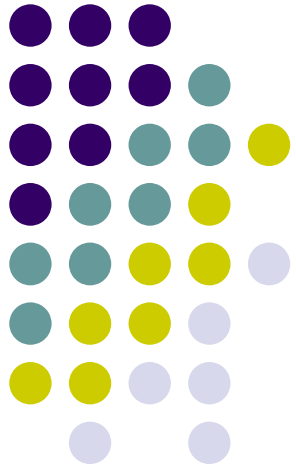
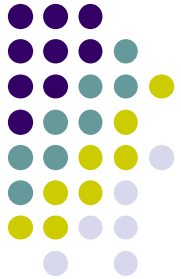
PODPROGRAMY

1. PROCEDURY
2. FUNKCE

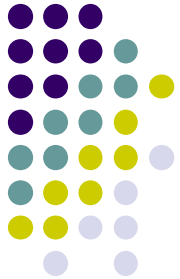


Podprogramy=procedury Funkce



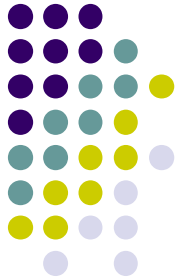
- Sekvence příkazů, části kódu, které se v programu opakují je lepší zapsat jen jednou a v programu je v případě potřeby vyvolávat
- Důvody:
  - Opakování – narůstání kódu do délky
  - Horší opravy – na všech místech programu
  - Zvyšování přehlednosti
  - Snížení chybovosti

Procedury a funkce



- Jsou to logicky uzavřené programové celky (posloupnosti instrukcí), které opakovaně v programu voláme jejich jménem (i s parametry)
- Protože podprogramy/funkce mohou být volán v hlavním programu několikrát pokaždé s jinými parametry (hodnotami), musí být jejich deklarace obecná
- Jediný rozdíl mezi procedurou a funkcí je ten, že funkce VŽDYCKY vrací hodnotu, kdežto procedura nikoli

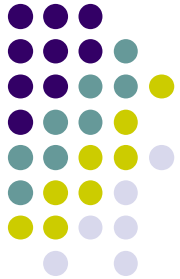
FUNKCE
do hlavního programu vrací hodnotu



Struktura:

- Jméno funkce a parametry (jejich datový typ a datový typ návratové hodnoty)
- Seznam proměnných
- Tělo funkce

Příklad funkce Jarda, která umí sčítat dvě čísla



```
Function JARDA(a,b:integer):integer;
```

vstupní parametry typu celá čísla

Datový typ výsledné hodnoty

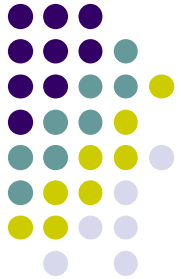
```
Var
```

zde bychom mohli definovat lokální proměnné, které by případně funkce potřebovala k výpočtům

```
Begin;
Jarda:=a+b;
End;
```

tělo funkce

Příklad použití funkce Jarda v hlavnim programu



- Volání funkce z hlavního programu:

.....

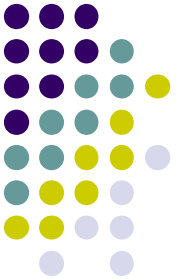
C:=25;

D:=41;

Vysl:=JARDA(a,b)

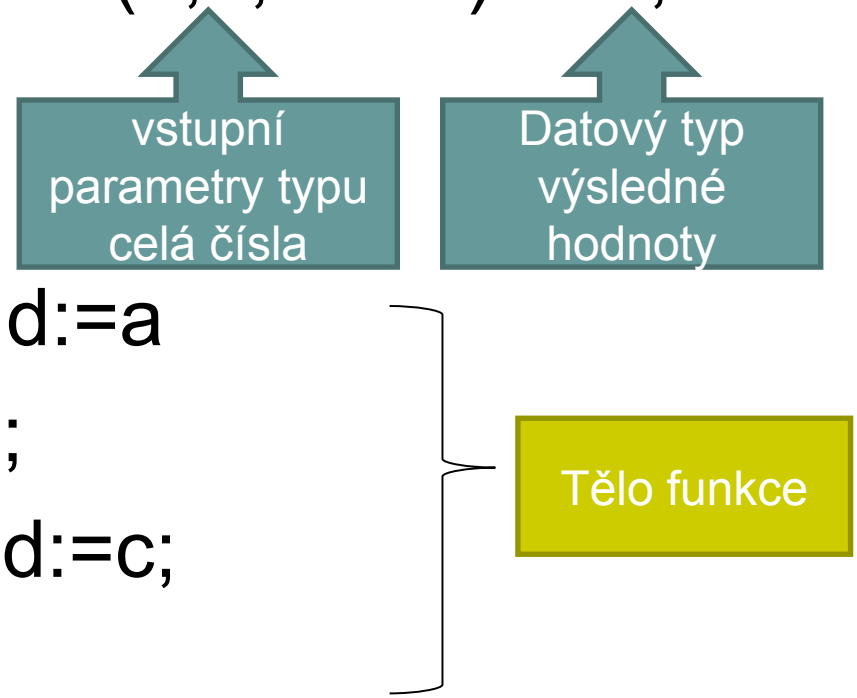
Nebo lze i rovnou zapsat Vysl:=JARDA(25,41)

Příklad funkce MAX, která umí určit maximum ze tří čísel

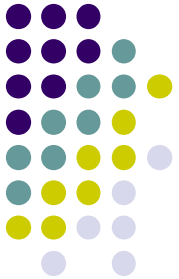


- Function MAX(a,b,c:real):real;

```
Var d:real;
Begin
   if a>b then d:=a
      else d:=b;
   if c>d then d:=c;
   MAX:=d;
End.
```



Příklad použití funkce MAX v programu



- Program

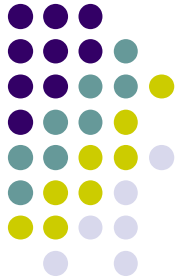
```
Var U,V,W : real;

.

…..
V:=5.5;
W:=1.5;

…..
U:=3*MAX(V, 17.5 , 3*W) +1
```

Význam funkce



- Dostane vstupní parametry
- Provede výpočet
- **Vrátí jednu vypočtenou hodnotu**
  - velké množství předdefinovaných funkcí (např. IntToStr(), Val(), SQR(), Abs()…)
- **NEVRACÍ hodnotu typu pole ani více než jednu hodnotu**

Procedury

Struktura:

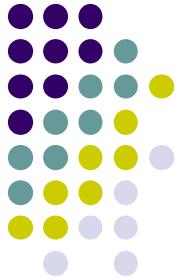
- Jméno procedury a parametry

  `Procedure` *`jméno`***`(`**parametry) nebo bez parametrů ***

  Formální parametry jsou v místě volání procedury nahrazeny skutečnými hodnotami

- Seznam proměnných a konstant
  - **Lokální** – jen pro tuto proceduru, nejsou přístupné pro další části programu, nutno deklarovat v podprogramu
  - **Globální** – proměnné a konstanty hlavního programu (definuje se jen v hlavním programu, můžou být použity i v podprogramech/funkcích)
- Tělo procedury

Lokální a globální proměnné

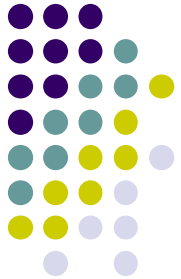


- **Lokální proměnné**

Mohou mít stejné jméno jako globální proměnné, program hledá v proceduře lokální proměnné, nenajde-li, hledá je v globálních (raději nepoužívat stejná jména)

Po zavolání procedury se jejím lokálním proměnným přiřadí paměť, po ukončení procedury z paměti smažou a uvolní místo pro proměnné dalších procedur

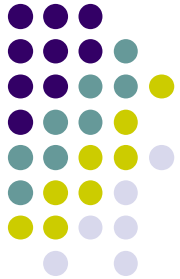
Parametry ***



- Protože každý podprogram nebo funkce může být z hlavního programu volán několikrát a pokaždé s jinými hodnotami, musí být deklarace procedury nebo funkce obecná. /pomocí parametrů)
- Parametry, které jsou uvedené v závorce za jménem procedury: *jméno: datový typ*
- Pro příkazy uvnitř procedury fungují jako proměnné

1)Procedury bez parametrů (za názvem procedury není žádná závorka s parametry)

Parametry ***



2) Parametry volané hodnotou

je vytvořena lokální kopie předávané proměnné, tzn. že žádná operace s parametrem uvnitř procedury nemůže ovlivnit žádnou nelokální proměnnou

Z hlavního programu předá do procedury/funkce hodnoty (dosadí je do jejích formálních parametrů, uvnitř procedury/funkce s nimi pracuje, nazpět do hlavního programu ale jejich hodnoty zpátky nepředá

Procedure NASOB (a:longint)

Parametry ***

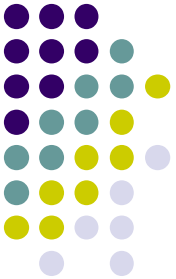
3) Parametry volané odkazem

Z hlavního programu se do procedury/funkce (do jejích formálních parametrů) dosadí skutečné hodnoty parametrů, jaké mají v místě volání procedury/funkce, uvnitř s nimi pracuje jako s globálními proměnnými .Při návratu zpět do hlavního programu předá nazpět parametry včetně hodnot, které v okamžiku výstupu z procedury nabývají

Procedura/funkce pracuje přímo s proměnnou (předává se její adresa), která je skutečným parametrem (a může tedy její obsah měnit), využití zejména pro výstupní parametry. Úspora paměti i času

Procedure SECTI (**var** a:longint)

Příklad



- Ředitel (hl.program) zadá úkol podřízeným (vyřeší ho procedura). Data k úkolu má ve svém bloku.

a) **Předání parametru hodnotou**

Podřízený si opíše data do svého bloku a dál s nimi pracuje

b) **Předání odkazem** (podřízený používá při práci blok ředitele – vstupní data, ale i zapisuje výstupy, může tam data i změnit)