Vícedimenzionální pole (1)

- Složky pole mohou být i strukturované
- Je-li jejich typem opět typ pole, pak se původní pole nazývá vícedimenzionálním
- Deklarace může být zapsána takto:
  - var m:array[a..b] of array[c..d] of T;
- Potom m[i][j] označuje složku j (typu T) složky i proměnné m

Vícedimenzionální pole (2)

- Většinou však používáme zkrácený zápis:
  - var m:array[a..b, c..d] of T; (pro deklaraci)
  - m[i, j] (pro zpřístupnění položky s indexy i, j)
- Dvoudimenzionální pole se nazývá matice
- Je-li:

```
type matice = array[1..n, 1..n] of real;
var m:matice;
```

Vícedimenzionální pole (3)

pak:

m[i] označuje proměnnou typu array[1..n] of real, která je i-tou složkou proměnné m (i-tým řádkem matice m) a m[i, j] označuje proměnnou typu real, která je j-tou složkou proměnné m[i] (prvek matice m, který leží v i-tém řádku a j-tém sloupci)

Vícedimenzionální pole (4)

- Deklarace var a:array[1..m, 1..n] of real pak tedy značí matici tvaru:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

kde a_{ij} jsou hodnoty typu real

Vícedimenzionální pole (5)

- Při práci s maticí je možné využít jednodušší manipulace s řádky:
  Př.: vynulování matice:

```
var a:array[1..n, 1..n] of real;

for i := 1 to n do a[1, i] := 0;
for i := 2 to n do a[i] := a[1];
```

Vícedimenzionální pole (6)

- Jsou-li proměnné x, y téhož (identického) typu pole (i vícedimenzionálního), pak je možné použít přiřazení:

  x := y;

- Pozn.: výše uvedenou úvahu o zavedení dvoudimenzionálního pole můžeme dále zobecnit a získat tak pole o n dimenzích
  - např.: var a:array[1..n_1, 1..n_2, 1..n_3] of real;

Podprogramy

- Podprogram je posloupnost příkazů opatřená jménem, které je možné později používat jako reprezentanta celé posloupnosti
- Jako podprogram je vhodné označit algoritmicky ucelenou programovou jednotku řešící vhodně vyčleněný problém
- Podprogramy v jazyce Pascal:
  - procedury
  - funkce

Blok (1)

- Příklad
  - Algoritmus řešení problému určení d=NSD(a,b) má tvar složeného příkazu:

```
begin
   repeat
      a := a mod b;
      pom := a; a := b; b := pom; (* výměna a, b *)
   until b = 0;
   d := a;
end;
```

Blok (2)

- a, b: proměnné odpovídající vstupním údajům
- d: proměnná odpovídající výstupnímu údaji
- pom: proměnná, která má význam pouze ve výše uvedeném příkazu. Říkáme, že je v něm lokální (rozsahem platnosti proměnné pom je výše uvedený příkaz)

- Obecně mohou být objekty, se kterými libovolný příkaz pracuje: návěští, konstanty, datové typy, proměnné, procedury a funkce

Blok (3)

- Vyčleníme-li ty objekty, které jsou v příkazu lokální a předřadíme-li jejich deklaraci popř. definici tomuto příkazu získáme tzv. blok
- Blok je tedy tvořen deklarační a příkazovou částí

Blok (4)

- V našem případě:

```
var pom:integer;
begin
    repeat
        a := a mod b;
        pom := a; a := b; b := pom; (* výměna a,b *)
    until b = 0;
    d := a;
end;
```

Blok (5)

- Pascal nedovoluje používat blok samostatně
- Blok v Pascalu musí být opatřen hlavičkou, která mu dává jméno a popř. seznam parametrů prostřednictvím nichž komunikuje se svým okolím
- Hlavičkou může být hlavička programu (nemůže obsahovat seznam parametrů) nebo hlavička podprogramu

Procedury (1)

- Deklarace:

```
procedure Jmeno (seznam formálních
                 parametrů s jejich typy);
Část deklarací a definic (lokální objekty)
begin
    příkazová část
end;
```

Hlavička

Blok

Procedury (2)

- Vyvolání (použití):

  Jmeno (seznam skutečných parametrů);
- Poznámka:

  Seznam formálních parametrů s jejich typy a tím i následně seznam skutečných parametrů mohou být prázdné. V tom případě neuvádíme ani kulaté závorky

Funkce (1)

- Deklarace:

```
function Jmeno (seznam formálních
        parametrů s jejich typy):typ_funkce;
Část deklarací a definic (lokální objekty)
begin
    příkazová část
    (obsahující alespoň jedno přiřazení
     tvaru Jmeno:=výraz;)
end;
```

Hlavička

Blok

Funkce (2)

- Vyvolání (použití):

  Jmeno (seznam skutečných parametrů)
- Poznámka:

  Vzhledem k tomu, že funkce nese hodnotu, bývá většinou volána jako součást výrazu (např. na pravé straně přiřazovacího příkazu, jako parametr příkazu write apod.)

Formální a skutečné parametry (1)

- Seznam formálních parametrů udává hlavička deklarace procedury nebo funkce
- Seznam skutečných parametrů je součástí příkazu procedury (nebo zápisu funkce)
- Vzájemná korespondence skutečných a formálních parametrů je dána pořadím v těchto seznamech, které se co do počtu musí shodovat a odpovídající si parametry musí být typově kompatibilní

Formální a skutečné parametry (2)

- Parametry (formální i skutečné) lze rozdělit do dvou skupin:
  - vstupní: odpovídají vstupním údajům
  - výstupní: odpovídají výstupním údajům
- Tomuto rozdělení odpovídá i volba jednoho ze dvou základních druhů substitucí skutečného parametru za formální:
  - volání hodnotou (parametr - hodnota)
  - volání odkazem, referencí (parametr - proměnná)