

Третье занятие: переменные и условный оператор

Переменные.

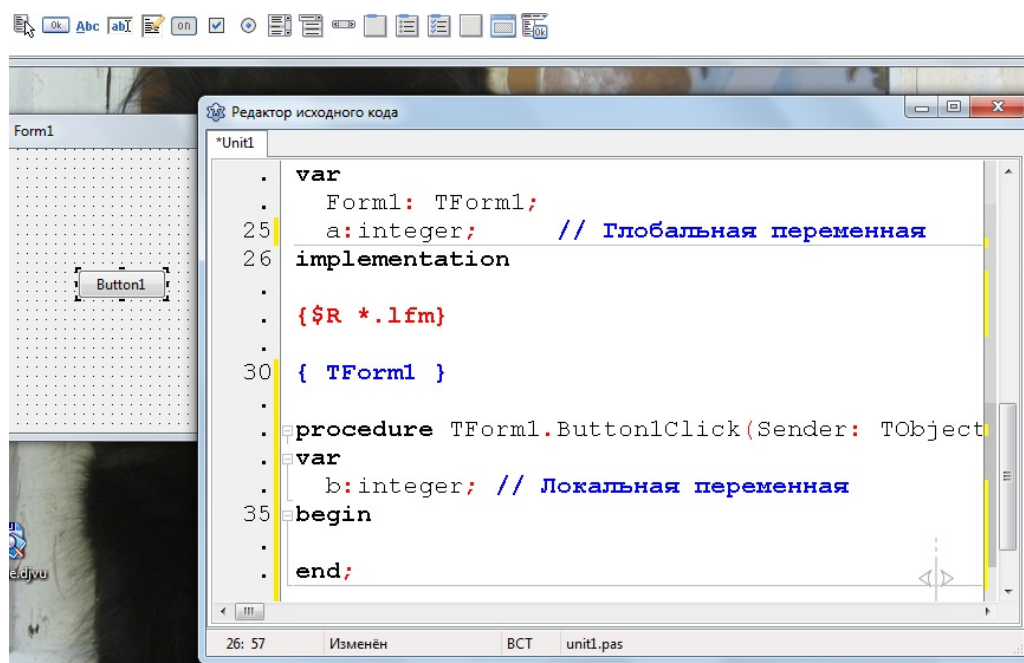


Рис. 1: Объявление переменных

Переменная - это область оперативной памяти, которая выделяется для хранения определённого типа данных, и которой присваивается определённое имя, по которому можно обращаться к этим данным. Например свойство метки **Label1.Caption** можно рассматривать, как переменную с таким именем, предназначенную для хранения строк.

Переменная создаётся ключевым словом **var**. После него (в той же строке или в следующей) указывается *имя* создаваемой переменной, затем двоеточие, и после двоеточия - её *тип*. Можно создать несколько переменных, перечислив их имена через запятую.

Имя создаваемой переменной должно состоять из латинских букв (заглавные буквы считаются совпадающими с прописными) и цифр, причём начинаться - с буквы. У создаваемых переменных должны быть различные имена, если у них одинакова *область видимости* (см. дальше). Желательно делать все названия различными и осмысленными.

Тип определяет какие данные могут храниться в этой переменной: например для хранения целых чисел надо использовать тип **integer**, для хранения вещественных чисел - **real**, строк - **string**, отдельных символов - **char**.

Переменные делятся на *локальные* и *глобальные*. Если в программе есть несколько процедур (например обрабатываются нажатия нескольких кнопок), то переменные, созданные внутри процедуры называют *локальными* - их можно использовать только внутри этой процедуры. В таких случаях говорят, что эта переменная *видна* только в этой процедуре. Если же создать *глобальную* переменную, то она будет видна во всей программе и во всех процедурах, то есть её *областью видимости* является вся программа.

Но есть одно исключение: если создать *глобальную* переменную с каким-то именем, а потом в какой-то процедуре создать *локальную* переменную с таким же именем, то в этой процедуре будет видна локальная переменная вместо глобальной.

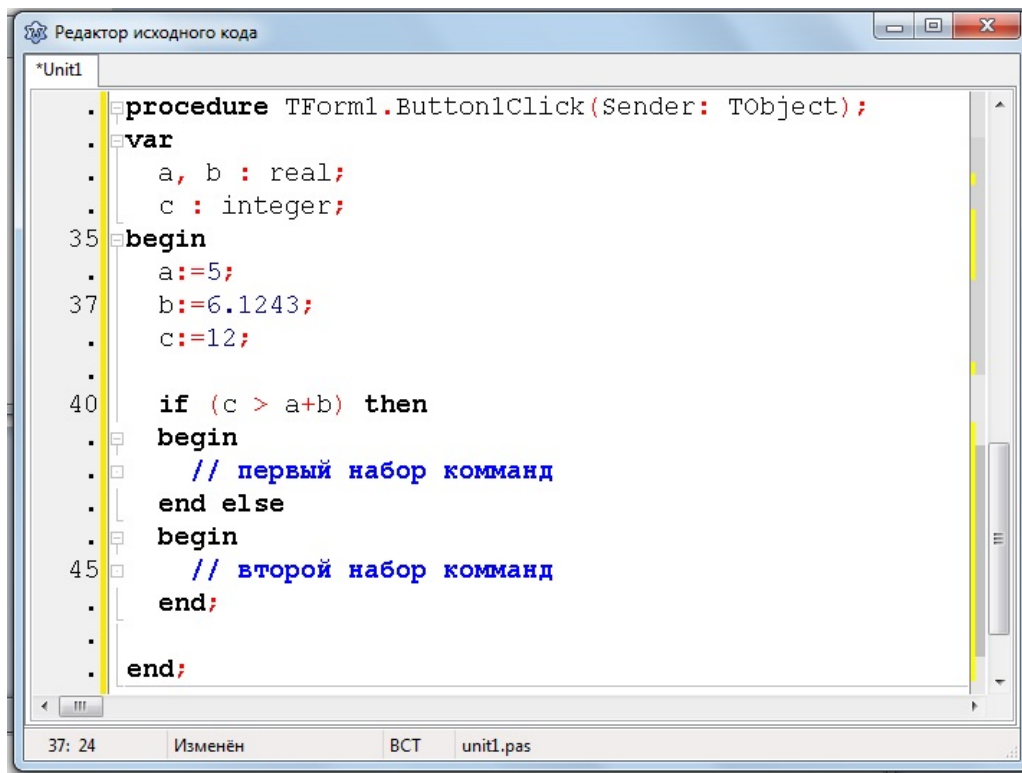


Рис. 2: Синтаксис условного оператора

Условный оператор

Условный оператор позволяет запрограммировать выполнение различных действий, в зависимости от выполнения какого-то условия. Если условие выполнено (в данном случае если $c > a + b$), то выполняется первый набор команд, в противном случае - второй набор команд. На рисунке 2 показан самый общий вид условного оператора, однако есть два более коротких частных случая:

1. Если нужно выполнять какие-то действия только если условие выполнено, а в противном случае ничего делать не надо, то можно написать так (рис.3):

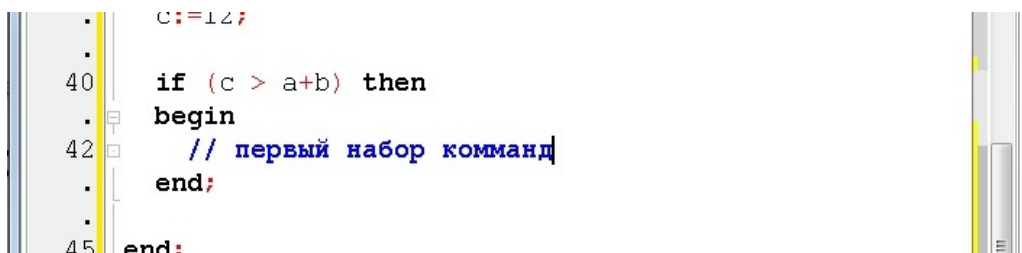


Рис. 3: Синтаксис условного оператора без else

2. Если нужно выполнять не набор команд, а всего одну команду, то можно не писать **begin** и **end**. При этом перед else после команды не должна ставиться точка с запятой, в отличие от большинства ситуаций. (Кстати обратите внимание как ставятся точки с запятой в общем виде условного оператора).

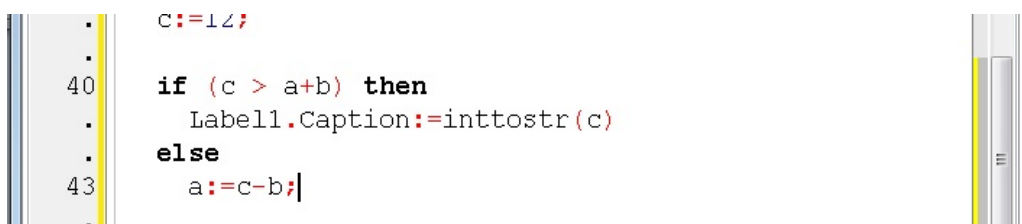


Рис. 4: Синтаксис условного оператора с одной командой

Эти способы можно комбинировать: может быть **if** без **else** с одной командой; так же может быть первый набор команд в виде набора, а второй - одной командой (или наоборот).

Задания

1. Продолжение a+b. Напишите программу аналогичную **a+b** из второго домашнего задания, которая может не только складывать, но и вычитать, умножать и делить два числа. При этом числа могут быть не целые; действие должно определяться не разными кнопками, а отдельным окошком **edit**, в которое будет вводиться **+**, **-**, ***** или **/**. По нажатию кнопки программа должна проверить что записано в этом третьем **edit**-е и в зависимости от этого выполнить нужную арифметическую операцию. Кроме этого при делении нужно проверять, чтобы делитель не был нулём (если он ноль то вместо результата деления надо вывести сообщение, что на ноль делить нельзя).

2. Пароль. *В принципе всё что нужно для выполнения этого задания уже было рассказано на последних двух занятиях. Но так как я не успел это повторить, то это задание будет засчитываться только как бонус.* Напишите модификацию программы, убегающей от мышки: кнопка должна убегать от курсора не по наведению мышки, а при неверном вводе пароля в окошко **edit**; при правильном содержании **edit**-а программа должна закрываться. У окна программы не должно быть кнопок закрытия, сворачивания и разворачивания на весь экран: для этого надо развернуть свойство **BorderIcons** формы, и поставить в нём **False** напротив всех четырёх пунктов (один из них **False** по умолчанию). При нажатии кнопки программа должна проверять содержимое **edit**-а, и при правильном его значении - закрываться, а при неправильном пароле текст в **edit**-е должен обнуляться, а кнопка - убегать в произвольное место на форме.

Команды, которые понадобятся для выполнения заданий:

Для первого:

StrToFloat(string) - преобразование строки **string** в число.

FloatToStr(real) - преобразование числа **real** (не обязательно целого) в строку.

+, **-**, *****, **/** - операции, которые можно применять к числам.

Условный оператор (можно использовать не только в общем виде - см. выше):

```
if (условие) then
begin
    команды;
end;
```

В качестве условия может выступать сравнение двух значений, например двух переменных или переменной и константы (сравнивать две константы обычно не имеет смысла). Таким образом условие может выглядеть так **if (edit3.Text = '+') then ...**. **Edit3.Text** - это строка, которую можно сравнивать только с другой строкой. Поэтому из **+** надо сделать строку, взяв его в одинарные кавычки.

Проверка на то, что делитель ноль, может выглядеть так: **if (strtofloat(edit2.text) = 0) then ...**

Таким образом код процедуры нажатия кнопки может выглядеть так, как показано в конце pdf-ки.

Для второго:

random(n) - функция, которая возвращает случайное целое число от 0 до n-1.

Это значит, что при приравнивании к ней какой-то переменной, в этой переменной получится случайное значение.

Свойства формы **form1.Height** и **form1.Width** - её высота и ширина соответственно. Это ширина в пикселях; такие же свойства есть и у других компонентов: кнопок, меток и т.п.

Свойства кнопки **button1.Top** и **button1.Left** - координаты её левого верхнего угла относительно левого верхнего угла формы. **Top** - отступ по вертикали сверху, **Left** - отступ по горизонтали слева.

close; - команда на закрытие программы.

Возможный код процедуры для первой программы:

```
procedure TForm1.Button1Click(Sender: TObject); // эта строчка создаётся Lazarus-ом
var a, b: real;
begin // эта строчка создаётся Lazarus-ом

    a:=StrToFloat(edit1.Text);
    b:=StrToFloat(edit2.Text);
    if (edit3.text = '/') then
    begin
        if (b = 0) then
        begin
            label1.Caption:='На ноль делить нельзя!';
        end else
        begin
            label1.Caption:=FloatToStr(a / b);
        end;
    end;

    if (edit3.text = '*') then
    begin
        label1.Caption:=FloatToStr(a * b);
    end;

    if (edit3.text = '+') then
    begin
        label1.Caption:=FloatToStr(a + b);
    end;

    if (edit3.text = '-') then
    begin
        label1.Caption:=FloatToStr(a - b);
    end;

end; // эта строчка создаётся Lazarus-ом
```