Okruh 4 – Bloková struktura programu

1 Podprogramy (procedury a funkce)

1.1 Struktura

1. Hlavička (povinná)

   procedure Zamena (var A,B: integer);
   function Fak (N:byte):longint;
2. Deklarační blok {deklarace lokálních proměnných}

   var I:integer;
3. Příkazová část

   begin
   příkazy;
   end;

1.2 Příklady

```
procedure mocninaProc(z:real;e:integer; var v:real); {hlavicka}
var i:integer; {deklaracni blok}
begin {dale je prikazova cast}
  v:=1;
  for i:=1 to e do v:=v*z;
end;

function mocninaFce(z:longint;e:integer):longint;
var i:integer;
begin
  v:=1;
  for i:=1 to e do v:=v*z;
  mocnina:=v;
end;
```

1.3 Volání

1.3.1 Procedury

```
mocninaProc (x,y,v); {vystupni parametr je predan odkazem}
```

1.3.2 funkce

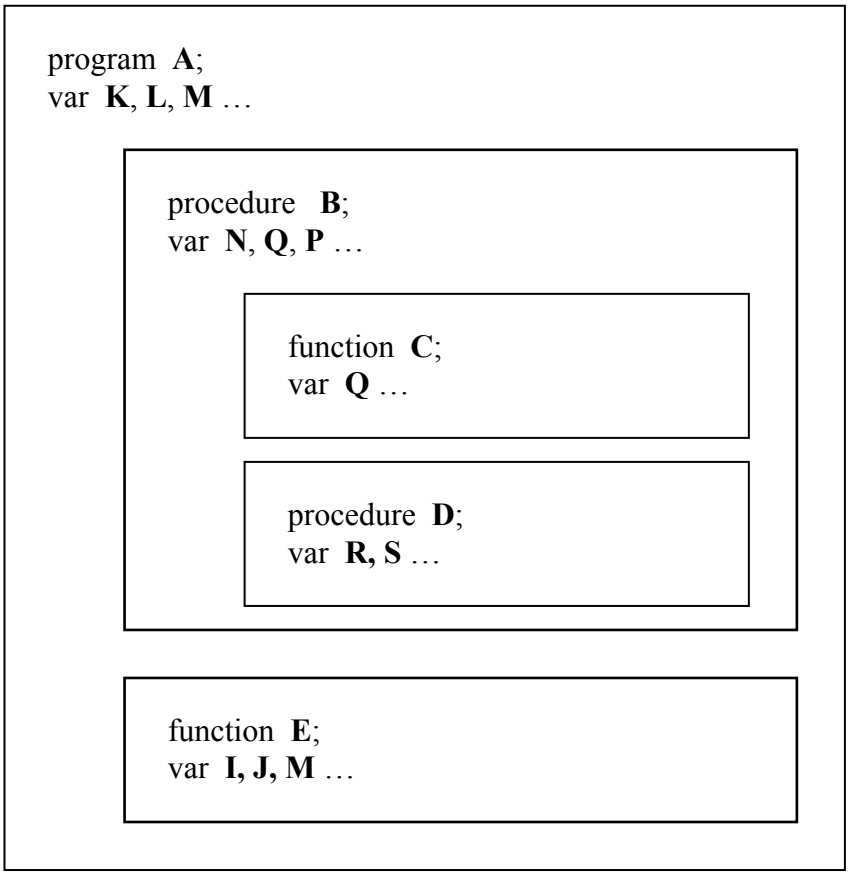
```
mocninaFce (x,y); {pred ukončením podprogramu je upravena globalni promenna}
```

1.4 Rozdíly mezi procedurou a funkcí

Liší se	**Procedura**	**Funkce**
Datovým typem	Nemá datový typ	Musí být určitého datového typu
Návratovými hodnotami	Vrací více hodnot, nebo žádnou (pomocí parametrů volaných odkazem)	Vrací vždy alespoň jednu hodnotu
Hlavičkou	procedure jmeno (var f1,f2:integer);	function jmeno (f1, f2:integer):integer;
Spouštěním (voláním)	Jmeno (A,B)	Přiřazením do proměnné: I:=jmeno (skutecneParametry) Ve výpisu: writeln (cos(x)) V podmínce: if cos (x)=1 then..

2 Bloková struktura programu

= hierarchická struktura „krabiček v krabičce“



2.1 Zásady

- zevnitř ven VŠECHNO VIDITELNÉ, POUŽITELNÉ, zvenčí dovnitř NEPRŮHLEDNÉ
- datové objekty (konstanty, proměnné, …) – globální, lokální
- při shodě jména lokálního a globálního objektu dojde k „zastínění identifikátoru“, tj. PLATNÝ je pouze LOKÁLNÍ

2.2 Globální a lokální datové objekty

2.2.1 Globální datové objekty

Znamená obecně platné, deklarovány v hlavním programu nebo aspoň o úroveň výš, než jsou použity

2.2.2 Lokální datové objekty

Jsou místně platné, použitelné pouze v bloku, v němž jsou deklarovány (tj. v podprogramu a jeho vnořených podprogramech)

3 Parametry podprogramů

3.1 Formální vs. skutečné parametry

3.1.1 Formální parametry

- Deklarovány v hlavičce podprogramu (v závorce za jeho jménem)

- Funkční jen v podprogramu (jako lokální konstanty a proměnné)
- Podprogram je s nimi jednou zapsán (jsou použitelné pouze v jeho příkazové části)

3.1.2 Skutečné parametry

- Podprogram je s nimi skutečně uváděn
- Uvedeny při volání podprogramu a jsou dosazeny místo formálních
- Mezi formálními a skutečnými parametry musí být zachován jejich počet, pořadí a musí být stejného (při volání odkazem) nebo kompatibilního datového typu jako formální

3.2 Parametry volané hodnotou vs. odkazem

Parametry volané	**Hodnotou**	**Odkazem**
Při deklaraci v hlavičce	Nemají var Function FAK(N:byte):longint;	Mají var Procedure zamena (var A,B:integer)
do/z podprogramu	Pouze vstupují	Jsou i výstupní
Skutečný předá formálnímu	Pouze svoji hodnotu	Adresu místa v paměti, kde je hodnota uložena
Na místě skutečného musí být	Hodnota Proměnná Výraz vyčíslitelný hodnotou	Vždy proměnná (aby bylo možné ji změnit)
Změna formálního parametru	Skutečný zůstane zachován	Změní i skutečný

4 Náhodná (pseudonáhodná) čísla

Jedná se o prostředky použité z unity System (k programu se připojuje automaticky).

Funkce *Random* vrací jistým způsobem zvolené (vypočtené) pseudonáhodné číslo, které lze použít pro výběr náhodné hodnoty požadovaného typu. Obor hodnot funkce závisí na parametru *Rozsah*. Při volání bez parametru je funkcí vráceno reálné číslo z polootevřeného intervalu <0, 1). Pokud naopak parametr uveden je, funkce vrací celé číslo typu *Word*, a to z intervalu <0, *Rozsah*) pro nenulový parametr resp. nulu pro parametr nulový.

V unitě Systém je deklarováno mimo jiné následující:

```
function Random: Real; {hlavička funkce volané bez intervalu}
function Random (Rozsah: Word): Word; {hlavička funkce, která se volá, pokud je zadán rozsah}

procedure Randomize; {vysvetleno nize}

const RandSeed: LongInt = 0; {také vysvetleno nize}
```

Výběr pseudonáhodného čísla, vraceného funkcí *Random*, je řízen aktuální hodnotou proměnné *RandSeed*. K dané hodnotě proměnné *RandSeed* funkce vždy vrátí totéž pevně dané náhodné číslo a následně aktualizuje hodnotu proměnné *RandSeed* pevně danou hodnotou následující. Pro danou počáteční hodnotu proměnné *RandSeed* (implicitně nula) je tedy generována vždy stejná posloupnost pevně daných pseudonáhodných čísel. Je-li třeba, aby aplikační program pracoval pokaždé s jinou posloupností pseudonáhodných čísel, je nutné inicializovat proměnnou *RandSeed* nějakou „náhodnou“ hodnotou, tj. takovou, která nebude při každém spuštění programu stejná. K tomu lze použít například uživatelem zadané náhodné číslo nebo proceduru *Randomize*, která do proměnné RandSeed dosadí hodnotu odvozenou z aktuálního stavu systémových hodin počítače.

4.1 Příklad naplnění pole náhodnými čísly o 50 prvcích:

```
program naplneniPoleRandomem;
uses crt;
type tpole=array[1..50] of byte;
var pole:tpole;
    i:byte;
begin
  clrscr;
  randomize;
  for i:=1 to 20 do
  begin
    pole[i]:=random(255);
  end;
  for i:=1 to 40 do
  begin
    writeln ('pole s indexem ', i, 'ma hodnotu ', pole[i])
  end;
  readln;
end.
```

5 Rekurze

5.1 Princip

musí být dán vztah, jak vypočítat n-tý člen n základě členů předchozích a hodnotě startovací (počáteční n=0)

5.2 Typy rekurzí

- **Přímá**– podprogram volá sám sebe, ale s jinými vstupními parametry
- **Nepřímá**– podprogram zavolá podprogram, který zase zavolá původní

5.3 Výhody a nevýhody

- **Výhody**– kratší kód, elegantnější pro práci
- **Nevýhody**– zpomalení programu (každé volání podprogramu vyžaduje nové vytváření lokálních proměnných, což je nevýhodné pro velký počet vnoření)

5.4 Příklad: Faktoriál pomocí rekurze:-)

```
program faktorialRekurzi;
uses crt;
var v:longint; x:integer;
function fak(n:integer):longint;
var i:integer;
begin
  if n<=1 then fak:=1
  else fak:=n*fak(n-1);
end;
begin
  clrscr;
  writeln ('zadejte cislo,u ktereho chcete spocitat FAKTORIAL');
  readln(x);
  v:=fak(x);
  writeln('vysledek: ',v);
  readln;
end.
```