LEKCJA 10 – Funkcje

W matematyce funkcja wylicza jakąś wartość dla podanego argumentu. Na przykład funkcja sinus dla kąta równego 90° wylicza wartość 1. W programowaniu funkcja pełni identyczną rolę, jak procedura – jest to fragment programu zamknięty w jedną całość. Od procedury odróżnia ją możliwość wyliczania (jak w matematyce) jakiejś wartości. W pascalu istnieje wiele gotowych funkcji (np. cosinus sinus itp.), ale też można definiować swoje własne. Funkcje deklarujemy w podobnym miejscu programu jak procedury. Przykład dla funkcji obliczającej kwadrat liczby (ramka obok):

```
FUNCTION kwadrat(liczba:byte):longint;
begin
  kwadrat:=liczba*liczba;
end;
```

W programie głównym funkcję nie możemy wywołać tak bezpośrednio, jak procedurę. Funkcja musi być skojarzona ze zmienną lub jej wynik podany bezpośrednio jako argument dla innej procedury lub funkcji.

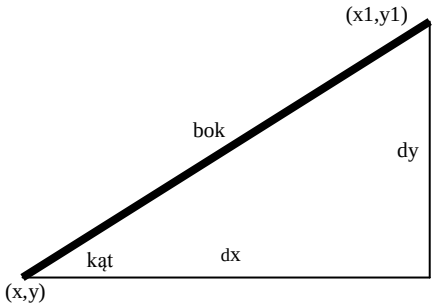
k:=kwadrat(10); wynik funkcji przypisujemy do zmiennej K

ShowMessage(IntToStr(kwadrat(10))); wynik funkcji przekazany bezpośrednio do innej instrukcji

Wewnątrz funkcji można zdeklarować zmienne i stałe **lokalne** (jak w procedurze), które funkcjonują jedynie podczas działania funkcji i giną po jej zakończeniu (nie zajmują pamięci). Nie kolidują te zmienne ze zmiennymi o takich samych nazwach zdeklarowanych w programie głównym – zmienne **globalne**. Jest to tzw. **przesłanianie zmiennych**. We wnętrzu funkcji **musi wystąpić instrukcja przypisania do nazwy funkcji wyliczonej wartości,** np. **kwadrat:=liczba*liczba**. Tą wyliczoną wartość będzie zwracała funkcja, gdy wykonamy ją w programie głównym.

FUNKCJE W GRAFICE

Bardzo łatwo jest narysować linie, które są określone końcowymi punktami (jak w LineTo). Gorzej, jeśli mamy podany punkt początkowy, długość boku i należy linię wykreślić pod odpowiednim kątem (tak jak np. w logo). Ponieważ w Pascalu brak jest odpowiednich instrukcji, dlatego je zdefiniujemy.



Punkt końcowy linii (X1,Y1) wyliczymy z wzorów (trójkąt prostokątny):

X1 = X+dX = X+bok*cos(kat*pi/180)

Y1 = Y+dY = Y+bok*sin(kat*pi/180)

Kąt podajemy w stopniach, dlatego należy zamienić go na radiany. Stała PI jest ustalona przez system.

DX i DY należy zamienić na odpowiednie funkcje o nazwach deX i deY.

Gotowy wzór w pascalu będzie miał postać:

dX:=round(bok*cos(kat*pi/180))

dY:=round(bok*sin(kat*pi/180))

zaś gotowe funkcje będą miały postać:

```
function deX(bok,kat:integer):integer;
begin
   deX:=round(bok*cos(kat*pi/180))
end;

function deY(bok,kat:integer):integer;
begin
   deY:=round(bok*sin(kat*pi/180))
end;
```

W obu funkcjach użyliśmy dodatkowej funkcji ROUND, która zamienia liczbę rzeczywistą, powstałą w wyniku mnożenia i dzielenia na liczbę całkowitą, bo tylko w oparciu o liczby całkowite można rysować na ekranie graficznym.

Procedura LINIA - rysowanie odcinka za pomocą funkcji deX i deY - podajemy punkt początkowy, długość i kąt oraz kolor

```
procedure linia(x,y,bok,kat,kolor:integer);
begin
  canvas.pen.color:=kolor;
  canvas.moveto(x,y);
  canvas.LineTo(x+deX(bok,kat),y-deY(bok,kat))
end;
```

Rysowanie kwadratu za pomocą instrukcji LINIA

Kwadrat o boku 100, lewy górny róg w punkcie 200,200, kolor 12

```
linia(200,200,100,0,12);
linia(300,200,100,90,12);
linia(300,100,100,180,12);
linia(200,100,100,270,12);
```

PRZYKŁAD LEKCJA101 - kręcący się wskaźnik

- instrukcje umieszczone w procedurze do zdarzenia ONRESIZE formatki

```
for kat:=360 downto 1 do
begin
  linia(300,200,200,kat,13);
  sleep(20);
  linia(300,200,200,kat,0);
end;
```

PRZYKŁAD LEKCJA102 - sekundnik

- instrukcje umieszczone w procedurze do zdarzenia ONRESIZE formatki
- w procedurze do zdarzenia ONKEYDOWN formatki instrukcja KONIEC:=TRUE (jak w poprzednich programach z animacją)

```
kat:=360;
repeat
  linia(320,200,200,kat,clRed);
  sleep(1000);
  linia(320,200,200,kat,clBtnFace);
  kat:=kat-6;       {co 6 stopni skok}
  if kat <=0 then kat:=360;
  application.ProcessMessages;
until koniec;
close
```

PRZYKŁAD LEKCJA103 - radar

- instrukcje umieszczone w procedurze do zdarzenia ONRESIZE formatki
- w procedurze do zdarzenia ONKEYDOWN formatki instrukcja KONIEC:=TRUE (jak w poprzednich programach z animacją)
- PRAWY i DOLNY – szerokość i wysokość formatki – radar będzie odbijał się od brzegów aktualnego okienka

```
x:=0;
y:=0;
bok:=100;
kat:=0;
dx:=5;
dy:=5;
prawy:=Form1.Width;
dolny:=Form1.Height;
repeat
  linia(x,y,bok,kat,ClRed);
  sleep(10);
  linia(x,y,bok,kat,clBtnFace);
  kat:=kat+3;
  x:=x+dx;
  y:=y+dy;
  if kat >= 360 then kat:=0;
  if (x+bok > prawy) or (x < 0)
        then dx:=-dx;
  if (y+bok > dolny) or (y < 0)
        then dy:=-dy;

  application.ProcessMessages;
until koniec;
close
```

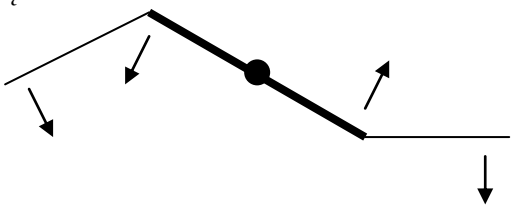
PRZYKŁAD LEKCJA104 – radar za pomocą komponentu TIMER

- zmienne zadeklarowane jako globalne
- inicjacja zmiennych (ustawienia początkowe) w procedurze do zdarzenia ONCREATE
- w procedurze do zdarzenia ONTIMER animacja (bez pętli REPEAT i zmiana kolejności, najpierw wymazanie, potem obliczanie i na końcu rysowanie (w ramce)
- nie potrzeba zmiennej koniec i procedury do zdarzenia ONKEYDOWN

```
linia(x,y,bok,kat,clBtnFace);
kat:=kat+3;
x:=x+dx;
y:=y+dy;
if kat >= 360 then kat:=0;
if (x+bok > prawy) or (x < 0) then dx:=-dx;
if (y+bok > dolny) or (y < 0) then dy:=-dy;
linia(x,y,bok,kat,ClRed);
```

ĆWICZENIA

1. Obracająca się linia zostawia za sobą kolorowe punkty (PUTPIXEL) - jedno lub wielokolorowe (RANDOM)
2. Linia z zadania 1 dodatkowo przemieszcza się po ekranie (animacja)
3. Linia z zadania 2 odbija się od brzegów ekranu (warunki logiczne)
4. Po ekranie porusza się kilka obracających się linii (tablica). Losowo dobieramy kolor, długość, prędkość i kierunek obrotu.
5. Wokół środka obraca się odcinek, a na jego końcach obracają się dodatkowe odcinki.



6. Kwadrat obraca się wokół swojego środka
7. Kwadrat obraca się wokół jednego ze swoich rogów
8. Kwadrat z zadania 6 i 7 porusza się i odbija od brzegów