

Informatyka 1

Wykład V

Procedury i funkcje, struktura programu w Pascalu

*Robert Muszyński
ZPCiR IIAiR PWr*

Zagadnienia: deklaracje procedury i funkcji, parametry procedur i funkcji, reguły użycia parametrów VAR, podprogramy, struktura programu w Pascalu.

Copyright © 2001–2006 Robert Muszyński

Niniejszy dokument zawiera materiały do wykładu na temat podstaw programowania w językach wysokiego poziomu. Jest on udostępniony pod warunkiem wykorzystania wyłącznie do własnych, prywatnych potrzeb i może być kopiowany wyłącznie w całości, razem ze stroną tytułową.

Przykład procedury

```
PROCEDURE CzyscEkran(LiczbaLinii: INTEGER);  
BEGIN  
  FOR i := 1 TO LiczbaLinii DO  
    WRITELN;  
END;
```

Przykład procedury

```
PROCEDURE CzyscEkran(LiczbaLinii: INTEGER);  
BEGIN  
  FOR i := 1 TO LiczbaLinii DO  
    WRITELN;  
END;
```

Przykład wywołania:

```
CzyscEkran(32);
```

Przykład procedury

```
PROCEDURE CzyscEkran(LiczbaLinii: INTEGER);  
BEGIN  
  FOR i := 1 TO LiczbaLinii DO  
    WRITELN;  
END;
```

Przykład wywołania:

```
CzyscEkran(32);
```

Terminologia:

- parametry formalne — LiczbaLinii,

Przykład procedury

```
PROCEDURE CzyscEkran(LiczbaLinii: INTEGER);  
BEGIN  
  FOR i := 1 TO LiczbaLinii DO  
    WRITELN;  
END;
```

Przykład wywołania:

```
CzyscEkran(32);
```

Terminologia:

- parametry formalne — LiczbaLinii,
- parametry aktualne — 32.

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```


Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Terminologia:

- parametry formalne — a, b,

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Terminologia:

- parametry formalne — a, b,
- parametry aktualne — *WiekTaty, WiekMamy,*

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Terminologia:

- parametry formalne — a, b,
- parametry aktualne — WiekTaty, WiekMamy,
- **zmienne globalne** — Starszy, WiekTaty, WiekMamy,

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Terminologia:

- parametry formalne — a, b,
- parametry aktualne — WiekTaty, WiekMamy,
- zmienne globalne — Starszy, WiekTaty, WiekMamy,
- **zmienne lokalne** — tmp,

Przykłady funkcji

```
FUNCTION Max(a,b:INTEGER):INTEGER;  
BEGIN  
  IF a > b THEN  
    Max := a  
  ELSE  
    Max := b  
END;
```

Przykład wywołania:

```
Starszy := Max(WiekTaty,WiekMamy);
```

```
FUNCTION Porz(a,b:INTEGER):INTEGER;  
VAR tmp : INTEGER;  
BEGIN  
  IF a > b THEN  
    BEGIN  
      tmp := a;  
      a := b;  
      b := tmp;  
    END;  
END; {Porz}
```

Źle!

Terminologia:

- parametry formalne — a, b,
- parametry aktualne — WiekTaty, WiekMamy,
- zmienne globalne — Starszy, WiekTaty, WiekMamy,
- **zmienne lokalne** — tmp, a, b.

Dalsze przykłady procedur i funkcji

```
FUNCTION JestPierwsza(N: INTEGER): BOOLEAN;
{ wersja pierwsza -- pojedyncza funkcja }
VAR Dzielnik : INTEGER;
    JestP      : BOOLEAN;
BEGIN
    JestP := TRUE;
    Dzielnik := 2;
    WHILE JestP AND (Dzielnik <= (N DIV 2)) DO
        IF (N MOD Dzielnik) = 0
            THEN JestP := FALSE
            ELSE Dzielnik := Dzielnik + 1;
    JestPierwsza := JestP
END; { JestPierwsza }
```

Dalsze przykłady procedur i funkcji

```
FUNCTION JestPierwsza(N: INTEGER): BOOLEAN;  
{ wersja pierwsza -- pojedyncza funkcja }  
VAR Dzielnik : INTEGER;  
    JestP      : BOOLEAN;  
BEGIN  
    JestP := TRUE;  
    Dzielnik := 2;  
    WHILE JestP AND (Dzielnik <= (N DIV 2)) DO  
        IF (N MOD Dzielnik) = 0  
            THEN JestP := FALSE  
            ELSE Dzielnik := Dzielnik + 1;  
    JestPierwsza := JestP  
END; { JestPierwsza }
```

Przykład wywołania

```
zmienna := JestPierwsza(13+delta);
```


Jeszcze dalsze przykłady procedur i funkcji

```
FUNCTION Dzieli(Dzielnik, Dzielna: INTEGER): BOOLEAN;  
BEGIN  
  Dzieli := (Dzielna MOD Dzielnik) = 0  
END; { Dzieli }
```

```
FUNCTION JestPierwsza(N: INTEGER): BOOLEAN;  
{ wersja druga -- z rozbiciem na dwie funkcje }  
VAR Dzielnik : INTEGER;  
    JestP      : BOOLEAN;  
BEGIN  
  JestP := TRUE;  
  Dzielnik := 2;  
  WHILE JestP AND (Dzielnik <= (N DIV 2)) DO  
    IF Dzieli(Dzielnik,N) THEN  
      JestP := FALSE  
    ELSE  
      Dzielnik := Dzielnik + 1;  
  JestPierwsza := JestP  
END; { JestPierwsza }
```

```
PROGRAM pierwsze(INPUT,OUTPUT);
VAR A,B: INTEGER;
        { deklaracje procedur i funkcji }
PROCEDURE DrukujPierwsze(N, M: INTEGER);
VAR I: INTEGER;
BEGIN
  FOR I := N TO M DO
    IF JestPierwsza(I) THEN
      WRITELN('Znaleziona liczba pierwsza: ', I)
END; { DrukujPierwsze }
        { program glowny }
BEGIN
  WRITELN('Wyliczanie liczb pierwszych w przedziale.');
```

WRITELN('Podaj poczatek przedzialu:');

READLN(A);

WRITELN('Podaj koniec przedzialu:');

READLN(B);

IF A <= B

THEN DrukujPierwsze(A,B)

ELSE WRITELN('Blednie podany przedzial:',A,'>',B);

END.

Parametry procedur i funkcji

```
PROCEDURE Porz(a,b: INTEGER);  
VAR tmp : INTEGER;  
BEGIN  
{1}   IF a > b THEN  
      BEGIN  
{2}     tmp := a;  
{3}     a := b;  
{4}     b := tmp;  
      END  
END; {Porz}
```

Parametry procedur i funkcji

```
PROCEDURE Porz(a,b: INTEGER);  
VAR tmp : INTEGER;  
BEGIN  
{1}   IF a > b THEN  
      BEGIN  
{2}     tmp := a;  
{3}     a := b;  
{4}     b := tmp;  
      END  
END; {Porz}
```

```
{1} x := 5;  
{2} y := 3;  
{3} Porz(x,y);  
{4} IF x > y THEN  
{5}   WRITELN('wartosci nieuporzadkowane');
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```


Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		Wywołanie procedury

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		
						Wywołanie procedury TRUE

Porz:1

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		
Porz:1						
Porz:2					5	
						Wywołanie procedury TRUE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		
Porz:1						
Porz:2					5	
Porz:3			3			
						Wywołanie procedury TRUE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		
Porz:1						
Porz:2					5	
Porz:3			3			
Porz:4				5		
						Wywołanie procedury TRUE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		Wywołanie procedury TRUE
Porz:1						
Porz:2					5	
Porz:3			3			
Porz:4				5		
4			?	?	?	TRUE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry procedur i funkcji

```

PROCEDURE Porz(a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x	y	a	b	tmp	Uwagi
	?	?	?	?	?	
1	5					
2		3				
3			5	3		Wywołanie procedury TRUE
Porz:1						
Porz:2					5	
Porz:3			3			
Porz:4				5		
4			?	?	?	TRUE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

Parametry przekazywane przez wartość

Parametry procedur i funkcji

```
PROCEDURE Porz(VAR a,b: INTEGER);  
VAR tmp : INTEGER;  
BEGIN  
{1}   IF a > b THEN  
      BEGIN  
{2}     tmp := a;  
{3}     a := b;  
{4}     b := tmp;  
      END  
END;  
{Porz}
```


Parametry procedur i funkcji

```
PROCEDURE Porz(VAR a,b: INTEGER);  
VAR tmp : INTEGER;  
BEGIN  
{1}   IF a > b THEN  
      BEGIN  
{2}     tmp := a;  
{3}     a := b;  
{4}     b := tmp;  
      END  
END; {Porz}
```

```
{1} x := 5;  
{2} y := 3;  
{3} Porz(x,y);  
{4} IF x > y THEN  
{5}   WRITELN('wartosci nieuporzadkowane');
```

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x	y	tmp	Uwagi

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x	y	tmp	Uwagi
	?	?	?	

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1}  x := 5;
{2}  y := 3;
{3}  Porz(x,y);
{4}  IF x > y THEN
{5}    Writeln('wartosci nieuporzadkowane');
    
```

nr linii	x	y	tmp	Uwagi
1	5	?	?	

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x	y	tmp	Uwagi
	?	?	?	
1	5			
2		3		

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	$x \equiv a$	$y \equiv b$	tmp	Uwagi
	?	?	?	
1	5			
2		3		
3				Wywołanie procedury

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			
2		3		
3				
	Porz:1			Wywołanie procedury TRUE

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			
2		3		
3				
Porz:1				Wywołanie procedury TRUE
Porz:2			5	

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			Wywołanie procedury TRUE
2		3		
3				
Porz:1			5	
Porz:2				
Porz:3	3			

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			Wywołanie procedury TRUE
2		3		
3				
Porz:1 Porz:2 Porz:3 Porz:4	3	5	5	

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   WRITELN('wartosci nieuporzadkowane');
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			Wywołanie procedury TRUE
2		3		
3				
Porz:1			5	
Porz:2				
Porz:3	3			
Porz:4		5		
4			?	FALSE

Parametry procedur i funkcji

```

PROCEDURE Porz(VAR a,b: INTEGER);
VAR tmp : INTEGER;
BEGIN
{1}   IF a > b THEN
      BEGIN
{2}     tmp := a;
{3}     a := b;
{4}     b := tmp;
      END
END; {Porz}
    
```

nr linii	x≡a	y≡b	tmp	Uwagi
	?	?	?	
1	5			Wywołanie procedury TRUE
2		3		
3				
Porz:1			5	
Porz:2				
Porz:3	3			
Porz:4		5		
4			?	FALSE

```

{1} x := 5;
{2} y := 3;
{3} Porz(x,y);
{4} IF x > y THEN
{5}   Writeln('wartosci nieuporzadkowane');
    
```

Parametry przekazywane przez zmienną

Reguły użycia parametrów VAR

- (a) Parametry formalne mogą być dowolnych typów, dowolnie przemieszane, i każdy może być VAR. Jeśli nie jest VAR, to jest zwykłym parametrem.

Reguły użycia parametrów VAR

- (a) Parametry formalne mogą być dowolnych typów, dowolnie przemieszane, i każdy może być VAR. Jeśli nie jest VAR, to jest zwykłym parametrem.
- (b) Jeśli dany parametr formalny jest VAR, to odpowiadający mu parametr aktualny musi być zmienną, a nie dowolnym wyrażeniem, jak dla zwykłych parametrów.

Reguły użycia parametrów VAR

- (a) Parametry formalne mogą być dowolnych typów, dowolnie przemieszane, i każdy może być VAR. Jeśli nie jest VAR, to jest zwykłym parametrem.
- (b) Jeśli dany parametr formalny jest VAR, to odpowiadający mu parametr aktualny musi być zmienną, a nie dowolnym wyrażeniem, jak dla zwykłych parametrów.
- (c) Jeśli dany parametr formalny jest VAR, to odpowiadający mu parametr aktualny jest utożsamiany z parametrem formalnym — jest jego aliasem. Oba te parametry odwołują się do tej samej komórki pamięci.

Reguły użycia parametrów VAR

- (a) Parametry formalne mogą być dowolnych typów, dowolnie przemieszane, i każdy może być VAR. Jeśli nie jest VAR, to jest zwykłym parametrem.
- (b) Jeśli dany parametr formalny jest VAR, to odpowiadający mu parametr aktualny musi być zmienną, a nie dowolnym wyrażeniem, jak dla zwykłych parametrów.
- (c) Jeśli dany parametr formalny jest VAR, to odpowiadający mu parametr aktualny jest utożsamiany z parametrem formalnym — jest jego aliasem. Oba te parametry odwołują się do tej samej komórki pamięci.

Używać tylko tam gdzie są niezbędne

Przykład użycia parametrów VAR

```
PROCEDURE Podstaw2(VAR X1, X2: REAL;  
                   Val1, Val2: REAL);  
  
BEGIN  
  X1 := Val1;  
  X2 := Val2;  
END;
```

Przykład użycia parametrów VAR

```
PROCEDURE Podstaw2(VAR X1, X2: REAL;  
                    Val1, Val2: REAL);  
  
BEGIN  
    X1 := Val1;  
    X2 := Val2;  
END;
```

Przykład wywołania:

```
Podstaw2(pierw1,pierw2,(-b-sqdelta)/(2.0*a),(-b+sqdelta)/(2.0*a));
```

Przykład użycia parametrów VAR

```
PROCEDURE Podstaw2(VAR X1, X2: REAL;  
                    Val1, Val2: REAL);  
  
BEGIN  
    X1 := Val1;  
    X2 := Val2;  
END;
```

Przykład wywołania:

```
Podstaw2(pierw1,pierw2,(-b-sqdelta)/(2.0*a),(-b+sqdelta)/(2.0*a));
```

Porównaj:

```
Podstaw2(pierw1,pierw2,pierw1+pierw2,pierw1*pierw2);
```

Przykład użycia parametrów VAR

```
PROCEDURE Podstaw2(VAR X1, X2: REAL;  
                    Val1, Val2: REAL);  
  
BEGIN  
    X1 := Val1;  
    X2 := Val2;  
END;
```

Przykład wywołania:

```
Podstaw2(pierw1,pierw2,(-b-sqdelta)/(2.0*a),(-b+sqdelta)/(2.0*a));
```

Porównaj:

```
Podstaw2(pierw1,pierw2,pierw1+pierw2,pierw1*pierw2);
```

```
Podstaw2(pierw1+pierw2,pierw1*pierw2,pierw1,pierw2);
```

Przykład użycia parametrów VAR

```
PROCEDURE Podstaw2(VAR X1, X2: REAL;
                    Val1, Val2: REAL);
BEGIN
  X1 := Val1;
  X2 := Val2;
END;
```

Przykład wywołania:

```
Podstaw2(pierw1,pierw2,(-b-sqdelta)/(2.0*a),(-b+sqdelta)/(2.0*a));
```

Porównaj:

```
Podstaw2(pierw1,pierw2,pierw1+pierw2,pierw1*pierw2);
```

```
Podstaw2(pierw1+pierw2,pierw1*pierw2,pierw1,pierw2);
```

czy nawet:

```
Podstaw2(pierw1,pierw1,pierw1,pierw1);
```

Deklaracje procedur i funkcji

```
FUNCTION F(X:INTEGER): INTEGER; FORWARD;
```

Deklaracje procedur i funkcji

```
FUNCTION F(X:INTEGER): INTEGER; FORWARD;  
FUNCTION G(X:INTEGER): INTEGER;  
VAR ...           { deklaracja zmiennych }  
BEGIN  
    ... F(A)     ... { sekcja operacyjna }  
END;
```

Deklaracje procedur i funkcji

```
FUNCTION F(X:INTEGER): INTEGER; FORWARD;
FUNCTION G(X:INTEGER): INTEGER;
VAR ...           { deklaracja zmiennych }
BEGIN
    ... F(A)     ... { sekcja operacyjna }
END;
FUNCTION F;
VAR ...           { deklaracja zmiennych }
BEGIN
    ... G(B)     ... { sekcja operacyjna }
END;
```


Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych

Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych
- Zmienne globalne a lokalne

Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych
- Zmienne globalne a lokalne
- Przesłanianie zmiennych

Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych
- Zmienne globalne a lokalne
- Przesłanianie zmiennych
- Parametry formalne a aktualne

Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych
- Zmienne globalne a lokalne
- Przesłanianie zmiennych
- Parametry formalne a aktualne
- Zgodność typu parametrów formalnych i aktualnych

Procedury i funkcje — zagadnienia

- Zakres leksykalny zmiennych
- Zmienne globalne a lokalne
- Przesłanianie zmiennych
- Parametry formalne a aktualne
- Zgodność typu parametrów formalnych i aktualnych
- Przekazywanie parametrów przez wartość
- Przekazywanie parametrów przez zmienną — aliasy

Procedury i funkcje — podsumowanie

Dekompozycja problemu: podział problemu na wiele mniejszych i wydzielanie rozwiązań tych mniejszych problemów od rozwiązania problemu głównego jest drogą do zredukowania wielkości i stopnia komplikacji problemu.

Procedury i funkcje — podsumowanie

Dekompozycja problemu: podział problemu na wiele mniejszych i wydzielanie rozwiązań tych mniejszych problemów od rozwiązania problemu głównego jest drogą do zredukowania wielkości i stopnia komplikacji problemu.

Czytelność i przejrzystość programu: niezależnie od stopnia złożoności problemu, podział na procedury i funkcje jest drogą do zwiększenia czytelności programu, co zawsze jest celowe. Jednakże dodawanie procedur nie jest celem samym w sobie.

Procedury i funkcje — podsumowanie

Dekompozycja problemu: podział problemu na wiele mniejszych i wydzielanie rozwiązań tych mniejszych problemów od rozwiązania problemu głównego jest drogą do zredukowania wielkości i stopnia komplikacji problemu.

Czytelność i przejrzystość programu: niezależnie od stopnia złożoności problemu, podział na procedury i funkcje jest drogą do zwiększenia czytelności programu, co zawsze jest celowe. Jednakże dodawanie procedur nie jest celem samym w sobie.

Unikanie powtórzeń: często warto wyodrębnić w procedurę/funkcję powtarzający się zestaw instrukcji, wyrażeń, bądź schemat obliczeniowy.

Procedury i funkcje — podsumowanie

Dekompozycja problemu: podział problemu na wiele mniejszych i wydzielanie rozwiązań tych mniejszych problemów od rozwiązania problemu głównego jest drogą do zredukowania wielkości i stopnia komplikacji problemu.

Czytelność i przejrzystość programu: niezależnie od stopnia złożoności problemu, podział na procedury i funkcje jest drogą do zwiększenia czytelności programu, co zawsze jest celowe. Jednakże dodawanie procedur nie jest celem samym w sobie.

Unikanie powtórzeń: często warto wyodrębnić w procedurę/funkcję powtarzający się zestaw instrukcji, wyrażeń, bądź schemat obliczeniowy.

Zasada lokalności: określa, że wszystkie elementy mające ze sobą związek powinny znaleźć się jak najbliżej siebie w programie.

Procedury i funkcje — podsumowanie

Dekompozycja problemu: podział problemu na wiele mniejszych i wydzielenie rozwiązań tych mniejszych problemów od rozwiązania problemu głównego jest drogą do zredukowania wielkości i stopnia komplikacji problemu.

Czytelność i przejrzystość programu: niezależnie od stopnia złożoności problemu, podział na procedury i funkcje jest drogą do zwiększenia czytelności programu, co zawsze jest celowe. Jednakże dodawanie procedur nie jest celem samym w sobie.

Unikanie powtórzeń: często warto wyodrębnić w procedurę/funkcję powtarzający się zestaw instrukcji, wyrażeń, bądź schemat obliczeniowy.

Zasada lokalności: określa, że wszystkie elementy mające ze sobą związek powinny znaleźć się jak najbliżej siebie w programie.

Wielkość procedur/funkcji: niezbyt duże, np. < 50 linii, ważna jest też spójność.

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                  [ "(" parametry-programu ")" ].
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.  
blok-programu = blok
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.  
blok-programu = blok  
blok = sekcja-deklaracji-etykiet  
      sekcja-deklaracji-stalych  
      sekcja-definicji-typow  
      sekcja-deklaracji-zmiennych  
      sekcja-deklaracji-procedur-i-funkcji  
      sekcja-operacyjna.
```


Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.  
blok-programu = blok  
blok = sekcja-deklaracji-etykiet  
      sekcja-deklaracji-stalych  
      sekcja-definicji-typow  
      sekcja-deklaracji-zmiennych  
      sekcja-deklaracji-procedur-i-funkcji  
      sekcja-operacyjna.  
sekcja-deklaracji-procedur-i-funkcji =  
  { ( deklaracja-procedury | deklaracja-funkcji) ";" }.
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.  
blok-programu = blok  
blok = sekcja-deklaracji-etykiet  
      sekcja-deklaracji-stalych  
      sekcja-definicji-typow  
      sekcja-deklaracji-zmiennych  
      sekcja-deklaracji-procedur-i-funkcji  
      sekcja-operacyjna.  
sekcja-deklaracji-procedur-i-funkcji =  
  { ( deklaracja-procedury | deklaracja-funkcji) ";" }.  
sekcja-operacyjna = instrukcja-zlozona.
```

Struktura programu w Pascalu

```
program = naglowek-programu ";" blok-programu ".".  
naglowek-programu = "PROGRAM" identyfikator  
                [ "(" parametry-programu ")" ].  
parametry-programu = lista-identyfikatorow.  
lista-identyfikatorow = identyfikator {"," identyfikator}.  
blok-programu = blok  
blok = sekcja-deklaracji-etykiet  
      sekcja-deklaracji-stalych  
      sekcja-definicji-typow  
      sekcja-deklaracji-zmiennych  
      sekcja-deklaracji-procedur-i-funkcji  
      sekcja-operacyjna.  
sekcja-deklaracji-procedur-i-funkcji =  
  { ( deklaracja-procedury | deklaracja-funkcji) ";" }.  
sekcja-operacyjna = instrukcja-zlozona.  
instrukcja-zlozona = "BEGIN" ciag-instrukcji "END".
```

Zakres leksykalny i przestąnianie zmiennych

— przykłady

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

Zakres leksykalny i przestanianie zmiennych

— przykłady

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

Zakres leksykalny i przestąnianie zmiennych

— przykłady

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

Zakres leksykalny i przestanianie zmiennych

— przykłady

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR x : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
BEGIN  
    x := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR z : INTEGER);  
BEGIN  
    z := 13;  
END;  
BEGIN  
    x := 21;  podstaw(x);  ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```



```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR z : INTEGER);  
VAR x: INTEGER;  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (z : INTEGER);  
  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

```
PROGRAM podstawianie;  
VAR x: INTEGER;  
  
PROCEDURE podstaw (VAR z : INTEGER);  
  
BEGIN  
    z := 13;  x := 14;  
END;  
BEGIN  
    x := 21;  podstaw(x); ...
```

Deklaracje i wywołania procedur — przykłady

```
PROGRAM przyklad (OUTPUT);
VAR x,y,v,z : INTEGER;

PROCEDURE Proc(VAR a,b : INTEGER; c,d : INTEGER);
BEGIN
    b:=2*b; c:=2*c; a:=b+c; d:=b;
    WRITELN('a=',a, 'b=',b, 'c=',c, 'd=',d)
END;

BEGIN
    x:=1; y:=10; v:=100; z:=1000;
    Proc(x,y,v,z);
    WRITELN('x=',x, 'y=',y, 'v=',v, 'z=',z);...
```

```
FUNCTION nsum(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a+b; nsum:=a  
END;
```

```
FUNCTION ndif(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a-b; ndif:=a  
END;
```

```
FUNCTION nsum(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a+b; nsum:=a  
END;
```

```
FUNCTION ndif(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a-b; ndif:=a  
END;
```

```
nsum(5,3,2) + ndif(10,2,3) = 15;
```

```
FUNCTION nsum(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a+b; nsum:=a  
END;
```

```
FUNCTION ndif(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a-b; ndif:=a  
END;
```

```
nsum(5,3,2) + ndif(10,2,3) = 15;
```

```
nsum(4,1,nsum(2,2,2)) = 8;
```

```
FUNCTION nsum(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a+b; nsum:=a  
END;
```

```
FUNCTION ndif(a,b,n : INTEGER): INTEGER;  
VAR i : INTEGER;  
BEGIN  
  FOR i := 1 TO n DO a:=a-b; ndif:=a  
END;
```

```
nsum(5,3,2) + ndif(10,2,3) = 15;
```

```
nsum(4,1,nsum(2,2,2)) = 8;
```

```
ndif(a,b,c) - ndif(d,b,e) = ndif(a-d,b,c-e);
```



```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);  
    Zamien(x,1,y,2);
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);  
    Zamien(x,1,y,2);  
    Zamien(x,y,x,y);
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);  
    Zamien(x,1,y,2);  
    Zamien(x,y,x,y);  
    Zamien(x,x,x,x);
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);  
    Zamien(x,1,y,2);  
    Zamien(x,y,x,y);  
    Zamien(x,x,x,x);  
    Zamien(x,y,x+y,y);
```

```
PROGRAM przyklad;  
VAR x,y : INTEGER;  
  
PROCEDURE Zamien(VAR a,b:INTEGER; c,d:INTEGER);  
BEGIN ... END;  
BEGIN  
    Zamien(x,y,2,1);  
    Zamien(x,1,y,2);  
    Zamien(x,y,x,y);  
    Zamien(x,x,x,x);  
    Zamien(x,y,x+y,y);  
    Zamien(x+y,y,x,y);
```

```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;
```



```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;  
BEGIN  
    Suma(x,y);
```

```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;  
BEGIN  
    Suma(x,y);  
    x := Suma(Trunc(Sin(x)),4);
```

```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;  
BEGIN  
    Suma(x,y);  
    x := Suma(Trunc(Sin(x)),4);  
    x := Suma(Suma(x,y),z);
```

```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;  
BEGIN  
    Suma(x,y);  
    x := Suma(Trunc(Sin(x)),4);  
    x := Suma(Suma(x,y),z);  
    z := Suma(x/y, y div x);
```

```
PROGRAM przyklad;  
VAR x,y,z : INTEGER;  
  
FUNCTION Suma( sk11, sk12 :INTEGER): INTEGER;  
BEGIN ... END;  
BEGIN  
    Suma(x,y);  
    x := Suma(Trunc(Sin(x)),4);  
    x := Suma(Suma(x,y),z);  
    z := Suma(x/y, y div x);  
    if Suma(x,y)+z > 0 then z := 0;
```

Indeks

- Przykład procedury
- Przykłady funkcji
- Dalsze przykłady procedur i funkcji
- Jeszcze dalsze przykłady procedur i funkcji
- Parametry procedur i funkcji
- Reguły użycia parametrów VAR
- Przykład użycia parametrów VAR
- Deklaracje procedur i funkcji
- Procedury i funkcje — zagadnienia
- Procedury i funkcje — podsumowanie
- Struktura programu w Pascalu
- Zakres leksykalny i przestanianie zmiennych — przykłady
- Deklaracje i wywołania procedur — przykłady