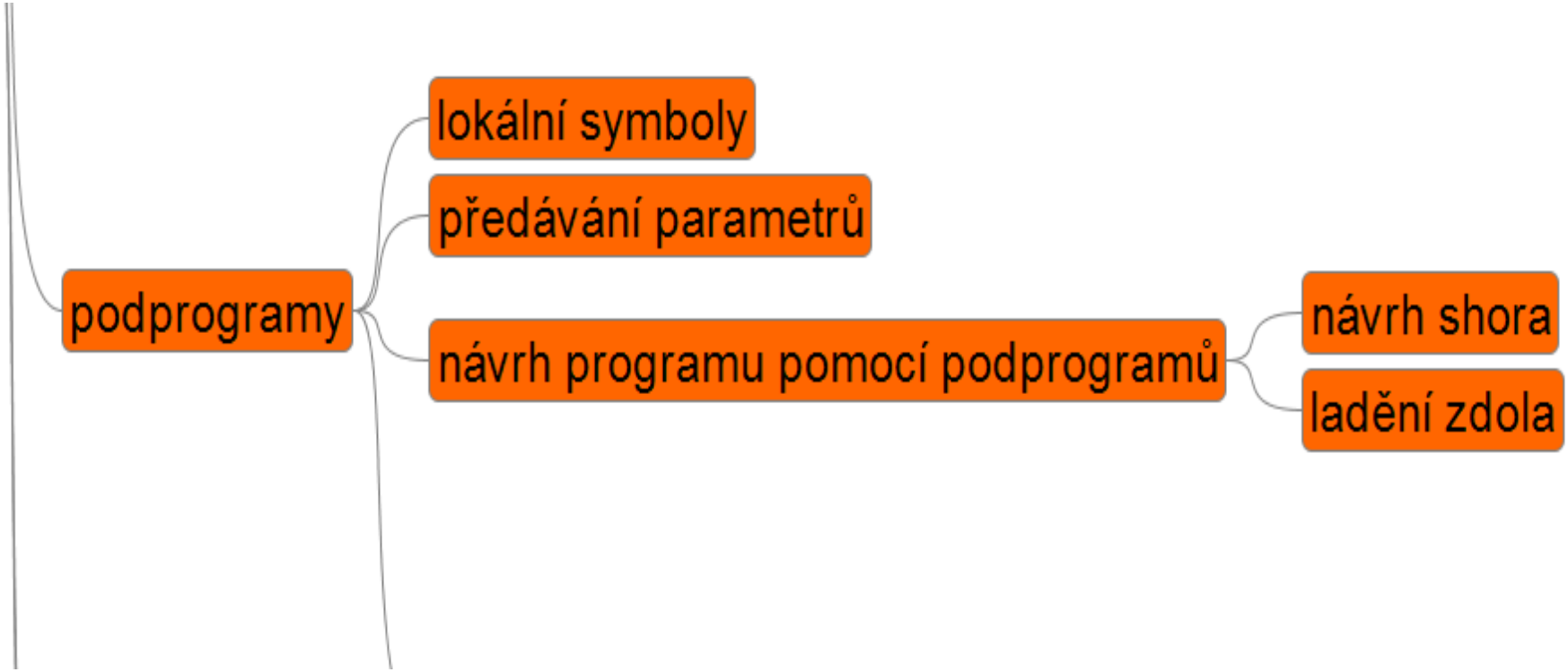

podprogramy
lokální symboly
předávání parametrů
návrh programu pomocí podprogramů
návrh shora
ladění zdola

Podprogramy

Příklad:

Vytiskněte tabulku malé násobilky ve tvaru

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X   X   1   2   3   4   5   6   7   8   9   10 X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X 1 X   1   2   3   4   5   6   7   8   9   10 X
X 2 X   2   4   6   8  10  12  14  16  18   20 X
X 3 X   3   6   9...............................
X 4 X   4   8...................................
X 5 X   5  10...................................
.........................................
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

blok

má tvar

```
const   definice konstant;
type    definice typů;
var     deklarace proměnných;
        deklarace procedur a funkcí;
tělo
```

Kromě těla může kterákoliv část chybět

Program, procedura i funkce mají tvar

hlavička; blok

BP/FP nevyžaduje dodržení pořadí a libovolná část z const, type, var, se může opakovat

např.

```
Program Muj;
const max = 15;
type  male = 1..max;
var A, B: male;
begin
   read( A,B );
   writeln( A+B )
end.
```

Hlavička procedury

```
procedure
   <identifikátor>
      (<seznam formálních parametrů>)
```

například:

```
procedure A;
procedure B( N: integer );
procedure C( M,N: integer );
procedure D( M: integer; c: char; k: integer );
```

Volání procedury

```
<identifikátor> (<seznam skutečných parametrů>)
```

například

```
A;
B( i );
C( k, 5 );
D( 2, 'a', 7 );
```

Hlavička funkce

```
function
   <identifikátor>
      (<seznam formálních parametrů>)
         : <typ výsledku>
```

například

```
function A: integer;
function B( N: integer ): boolean;
function C( M,N: integer ): char;
function D( M: integer; c: char ): char;
```

Příklad

```
function tg( x: real ): real;
begin
   tg := sin(x)/cos(x) { určení výsledné hodnoty }
end
```

Přirozený požadavek:

Bez ohledu na průběh výpočtu funkce musí být
v jejím těle jejímu identifikátoru
(nejméně jednou) přiřazena výsledná hodnota.

Pozn.: C# kontroluje

Volání funkce

<identifikátor> (<seznam skutečných parametrů>)
… uvnitř výrazu!

například

```
a := sqrt(z);
b := CisloDne( den, mesic, rok );
c := 40+round( 40*sin(fi*2*PI/360) );
```

Viditelnost identifikátorů

1. definice objektu musí předcházet jeho použití

2. viditelnost objektu je určena
   hierarchickou strukturou programu

3. jednoznačný význam identifikátoru v rámci

4. zastínění globáln(ějš)í definice lokáln(ějš)í definicí

```
program A
var i: integer; j: integer;
    procedure B( i: integer );
        function C( i: integer ): integer;
        begin
           C := i+j
        end;
    begin
       writeln( i );
       writeln( C(i) );
       i := i+1
    end;

    procedure D( x: integer );
    begin
       i := 1+x;
       B( x ); B( i )
    end;
begin
   i := 5;
   j := 2;
   D( 7 )
end.
```

Lokální symboly

- proměnné deklarované uvnitř podprogramů
- formální parametry
- a další

mají pouze lokální platnost

=>

- nelze s nimi pracovat v hlavním programu ani jinde mimo podprogram
- po vyvolání podprogramu (i opakovaném) hodnoty jeho lokálních proměnných NEJSOU DEFINOVÁNY

Shrnutí

- programátor může deklarovat nové podprogramy
- podprogram může mít své lokální proměnné (konstanty, typy, podprogramy)
- v hlavičce podprogramu mohou být deklarovány jeho formální parametry
- v příkazu volání uvedeme skutečné parametry
- jednou deklarovaný podprogram můžeme zavolat, kolikrát chceme

Proč podprogramy

- členění problému/programu na části,
  které můžeme řešit odděleně
- další úroveň oddělení CO TO DĚLÁ od JAK TO DĚLÁ
- skrývání proměnných atd., které mají význam jen pro řešení určité části
- re-use – možnost jednou vytvořený podprogram použít i v jiných programech

Předávání parametrů

a) hodnotou

b) odkazem

---

c) konstantní parametry (BP/FP)

d) výstupní parametry (C#)

e) výsledkem

f) jménem

g) …

Skutečným parametrem může být

a) výraz

b) proměnná odpovídajícího* typu

---

* odpovídající...

HODNOTOU

nová proměnná, do které se na začátku dosadí hodnota skutečného parametru

ODKAZEM

jen nové jméno pro proměnnou předanou jako parametr

```
procedure MINMAX( p: POLE; var MIN,MAX: real );
```

var platí pro všechny následující parametry až do dvojtečky

```
příklad: P( A, B, C: integer )...
```

Volba způsobu předávání parametrů

1. Pokud má přenášet hodnotu ven
   => odkazem
2. Pokud má skutečným parametrem být výraz
   => hodnotou
3. Vstupní data jednoduchého typu zpravidla hodnotou
4. Velké proměnné zpravidla odkazem

Konstantní parametry
= předávané odkazem, ale překladač nedovolí dosadit

Příklad:

Napište program, který najde 10 nejčastějších slov v souboru

Programování shora (dolů)

= řešení úlohy rozkladem na pod-úlohy
= využívá volání (zatím neexistujících) podprogramů

Programování zdola (nahoru)

= vytváření podprogramů (řešení pod-úloh),
o kterých myslíme, že je budeme potřebovat
a následně z nich skládáme řešení větších
pod-úloh, až k hlavní úloze

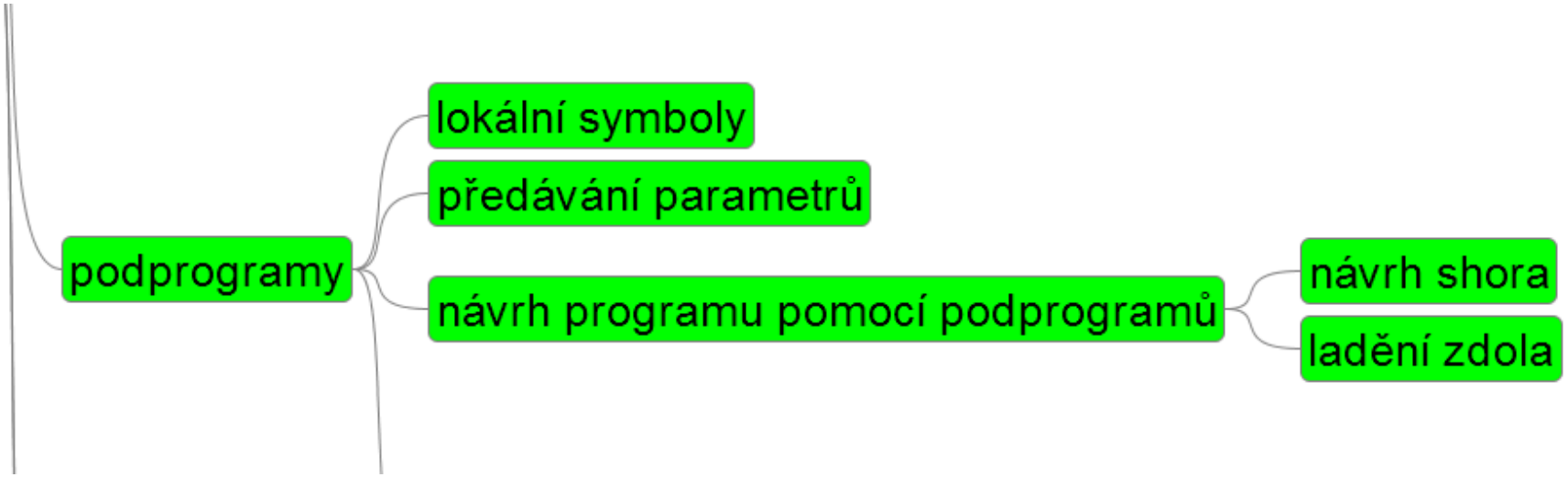
Ladění zdola

testovací podprogramy

Ladění shora

náhradní obsah podprogramů

Testovací podprogramy, vyhodnocování správnosti, počítání chyb, automatické testování


podprogramy
lokální symboly
předávání parametrů
návrh programu pomocí podprogramů
návrh shora
ladění zdola