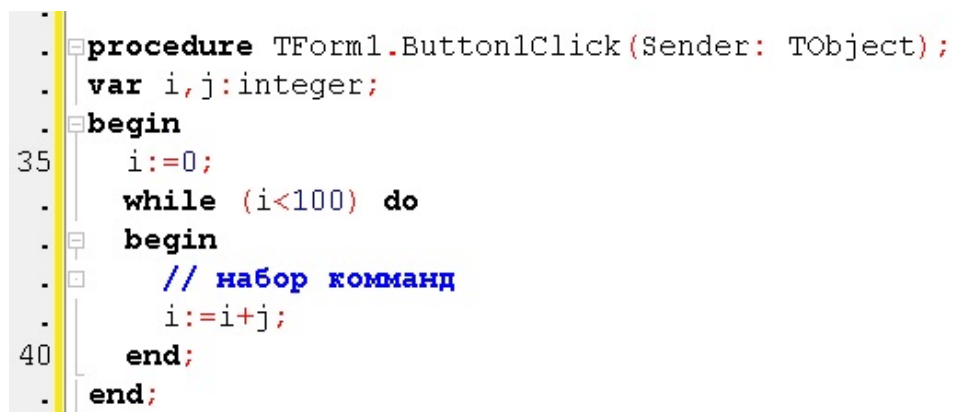


Цикл while, таймер.

В общем виде цикл **while** пишется так (см. рис. 1):

```
while ( условие повтора выполнения ) do
begin
    набор команд
end;
```



```
. procedure TForm1.Button1Click(Sender: TObject);
.   var i,j:integer;
.   begin
35     i:=0;
.     while (i<100) do
.       begin
.         // набор команд
.         i:=i+j;
40     end;
.   end;
```

Рис. 1: Синтаксис цикла **while**.

Так же как и для цикла **for** набор команд называют *телом цикла*, каждое их выполнение - *итерацией цикла*. Однако цикл **while** выполняется не фиксированное количество раз, а до тех пор, пока выполняется указанное условие. Если в тот момент, когда программа доходит до выполнения цикла, условие не верно, то цикл будет пропущен и набор команд не будет выполнен ни разу. Поэтому важно не забыть задать начальные значения используемых в условии переменных.

Кроме того нужно контролировать, что после некоторого количества итераций условие перестанет выполняться, в противном случае получится так называемый бесконечный цикл (именно отсюда пошло слово "зациклиться"; зациклившейся называют программу, которая тщетно пытается выполнить бесконечный цикл). Бесконечные циклы иногда используются, но при этом в них предусматривается возможность выхода командой в теле цикла, и "бесконечными" они являются только с точки зрения конструкции самого **while**-а.

Как обычно если нужно многократно выполнить всего одну команду, то **begin** и **end**; можно не писать.

Timer

Таймер (Timer) - специальный компонент, находящийся на вкладке **System** палитры компонентов (см. рис. 2). Для его использования таймер должен быть помещён на форму, однако после компиляции в получившейся в итоге программе он не отображается.

Основные параметры таймера - **Enabled** и **Interval**. Первый определяет включен ли таймер (отсчитывает ли он время); **True** - таймер включен, **False** - соответственно выключен. Второй определяет время, которое отсчитывает таймер (целое число в миллисекундах). Оба этих параметра могут изменяться внутри программы, например возможна команда **Timer1.Enabled:=True;**

Основная процедура таймера, как обычно, может быть создана двойным щелчком по нему; во вкладке событий в инспекторе объектов она называется **OnTimer**. Эта процедура начинает выполняться, когда таймер отсчитал заданное время. Таким образом при включении таймера (если он сразу включен - то при запуске программы) он начинает отсчёт. Когда проходит указанное в параметре **Interval** время

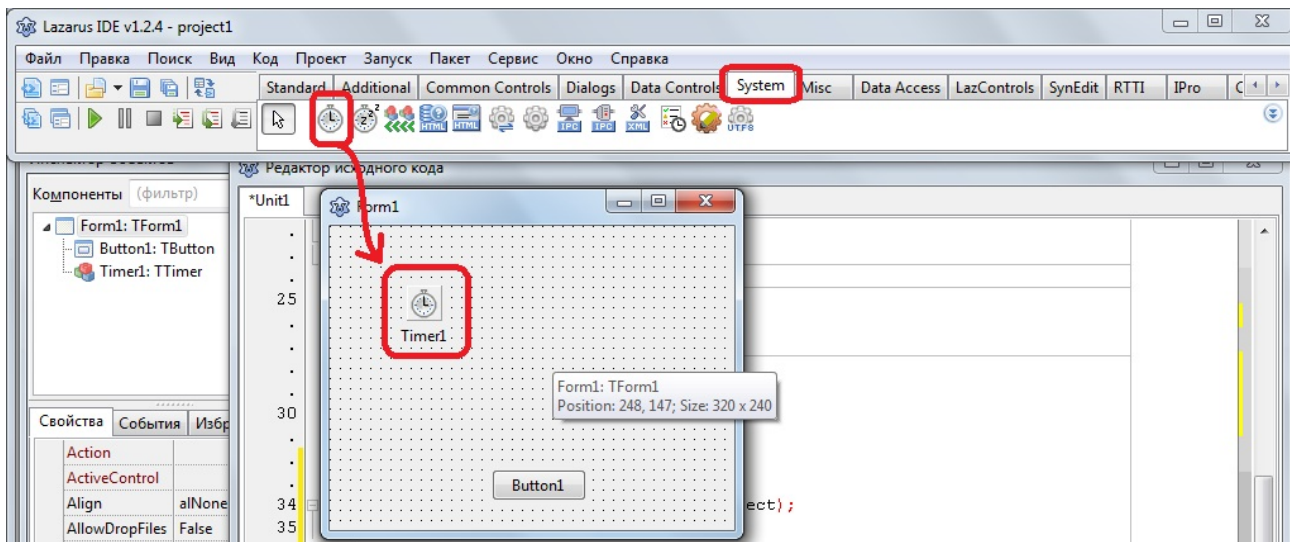


Рис. 2: Компонент **Timer**

- выполняется событие таймера, и он начинает считать заново (если в процедуре не выполнится команда **Timer1.Enabled:=False;**). Прекратить повторяющееся выполнение этой процедуры можно либо с помощью команды **Timer1.Enabled:=False;** выполняемой либо в процедуре таймера (возможно по какому-либо условию), либо с помощью внешнего (для таймера) события, по которому выполнится эта же команда.

Таким образом с помощью таймера, так же как и с помощью циклов, можно запрограммировать повторяющееся выполнение команд. Это используется в ситуации, когда нужно выполнять команды с некоторой задержкой, например при рисовании перемещающихся по канве с определённой скоростью объектов (скорость определяется в том числе установленной задержкой). Это можно сделать и в обычном цикле с помощью команды **sleep(время задержки);**, однако такой способ, в отличие от таймера, имеет существенные недостатки.

Домашнее задание: написать программу, которая рисует падающие капли, взяв за основу предыдущее домашнее задание. При этом надо использовать таймер, по срабатыванию которого происходит изменение картинки. Простая версия: по срабатыванию к картинке добавляется один ряд решётки каплей, остальные неподвижны. Более интересная версия - по срабатыванию все нарисованные капли ненамного сдвигаются вниз, при этом расстояние до верхнего ряда от верхнего края канвы увеличивается, и когда оно превышает расстояние между рядами - добавляется ещё один ряд капель.

В принципе для рисования ряда капель удобнее использовать цикл **for**, однако для тренировки можно сделать это и с помощью **while**.

P.S. На следующей странице добавлен разбор домашнего задания.

Если добавить на канву **Image1**, **Button1** и **Timer1**, то для кнопки и таймера должны выполняться процедуры, показанные ниже. Кроме того обратите внимание на объявление глобальных переменных.

```
...
var
    Form1: TForm1;
    i, y : integer;

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
    with form1.image1.canvas do
    begin
        brush.color:=clWhite;
        rectangle(-5,-5,5000,5000);
    end;

    timer1.enabled:=true;
    i:=0;
    y:=0;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var j,x:integer;
begin
    with form1.image1.canvas do
    begin
        brush.color:=clBlue;
        pen.color:=clBlue;
        i:=i+1;
        if i>50 then timer1.enabled:=false;
        y:=y+10;
        j:=0;
        while (j<100) do
        begin
            j:=j+1;
            x:=10*j;
            ellipse(x,y,x+5,y+7);
        end;
    end;
end;
end;
...
```