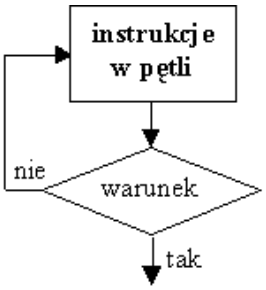
LEKCJA 06 – pętle logiczne

Pętla iteracyjna wykonywała się z góry określoną ilość razy, zwiększają (lub zmniejszając) swój licznik (zmienną sterującą). W programach zachodzą jednak sytuacje bardziej skomplikowane – należy wykonywać instrukcje cyklicznie, aż wystąpi określone zdarzenie (lub też dopóki ono zachodzi). Pętle takie nazywamy logicznymi – ich zakończenie zależy od warunku logicznego.

PRZYKŁAD LEKCJA061 – 11 pionowych odcinków – repeat, 11 poziomych - WHILE

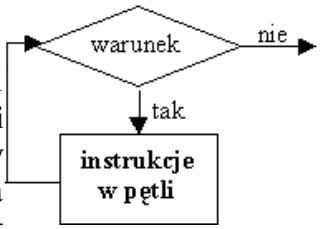
Pętla logiczna POWTARZAJ AŻ




```
i:=0;
repeat
  canvas.moveto(100+i*20,100);
  canvas.lineto(100+i*20,300);
  i:=i+1;
until i>10;
```

Ciąg instrukcji pomiędzy słowami REPEAT…UNTIL jest powtarzany, **aż warunek logiczny stanie się prawdziwy.** Innymi słowy: **dopóki jest fałszywy** instrukcje są wykonywane. Warunek logiczny jest sprawdzany **po wykonaniu instrukcji,** dlatego instrukcje muszą wykonać się **przynajmniej jeden raz**. Zmiana zmiennej sterującej musi być w tej pętli realizowana „ręcznie” (np. i:=i+1). Zmienna musi też być ustawiona (zainicjowana) na początku (np. i:=0;). Fragment procedury w ramce rysuje 11 pionowych odcinków.

Pętla logiczna DOPÓKI




```
i:=0;
while i<11 do
begin
  canvas.moveto(100,100+i*20);
  canvas.lineto(300,100+i*20);
  i:=i+1;
end;
```

Pętla wymaga dodatkowych słów BEGIN i END. Instrukcje wykonywane są **dopóki warunek logiczny jest prawdziwy**. Gdy warunek logiczny stanie się fałszywy, pętla kończy działanie. **Najpierw sprawdzany jest warunek, a potem dopiero wykonuje się działanie. Instrukcja nie wykona się ani raz,** jeśli warunek będzie spełniony zaraz na początku (np. źle zainicjowano zmienną sterującą lub zły jest warunek zakończenia). Fragment procedury w ramce rysuje 11 poziomych odcinków.

Pętle logiczne są bardziej uniwersalne niż iteracyjna – można w nich wszystko zmieniać i dowolnie kontrolować: ustawić początek, sprawdzać na końcu lub na początku, przed lub po wykonaniu instrukcji, zmieniać wartość zmiennej sterującej. Warunki zakończenia mogą być bardziej rozbudowane stosując wyrażenia logiczne. Jest tylko jeden poważny problem - jeśli błędnie skonstruujemy warunek wychodzenia z pętli, może nigdy się nie zakończyć – program się zawiesi!

Operatory i wyrażenia logiczne – wynikiem jest PRAWDA lub FAŁSZ (1 lub 0)

(dla poniższych przykładów - w zmiennej X przechowywana jest liczba 5)

suma OR	(X>0) OR (X <-1)	prawda
iloczyn AND	(X>0) OR (X <-1)	fałsz
zaprzeczenie NOT	NOT(x>0)	fałsz

<>	X<>5	fałsz
=	X = 1	fałsz

<=	X <= 5	prawda
>=	X >=5	prawda

<	X < 1	fałsz
>	X > 1	prawda

Animacja komputerowa

Wrażenie, że obiekty są ruchome uzyskujemy dzięki szybkiemu wyświetlaniu kolejnych nieruchomych obrazów. Aby "oszukać" wzrok - by animacja była płynna - kolejne obrazy muszą być wyświetlane w tempie przynajmniej 30 obrazów na sekundę. Prosta animacja realizowana jest schematycznie w następującej pętli:

(1) rysujemy obiekt,
(2) wstrzymujemy działanie programu,
(3) usuwamy obiekt,
(4) obliczamy nowe położenie obiektu

```
var x:integer;
begin
  x:=0;
  repeat
    Prost(x,100,40,200,clBlack);
    sleep(10);
    Prost(x,100,40,200,clBtnFace);
    x:=x+1;
  until x > 600;
end;
```

PRZYKŁAD - LEKCJA062

Fragment programu przesuwa prostokąt w prawo, aż zmienna X osiągnie wartość 600. Początkowo zmienna X ustawiona jest na 0 – start rysowania. Pętla kończy się, gdy zmienna X będzie większa niż 600.

```
repeat
  Prost(x,100,40,200,clBlack);
  sleep(5);
  Prost(x,100,40,200,clBtnFace);
  x:=x-1;
until x <0;
```

Jeśli chcemy, aby prostokąt przesuwał się w lewo – wystarczy zmniejszać współrzędną X. A skoro po zakończeniu pierwszej pętli współrzędna X miała wartość 601, nie trzeba jej specjalnie ustawiać, wystarczy dopisać nową pętlę (pętla w drugiej ramce).

```
licznik:=0;
repeat
  repeat
        {w prawo}
  until x>600;
  repeat
        {w lewo}
  until x<0;
  licznik:=licznik+1;
until licznik=5;
```

Skoro prostokąt wrócił na swoją pierwotną pozycję można cały proces (dwóch pętli) powtarzać w pętli obejmującej ruch w prawo i powrót w lewo. Kiedy taki cykliczny proces ma się zakończyć? Można wykonać zewnętrzną pętlę określoną ilość razy. W ramce obok posługujemy się zmienną LICZNIK. Ustawiamy jej początkową wartość na 0. Po każdym powrocie prostokąta, na końcu drugiej pętli, zwiększamy licznik o 1. Kończymy zewnętrzną pętlę, gdy licznik równy np. 5.

W podobny sposób można zrealizować przesuwanie w pionie – wystarczy zadeklarować zmienną Y i w podobny sposób (jak X) ją zmieniać. Jeśli jednocześnie będziemy zmieniać X i Y – prostokąt będzie przesuwał się ukośnie.

ANIMACJA w DELPHI

Nie jest to rozwiązanie szczęśliwe dla okienek Windows – musimy czekać, aż prostokąt zrobi 5 razy swoje „kółko". Nie możemy w tym czasie nic zrobić. Wstrzymywane jest wszelkie oddziaływanie myszy na okienka. Program przestaje być interaktywny dopóki nie zakończą się pętle. Gdyby warunek zakończenia byłyby źle skonstruowany – program by się nigdy nie skończył. W kolejnym przykładzie zmodyfikujemy program tak, aby można go było przerwać w momencie naciśnięcia dowolnego klawisza.

PRZYKŁAD LEKCJA063

Zakończenie pętli po naciśnięciu dowolnego klawisza Na przykładzie poprzedniego programu LEKCJA062:

- zadeklaruj zmienną globalną KONIEC (górna ramka)
- zdarzenia (EVENTS) formatki – kliknij podwójnie w ONKEYDOWN (sprawdzamy klawiaturę)
- wpisz instrukcję **KONIEC:=TRUE;**

W procedurze dla zdarzenia ONRESIZE

- wartość początkowa zmiennej KONIEC:=FALSE
- usuń deklarację zmiennej LICZNIK
- usuń z programu instrukcję LICZNIK:=LICZNIK+1;
- zmień zakończenie zewnętrznej pętli REPEAT na **UNTIL KONIEC=true;**
- wewnątrz obu pętli umieść instrukcję **Application.ProcessMessages;**

```
var
  Form1: TForm1;
  KONIEC:boolean;
```

```
procedure TForm1.FormKeyDown
(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  KONIEC := True;
end;

procedure TForm1.FormResize(Sender: TObject);
var x:integer;
begin
  x:=0;
  KONIEC:=false;
 REPEAT
        repeat
        {ruch w prawo}
        Application.ProcessMessages;
        until x > 600;

        repeat
        {ruch w lewo}
        Application.ProcessMessages;
        until x <0;
 UNTIL KONIEC=true;
```

Po każdym przesunięciu prostokąta o jeden piksel wykonywana jest instrukcja **Application.ProcessMessages** i sprawdzana jest klawiatura. Jeżeli w tym momencie naciśniemy dowolny klawisz nasza instrukcja urochamia kolejną **FormKeyDown** - zmienna KONIEC przyjmie wartość TRUE, i zewnętrzna pętla REPEAT zostanie zakończona.

ANIMACJA za pomocą komponentu TIMER

W typowym ObjectPascal'u animację realizuje się nieco inaczej. Jednym ze sposobów jest zastosowanie systemowego komponentu **TIMER**. Nie musimy sprawdzać reakcji na klawiaturę, nie musimy deklarować dodatkowych zmiennych, nie potrzeba też pętli REPEAT. Całą animacją zarządza komponent TIMER, który automatycznie jest uruchamiany, co określony czas. Nie ma pętli, więc nie jest blokowana myszka i klawiatura, okienka są interaktywne, i można je bez problemu skalować, minimalizować, zamykać itp.

PRZYKŁAD LEKCJA064 – przesuwany napis

Przesuwamy napis w górnej części okienka, od brzegu do brzegu formatki.

```
var
  Form1: TForm1;
  kierunek:integer=1;
```

- nowa aplikacja
- na formatkę wstawiamy komponent LABEL
- zmieniamy napis - właściwość CAPTION, np. 'ALA MA KOTA'
- zmieniamy czcionkę - właściwość FONT: czcionka, wielkość, kolor
- ustalamy położenie: z lewej strony, u góry formatki: właściwość LEFT – 0 i TOP – 0
- deklarujemy zmienną o nazwie **kierunek** i nadajemy jej od razu wartość początkową (patrz ramka obok)
- wstawiamy komponent TIMER z zakładki SYSTEM w dowolnym miejscu formatki
- nowa procedura TIMER1TIMER (klikamy podwójnie w komponent TIMER lub w zdarzeniach komponentu TIMER w pole ONTIMER)
- wpisujemy dwie instrukcje z poniższej ramki:

```
if (Label1.Left+label1.width>form1.width) or  (label1.left<0) then kierunek:=-kierunek;
label1.Left:=label1.left+kierunek;
```

UWAGA - instrukcję warunkową IF THEN poznamy na kolejnej lekcji

Tutaj sprawdzamy, czy prawy koniec etykiety nie przekroczył prawej krawędzi formatki LUB czy lewy koniec etykiety nie przekroczył lewego brzegu formatki, i wtedy zmieniamy kierunek na przeciwny. W drugiej instrukcji zmieniamy (zwiększamy lub zmniejszamy, gdy kierunek ujemny) położenie etykiety na formatce.

PRZYKŁAD LEKCJA065 – pulsujący przycisk

Wielkość przycisku zmienia się cyklicznie.

- nowa aplikacja
- kwadratowy przycisk BUTTON na środku formatki
- początkowe wymiary przycisku 100x100 (HEIGHT x WIDTH)
- w procedurze do zdarzenia ONCLICK przycisku
  * zadeklarować zmienną W
  * wpisać program w ramce

Zmieniamy wymiary przycisku, co 2 piksele i jednocześnie przesuwamy go, aby był zawsze na środku.

```
var w:integer;
begin
  w:=100;
  repeat
    Button1.Width:=w;
    Button1.Height:=200-w;
    Button1.Top:=Button1.Top+1;
    Button1.Left:=Button1.Left-1;
    w:=w+2;
    repaint;
  until w>=190;
  repeat
    Button1.Width:=w;
    Button1.Height:=200-w;
    Button1.Top:=Button1.Top-1;
    Button1.Left:=Button1.Left+1;
    w:=w-2;
    repaint;
  until w<=100;
end;
```

ĆWICZENIA (za pomocą repeat lub while) na podstawie LEKCJA063

pionowy odcinek przesuwa się poziomo od…do, w prawo lub w lewo
poziomy odcinek przesuwa się pionowo od…do, w górę lub w dół
prostokąt zmienia swoje wymiary w jednym lub kilku kierunkach

SPRAWDZIAN

Poziomy lub pionowy odcinek o podanej długości, przesuwa się w pionie (w górę lub w dół) lub poziomie (w prawo lub w lewo), od punktu do punktu na formatce. Odcinek „odbija się" od brzegów formatki. Dwa odcinki przesuwają się i odbijają jednocześnie.