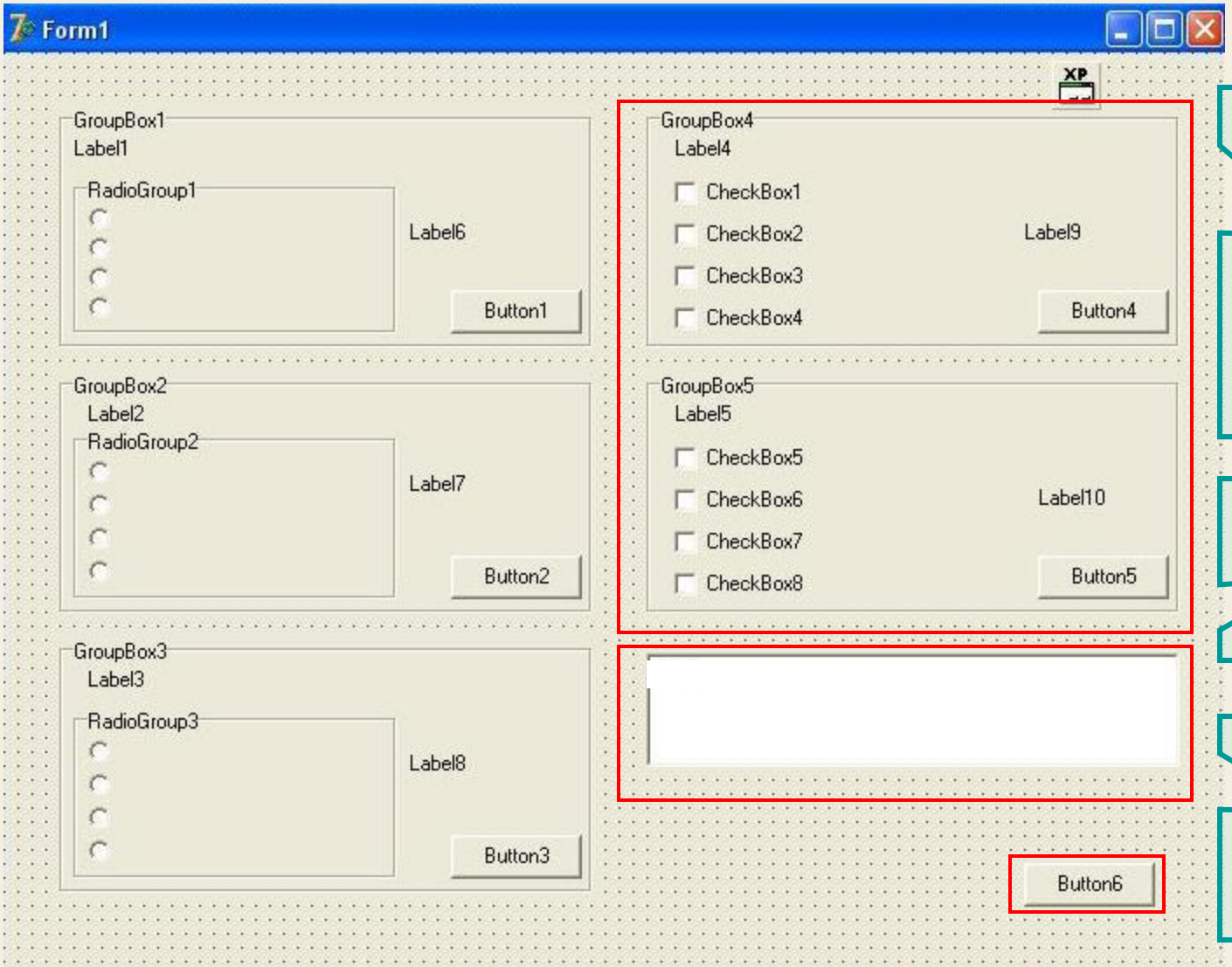
Следующим шагом будет разработка внешнего вида приложения и определение компонент, которых мы будем использовать для ввода и вывода информации




Аналогично оформляем место для 2 и 3 вопросов

**ШАГ 3**

Следующим шагом будет разработка внешнего вида приложения и определение компонент, которых мы будем использовать для ввода и вывода информации




Для 4 и 5 вопросов (множественный выбор) вместо радиокнопок вставляем по 4 Check Box - а

Размещаем компонент Memo для вывода результатов теста

И, наконец, кнопку для завершения работы с программой

**ШАГ 4**

Сделаем соответствующие надписи на компонентах формы (и не забудем разместить манифест XP)

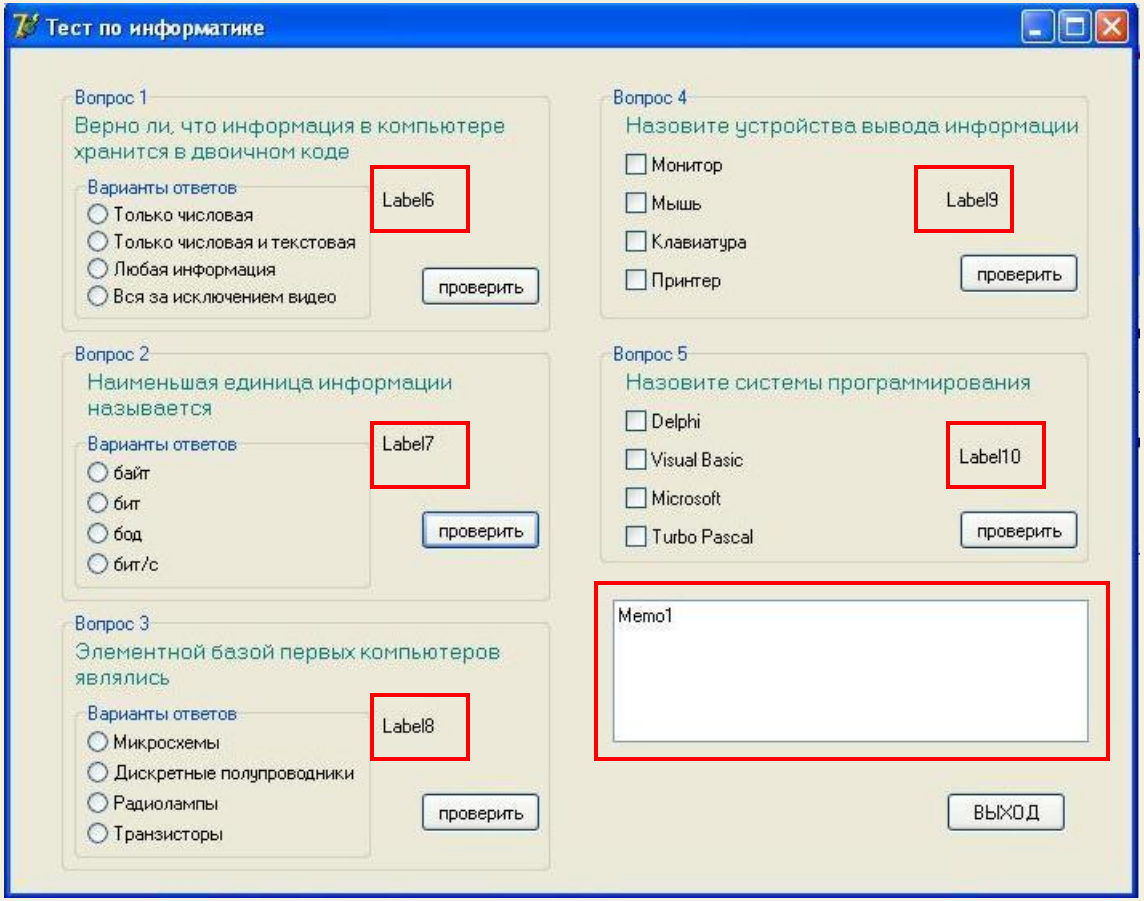



В результате мы получили примерно такую форму, на которой есть вопросы, варианты ответов, Memo для вывода результатов, кнопки проверки и выхода

**ШАГ 5**

Приложение оформлено. Начнем программирование событий. И самое первое событие, которое возникает каждый раз при запуске программы – создание формы (On Create)

## Что должно происходить при запуске программы?




1. Метки (Label 6,7,8,9,10) должны быть невидимы
2. Memo тоже должен быть невидим и появляться только после прохождения теста
3. Счетчик k верно выполненных вопросов должен обнуляться

**ШАГ 5**

Приложение оформлено. Начнем программирование событий. И самое первое событие, которое возникает каждый раз при запуске программы – создание формы (On Create)

Сделаем двойной щелчок по форме и в редакторе кода запишем процедуру создания формы (On Create)

```
var
  Form1: TForm1;
  k:integer;
implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
Memo1.Visible:=false;
Memo1.Text:='';
Label6.Visible:=false;
Label7.Visible:=false;
Label8.Visible:=false;
Label9.Visible:=false;
Label10.Visible:=false;
k:=0;
end;
```

Объявляем переменную k, которая будет являться счетчиком числа верных ответов

Делаем невидимыми Label – ы и Memo

Счетчику k присваиваем ноль

**ШАГ 6**

Сейчас напишем отклик на событие нажатия на кнопку «ПРОВЕРИТЬ»

Сделаем двойной щелчок по кнопке «ПРОВЕРИТЬ» в первом вопросе и запишем соответствующий код.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Enabled:=false;
if radiogroup1.ItemIndex=2 then
begin
k:=k+1;
label6.Font.Color:=rgb(0,150,0);
label6.Visible:=true;
label6.Caption:='ВЕРНО !';

end
else
begin
label6.Font.Color:=rgb(150,0,0);
label6.Visible:=true;
label6.Caption:='НЕВЕРНО !'
end;
end;
```

Давайте его разберем

Кнопку «ПРОВЕРИТЬ» в первом вопросе делаем недоступной, после того, как она нажата.

Это исключает возможность возврата к первому вопросу и выбора другого ответа

Проверяем условие: соответствует ли наш выбор правильному, т.е выбран ли переключатель с индексом 2

Заметьте ! Индексы переключателей в Delphi имеют нумерацию, начинающуюся с нуля, поэтому верный ответ у нас обозначен индексом 2, хотя на самом деле это третий вариант ответов (посмотрите на тест в бумажном варианте)

**ШАГ 6**

Сейчас напишем отклик на событие нажатия на кнопку «ПРОВЕРИТЬ»

Сделаем двойной щелчок по кнопке «ПРОВЕРИТЬ» в первом вопросе и запишем соответствующий код.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Enabled:=false;
if radiogroup1.ItemIndex=2 then
begin
k:=k+1;
label6.Font.Color:=rgb(0,150,0);
label6.Visible:=true;
label6.Caption:='ВЕРНО !';

end
else
begin
label6.Font.Color:=rgb(150,0,0);
label6.Visible:=true;
label6.Caption:='НЕВЕРНО !'
end;
end;
```

Давайте его разберем

Если условие выполняется (выбран верный ответ), то - значение счетчика верных ответов увеличиваем на единицу

- Метку 6(Label6), где выводится : верно/неверно делаем видимой и пишем в ней надпись ВЕРНО зеленым цветом

Если условие не выполняется, то счетчик не увеличиваем, а метку 6 делаем видимой и выводим надпись НЕВЕРНО, причем красным цветом

Очевидно, что для кнопок проверки 2 и 3 вопросов код будет абсолютно таким же за исключением номеров Label-ов для вывода ВЕРНО/НЕВЕРНО, поэтому Вы можете написать его самостоятельно (не забывайте, что в редакторе можно использовать копирование кода – это гораздо быстрее)

**ШАГ 7**

Следующим шагом опишем процедуры проверки ответа в вопросах 4 и 5 – там код будет немного другой ввиду множественного выбора и других используемых компонент

Сделаем двойной щелчок по кнопке «ПРОВЕРИТЬ» в 4 вопросе и запишем соответствующий код.

```
procedure TForm1.Button4Click(Sender: TObject);
begin
   Button4.Enabled:=false;
  if (checkbox1.Checked=true) and  (checkbox4.Checked=true)
  and  (checkbox2.Checked=false) and  (checkbox3.Checked=false)then
begin
k:=k+1;
label9.Font.Color:=rgb(0,150,0);
label9.Visible:=true;
label9.Caption:='ВЕРНО !'
end
else
begin
label9.Font.Color:=rgb(150,0,0);
label9.Visible:=true;
label9.Caption:='НЕВЕРНО !'
end;
end;
```

Проверяем выбранные ответы : ответ будет верным только тогда, когда первый и четвертый боксы будут выбраны (True), а второй и третий не выбраны (false)

В этом случае- значение счетчика верных ответов увеличиваем на единицу

- Метку 9(Label9) делаем видимой и пишем в ней надпись ВЕРНО зеленым цветом

Если условие не выполняется, то счетчик не увеличиваем, метку 9 делаем видимой и выводим надпись НЕВЕРНО красным цветом

**ШАГ 8**

Хотя вопрос 5 тоже с множественным выбором, то код обработки этой кнопки аналогичен кнопке 4, но давайте добавим в нее дополнительные функции: после нажатия этой кнопки должны выводиться результаты теста в Memo

```
procedure TForm1.Button5Click(Sender: TObject);
begin
   Button5.Enabled:=false;
 if (checkbox5.Checked=true) and  (checkbox6.Checked=true)
 and  (checkbox8.Checked=true) and  (checkbox7.Checked=false)then
begin
k:=k+1;
label10.Font.Color:=rgb(0,150,0);
label10.Visible:=true;
label10.Caption:='ВЕРНО !'
end
else
begin
label10.Font.Color:=rgb(150,0,0);
label10.Visible:=true;
label10.Caption:='НЕВЕРНО !';
end;
memo1.Visible:=true;
memo1.Text:='Всего вопросов - 5'+#13+#10+'Количество верных ответов - '
+inttostr(k)+#13+#10+ 'Процент выполнения теста - '+floattostr(k/5*100)+' %';
```

Эта часть аналогична кнопке 4 вопроса и объяснений здесь не требуется

Делаем видимым компонент Memo для вывода результатов

Выводим результаты теста

**ШАГ 8**

Давайте посмотрим, как выводится в Memo результат теста

```
memo1.Text:='Всего вопросов - 5'+#13+#10+'Количество верных ответов - '
+inttostr(k)+#13+#10+ 'Процент выполнения теста - '+floattostr(k/5*100)+' %';
```

Всего вопросов - 5

В апострофах (одинарных кавычках), как и в Турбо Паскале, выводится строка символов:

Количество верных ответов

Опять вывод строки

Давайте вспомним таблицу кодировки символов - ASCII – символы с номерами 13 и 10 зарезервированы для перевода курсора на следующую строку (Enter) и в ее начало соответственно

Это значит, что вывод следующего элемента начнется с начала следующей строки

БАЗОВАЯ ТАБЛИЦА ASCII

| 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ♥ 3 | ♦ 4 | ♣ 5 | ♠ 6 | • 7 | ◘ 8 | ○ 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ! 19 | 20 | 21 | _ 22 | 23 | ↑ 24 | ↓ 25 | → 26 | ← 27 | 28 | ↔ 29 | ▲ 30 | ▼ 31 |
| # 35 | $ 36 | % 37 | & 38 | ' 39 | ( 40 | ) 41 | * 42 | + 43 | , 44 | - 45 | . 46 | / 47 |
| 3 51 | 4 52 | 5 53 | 6 54 | 7 55 | 8 56 | 9 57 | : 58 | ; 59 | < 60 | = 61 | > 62 | ? 63 |

**ШАГ 8**

Давайте посмотрим, как выводится в Memo результат теста

```
memo1.Text:='Всего вопросов - 5'+#13+#10+'Количество верных ответов - '
+inttostr(k)+#13+#10+'Процент выполнения теста - '+floattostr(k/5*100)+' %';
```

Вывод символа %

Дальше подсчитываем процент выполнения и переводим его в строковый тип (из вещественного) (floattostr)

Вывод текст Процент выполнения теста

Вывод значение счетчика k, естественно, перевод его в строковый тип (с этим мы уже знакомы) (inttostr)

Опять переводим курсор на начало следующей строки

И вот результат, выведенный в Memo

**ШАГ 9**

Последняя кнопка – ВЫХОД

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

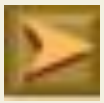
Закрывается

**ШАГ 10**

Сохраняем и компилируем программу (см. предыдущие уроки)

А сейчас попробуем запустить наш тест

**Запустить ->**



На этом наш урок закончен.

Конечно, наша тестирующая программа получилась довольно примитивная, у нее отсутствуют многие необходимые функции. По мере изучения Delphi мы составим более серьезную и удобную тестирующую оболочку, а пока самое главное, что наша программа работает и мы научились использовать компоненты для проверки условий