**PODPROGRAMY – FUNKCJE I PROCEDURY**

Bywa często, że pewną sekwencję instrukcji w programie powtarzamy wiele razy. Można jednak uniknąć takiej sytuacji przez zastosowanie podprogramów. Polega to na tym, że fragment kodu – te naszą powtarzającą się sekwencję instrukcji - umieszczamy w podprogramie (funkcji lub procedurze).

Procedury tak jak funkcję powinno umieszczać się przed główną sekcją programy (BEGIN..END.).

Procedury deklaruje się w następujący sposób:

```
PROCEDURE NAZW-PROCEDURY (PARAMETRY);
VAR
        ZMIENNA : TYP;
BEGIN
        INSTRUKCJE
END;
```

Gdzie:

**Nazwa**_procedury – to dowolna nazwa, byle była unikalna w skal całego programu oraz spełniała standardowe założenia i wymagania (jak przy nazwach zmiennych).

**Parametry** – są to dane wprowadzane do procedury w momencie jej użycia. Procedura może nie potrzebować do działania parametrów przekazywanych z zewnątrz – wówczas ten element deklaracji procedury pomija się.

**Zmianna** – w procedurze można również umieścić sekcję z deklaracją zmiennych jak i stałych. Będą to jednak zmienne/stałe widoczne (możliwe do użycia) jedynie w procedurze – po za nią nie można ich użyć. To odróżnia stałe i zmienne deklarowane w programie w standardowej sekcji ze zmiennymi czy stałymi od tych stałych i zmiennych deklarowanych w procedurze. Jeśli procedura nie wymaga deklaracji swoich zmiennych i stałych te sekcję się pomija.

**Instrukcje** – sekwencja instrukcji, które procedura ma wykonać.

PRZYKŁADY PROCEDUR:

```
PROGRAM PRZYKŁAD1;
VAR
        A,B:INTEGER;
PROCEDURE DODAJ(X,Y:INTEGER);
BEGIN
        WRITELN('DODAJE:',X+Y);
END;

PROCEDURE POBIERZ_I_DODAJ;
VAR
        A1,A2:INTEGER;
BEGIN
        WRITELN('POBIERAM DANE');
        READLN(A1,A2);
        WRITELN('WYNIK',A1+A2);
END;

BEGIN
        WRITELN('PODAJ A I B');
        READLN(A,B);
        DODAJ(A,B);
        POBIERZ_I_DODAJ;
END.
```

```
PROGRAM PRZYKLAD2;
VAR R:REAL;
PROCEDURE DODAJE_I_MNOZE(A,B:INTEGER; X:REAL);
BEGIN
        WRITELN('WYNIK',(A+B)*X);
END;
BEGIN
        R:= 2.4;
        DODAJE_I_MNOZE(2, 4, R);
END.

PROGRAM PRZYKLD3;
VAR R:REAL;
PROCEDURE POLE_KOLA(PROMIEN:REAL);
CONST
        PI=3.14;
BEGIN
        WRITELN('POLE=',PI*PROMIEN*PROMIEN);
END;
BEGIN
        R:=34;
        OBWOD_KOLA(R);
        R:=2.4;
        OBWOD_KOLA(R);
END.
```

Zadanie do domu: WPISZ POWYŻSZE PRZYKŁADY DO KOMPILATORA I PRZEANALIZUJ KOD. SPRÓBÓJ SAMODZIELNIE ZMODYFIKOWAĆ KOD TAK BY ZAOBSERWOWAĆ JAKA JEST ZASADA DZIAŁANIA PROCEDUR.

**WAŻNE!!!!**

**JAK MOŻNA BYŁO ZAOBSERWOWAĆ, PROCEDURA MOŻE (NIE MUSI) PRZYJMOWAĆ PARAMETRY (WARTOŚCI PRZEKAZYWANE W TRAKCIE URUCHAMIANIA PROCEDURY Z ZEWNĄTRZ, PROCEDURA MOŻE TEŻ POSIADAĆ WŁASNE ZMIENNE I STAŁE.**

**PROCEDURA WYKONUJE PEWIEN CIĄG (SEKWENCJĘ) INSTRUKCJI I MOŻE BYĆ WIELOKROTNIE UŻYWANA W PROGRAMIE – PROCEDURA JEDNAK NIE MOŻE ZWRACAĆ WARTOŚCI.**

FUNKCJE WYGLADAJĄ PODOBNIE JAK PROCEDURY, ŻĄDZĄ SIĘ PODOBNYMI PRAWAMI I UZYWA SIĘ ICH BARDZO PODOBNIE – JEDNAK W PORÓWNANIU Z PROCEDURAMI MAJĄ JEDNĄ BARDZO DUŻĄ RÓŻNICE – MUSZĄ ZWRACAĆ WARTOŚĆ JAKO WYNIK SWOJEGO DZIAŁANIA.
Funkcje zwracają wartość przez swoją nazwę – w pewnym sensie można je traktować jak zmienne, do których jednak nie można przypisać wartości ale można z niej wartość 'wyjać'.

Sposób deklaracji funkcji:

```
FUNCTION NAZWA-FUNKCJI ( LISTA_PARAMETROW ) : TYP_WYNIKU;
VAR
        ZMIENNE:TYP;
BEGIN
        INSTRUKCJE;
        NAZWA_FUNKCJI := WARTOŚĆ_ZWARACANA;
END;
```

JAK WIDAĆ DEKLARACJA FUNKCJI NIEWIELE RÓŻNI SIĘ OD DEKLARACJI PROCEDURY – INNE JEST SŁOWO KLUCZOWE (ZAMIAST PROCEDURE – FUNCTION) OKREŚLAJĄCE FUNKCJE.

DODATKOWO KONIECZNE JEST OKREŚLENIE TYPU WYNIKU ZWRACANAEGO PRZEZ WYNKCJE ORAZ TRZEBA NA KONCU SEKWENCJI INSTRUKCJI WYKONYWANYCH PRZEZ FUNKCJE DODAJ INSTRUKCJE PRZYPISANIA NAZWIE FUNKCJI WARTOŚCI, KTÓRA MA BYĆ ZWRACANA.

PRZYKLADY UZYCIA FUNKCJI:

```
PROGRAM PRZYKLAD4;
VAR
        A,B,C:INTEGER;
FUNCTION DODAJ(A,B:INTEGER) : INTEGER;
BEGIN
        DODAJ := A + B;
END;
BEGIN
        WRITELN('PODAJ DANE');
        READLN(A,B);
        {UZYCIE FUNKCJI SKUTKUJE ZWRÓCENIEM PEWNEJ WARTOŚCI, ZATEM MOŻNA FUNKCJĘ
        PRZYPISAĆ DO ZMIENNEJ}
        C := DODAJ(3,5);
        WRITELN(C);
        {SKORO FUNKCJA ZWRACA WARTOSC, MOŻNA JEJ UZYĆ JAK ZMIENNEJ I WARTOŚĆ PRZEZ NIĄ
        ZWRACNĄ WYPISAĆ NA EKRAN}
        WRITELN(DODAJ(A,B));
END.
```

```
PROGRAM PRZYKLAD5;
VAR
        R,WYNIK:REAL;
FUNCTION POLE_KOLA(PROMIEN:REAL) : REAL;
CONST
        PI=3.14;
BEGIN
        POLE_KOLA:= PI*PROMIEN*PROMIEN;
END;
BEGIN
        WRITELN('PODAJ DANE');
        READLN(R);
        WYNIK := POLE_KOLA(R);
        WRITELN(WYNIK);
END.
```

**Różnica pomiędzy procedurą a funkcją jest wyraźna. Procedura wykonuje tylko pewien ciąg instrukcji – funkcja dodatkowo zwraca wartość określonego w definicji typu.**

## ZMIENNE GLOBALNE I LOKALNE

Jeśli chcemy by jakaś zmienna była dostępna we wszystkich procedurach i funkcjach jak również w głównym bloku programy (BEZGIN..END.) deklarujemy ją na samym początku programy (zgodnie z szkieletem programu, który był omawiany kilka tygodni temu). Taka zmienna otrzymuje miano **zmiennej globalnej**, ponieważ jej zasięg jest globalny – jest widoczna i dostępna w całym programie łącznie z jego procedurami i funkcjami.
Dodatkowo można deklarować zmienne w procedurach i funkcjach (zgodnie z wzorcem, który pojawił się wyżej) – wówczas taka zmienna dostępna jest tylko wewnątrz procedury czy funkcji, w której została zdeklarowana. Jej zasięg jest lokalny – ponieważ dostęp ma do niej tylko ta

procedura/funkcja, w której została zdeklarowana. Dlatego taka zmienna nosi miano **zmiennej lokalnej**.

Najlepiej zobrazuje to przykład:

```
PROGRAM PRZYKLAD6;

VAR     {deklaracja zmiennych o zasięgu globalnym}
        {te zmienne będą dostępne w każdym miejscu programu, również w procedurach i funkcjach}
        ZMIENNA_GLOBALNA1:INTEGER;
        ZMIENNA_GLOBALNA2:REAL;

PROCEDURE PROC1 (Z1:INTEGER);
VAR     {deklaracja zmiennej lokalnej – będzie ona dostępna jedynie wewnątrz ten procedury}
        ZMIENNA_LOKALNA1:INTEGER;
BEGIN
        ZMIENNA_LOKALNA1 := 2;          {można się odwołać do zmiennej lokalnej bo została
                                        zdefiniowana w tej procedurze}
        WRITELN(ZMIENNA_LOKALNA1+Z1);
        ZMIENNA_GLOBALNA2 := 2.2;       {można się odwołać do zmiennej globalnej – jest dostępna
                                        wszędzie}
END;

BEGIN
        ZMIENNA_GLOBALNA1 := 2;
        PROC1(ZMIENNA_GLOBALNA1);

        ZMIENNA_LOKALNA1 := 2;  {BŁĄD – nie można się odwołać do zmiennej lokalnej poza funkcją
                                lub procedurą, w której została zdeklarowana}
END.
```

**PARAMERY FORMALNE I AKTUALNE**

Często deklaruje się funkcje i procedury z parametrem – parametry te (ich identyfikatory) symbolizują dane wprowadzane do funkcji lub procedury. W trakcie deklaracji te parametry nie mają jeszcze żadnych wartości – jednak już teraz można wewnątrz procedury czy funkcji używać ich. Dzięki temu jesteśmy w stanie opisać **formę** (to co ma wykonywać i jak ma wykonywać) procedury czy funkcji.

DLATEGO PARAMETRY TE OKREŚLA SIĘ MIANEM PARAMETRÓW **FORMALNYCH**, ponieważ przy ich użyciu określa się form, w obrębie której zachodzić będą obliczenia wykonywane przez procedurę czy funkcję.

PARAMETRY AKTUALNE występują w momencie wywołania procedury lub funkcji – mają one już teraz konkretne wartości. Inne określenie parametrów aktualnych to parametry rzeczywiste.

ZATEM – WARTOŚCI PODSTAWIONE POD PARAMETRY FORMALNE W CHWILI UŻYCIA FUNKCJI LUB PROCEDURY TO PARAMETRY AKTUALNE.

**PRZEKAZYWANIE PARAMETRÓW O FUNKCJI I PROCEDUR**

W PASCAL ISTNIEJĄ DWIE METODY PRZEKAZYWANIA PARAMETRÓW DO FUNKCJI/PROCEDUR:
-PRZEZ WARTOŚĆ
-PRZEZ ZMIENNĄ

PRZYKŁD:

```
VAR
        X:INTEGER;      {ZMIENNA GLOBALNA}

PROCEDURE PROC1(X1:INTEGER);        {PRZEKAZANIE PARAMETRU PRZEZ WARTOŚĆ}
BEGIN
        X1 := X1 *2;    {ZMIANA PARAMETRU PRZEKAZANEGO PRZEZ WARTOŚĆ WEWNĄTRZ
                         FUNKCJI CZY PROCEDURY NIE POWODUJE JEGO ZMIANY POZA NIĄ}
        WRITELN(X1);
END;

PROCEDURE PROC2(VAR X1:INTEGER); {PRZEKAZANIE PARAMETRU PRZEZ ZMIENNA}
BEGIN
        X1 := X1*2;     {ZMIANA PARAMETRU PRZEKAZANEGO PRZEZ ZMIENNA WEWNĄTRZ
                         FUNKCJI CZY PROCEDURY POWODUJE JEGO ZMIANĘ POZA NIĄ}
        WRITELN(X1);
END;

BEGIN
        X:=2;
        PROC1(X);       {PRZEKAZANIE WARTOŚCI ZMIENNEJ X – POMIMO, ŻE PROCEDURA TA
                         ZMIENIA WARTOŚĆ PARAMETRU, TO DZIEKU TEMU, ŻE ZOSTAŁ ON
                         PRZEKAZANY PRZEZ WARTOŚĆ POZA PROCEDURĄ POZOSTAJE ON
                         NIEZMIENIONY}
        WRITELN(X);     {POJAWI SIĘ NA EKRANIE WARTOŚC 2}

        PROC2(X);       {PRZEKAZANIE ZMIENNEJ X –PROCEDURA TA
                         ZMIENIA WARTOŚĆ PARAMETRU, A DZIEKU TEMU, ŻE ZOSTAŁ ON
                         PRZEKAZANY PRZEZ ZMIENNA POZA PROCEDURĄ ZMIANY WPROWADZONE
                         PRZEZ PROCEDURE RÓWNIEŻ SĄ WIDOCZNE NIEZMIENIONY}
        WRITELN(X);     {POJAWI SIĘ NA EKRANIE WARTOŚC 4 – PROCEDURA ZMIENIŁA JEGO
                         WARTOŚĆ }
END.
```

JAK WIDAĆ JEŚLI ZMIENNA, KTÓREJ UZYLIŚMY ABY PRZEKAZAĆ JĄ DO PROCEDURY JAKO PARAMETR PRZEZ WARTOŚĆ MIMO, ŻE PROCEDURA ZMIANIA WARTOŚC PARAMETRU, TO WARTOŚĆ ZMIENNEJ, KTÓRA POSŁUZYŁA DO PRZEKAZANIA TEN WARTOŚCI POZOSTAJE NIETKNIĘTA.

JEŚLI JEDNAK PRZEKAZANIE MIAŁO MIEJSCE NIE PRZEZ WARTOŚĆ, A PRZEZ ZMIENNĄ TO WÓWCZAS, JEŚLI PROCEDURA DOKONA JAKIEJKOLWIEK ZMIANY NA PARAMETRZE TO ZMIANE ULEGNIE RÓWNIEŻ ZMIENNA, KTÓRA DO PRZEKAZANIA PARAMETRU POSŁUZYŁA.

MOŻNA POWIEDZIEĆ, ŻE PRZEKAZANIE PARAMETRU PRZEZ WARTOŚĆ POWODUJE UTWORZENIE NOWEJ (LOKALNEJ) KOPII ZMIENNEJ – DLATEGO ZMIANY NA NIEJ DOKONANE SĄ WIDOCZNE JEDYNIE WEWNĄTRZ PROCEDURY/FUNKCJI, KTÓRA ZMIANY DOKONAŁA.
JEŚLI ZAŚ PARAMETR PRZEKAZANO PRZEZ ZMIENNĄ, TO FUNKCJA/PROCEDURA OPERUJE NA TEN SAMEJ ZMIENNEJ, KTÓREJ UZYTO PODCZAS WYWOŁANIA PROC/FUNKC.