Variantní záznamy

Datová struktura záznam umožňuje definovat tzv. variantní složku. Tímto způsobem lze vytvořit strukturu záznamů, ve které se jednotlivé záznamy odlišují alternativou variantní složky.

Rozlišovací složka – poslední společná složka, může to být libovolný ordinální typ.

Deklarace :

Type T = record

$S_1 : T_1$;
$S_2 : T_2$;
…
$S_{i-1} : T_{i-1}$;
} společné složky

case Si : T_i of

variantní složky {
$C_1 : (S_1^1 : T_1^1, \ldots, S_1^n : T_1^n)$;
$C_2 : (S_2^1 : T_2^1, \ldots, S_2^n : T_2^n)$;
…
$C_m : (S_m^1 : T_m^1, \ldots, S_m^n : T_m^n)$;

end ;

S_i, S_1^j - identifikátor i-té složky typu T_i , T_j^j
C_i - konstanty (hodnoty) ordinálního typu rozlišovací složky

POZOR !!! Používání záznamu s variantní složkou vyžaduje maximální opatrnost - záměna alternativ variantní složky není kontrolována !!!!!!!

Př.

```
Type STUDENT = record
                 EVIDCISLO : integer;
                 PRIJMENI,JMENO : string[20];
                 ADRESA: record
                           MÍSTO : record
                                     NAZEV : string[20];
                                     PSC :  1 .. 99999;
                                    end;
                           ULICE : string[15];
                           CISLO : 1 .. 9999;
                          End;
               case   POHLAVI : (ZENA, MUZ) of
                 ZENA : ( ROZENA : string[20];
                          POCET_DETI : 0 .. 20 ) ;
                 MUZ   :  ( VOJAK : boolean;
                            VYSKA, VAHA : real );
             end;

 var A : STUDENT;

begin
 …
 A.POHLAVI := MUZ ;    jsou přístupné položky VOJAK,
                       VYSKA , VAHA
 A.VOJAK := true;


 A.POHLAVI := ZENA  jsou přístupné položky ROZENA,
                    POČET_DETI
A.POCET_DETI := 5;

end.
```

Aktuální varianta záznamu je definována aktuální hodnotou jeho rozlišovací složky. Dojde-li ke změně aktuální hodnoty, varianta přestává existovat a vytvoří se nová aktuální varianta. Odkaz na neexistující variantu je chybný.

Záznam s variantními složkami bez rozlišovací složky

Rozlišovací složka nemá explicitní definici, definice záznamu obsahuje pouze ordinální typ T_i, jehož hodnoty odpovídají jednotlivým variantám a určují jejich počet. Přechod na novou variantu je způsoben odkazem na složku další možné varianty.
Aktuální je varianta, na kterou byl naposledy proveden odkaz.

Deklarace :

Type T = record

$S_1 : T_1$;
$S_2 : T_2$;
…
$S_{i-1} : T_{i-1}$;

} společné složky

case T_i of

variantní složky {

$C_1 : (S_1^1 : T_1^1, \dots , S_1^n : T_1^n)$;
$C_2 : (S_2^1 : T_2^1, \dots , S_2^n : T_2^n)$;
…
$C_m : (S_m^1 : T_m^1, \dots , S_m^n : T_m^n)$;

end ;

Př.

```
Type STUDENT = record
                  EVIDCISLO : integer;
                  PRIJMENI,JMENO : string[20];
                  ADRESA: record
                              MÍSTO : record
                                        NAZEV : string[20];
                                        PSC :  1 .. 99999;
                                       end;
                              ULICE : string[15];
                              CISLO : 1 .. 9999;
                             End;
               case (ZENA, MUZ) of
                 ZENA : ( ROZENA : string[20];
                          POCET_DETI : 0 .. 20 ) ;
                 MUZ   :  ( VOJAK : boolean;
                            VYSKA, VAHA : real );
            end;

 var A : STUDENT;
       V : real;
begin
 …
 A.VOJAK := true;    jsou přístupné položky VOJAK,
                           VYSKA , VAHA
 V := A.VYSKA;


 A.POCET_DETI := 2  jsou přístupné položky ROZENA,
                          POČET_DETI

V:= A.VAHA ;     CHYBA  !!!!
end.
```

Procedury a funkce

Řešení složitějších úloh – dekompozice (obvykle metodou shora dolů) na dílčí podproblémy ⟶ řešení jednotlivých podproblémů ⟶ sestavení dílčích podproblémů do jednoho celku, který řeší daný úkol.

Dílčí podproblémy jsou řešeny pomocí procedur, popř. funkcí.

Procedura (podprogram) – oddělený zápis části programu, který je využitelný v jednotlivých etapách výpočtu.

Důvody zavedení procedur a funkcí :

- **zvýší se přehlednost zápisu**
- **takto zapsaný podprogram umožňuje snadnou modifikaci**
- **části programu, které se opakují s různými datovými strukturami realizujeme pouze jednou**
- **možnost využívat knihovní procedury a funkce**

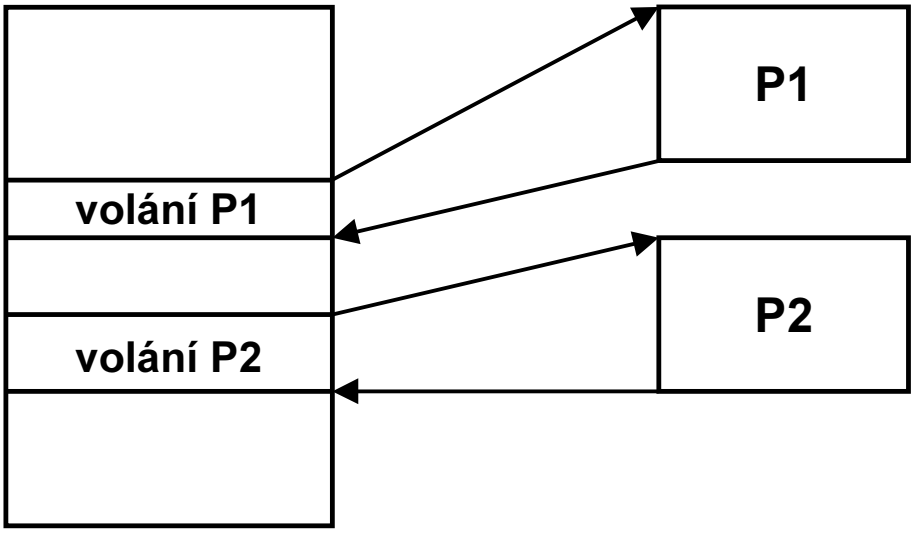
Zavedením procedury (funkce) „rozšiřujeme“ množinu příkazů daného programovacího jazyka.

Procedura je řídící struktura, která mění sekvenční vykonávání instrukcí.

Vykonání procedury :

- **volání procedury**
  - **uložení návratové adresy**
  - **předání parametrů**
  - **skok na první instrukci procedury**
- **vykonání příkazů procedury**
- **návrat do hlavního programu**

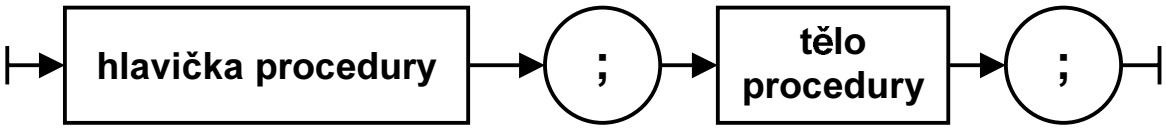
hlavní program



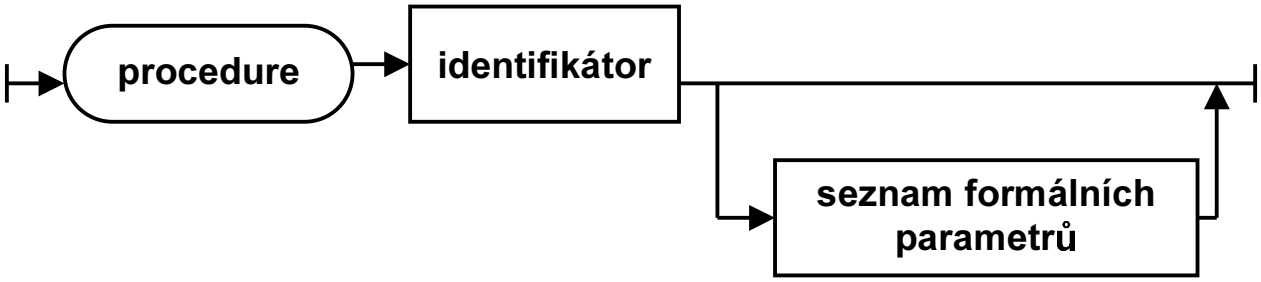
Procedury v Pascalu

Každá procedura používaná v programu musí být deklarována buď v deklarační části programu nebo v proceduře ve které se bude využívat.

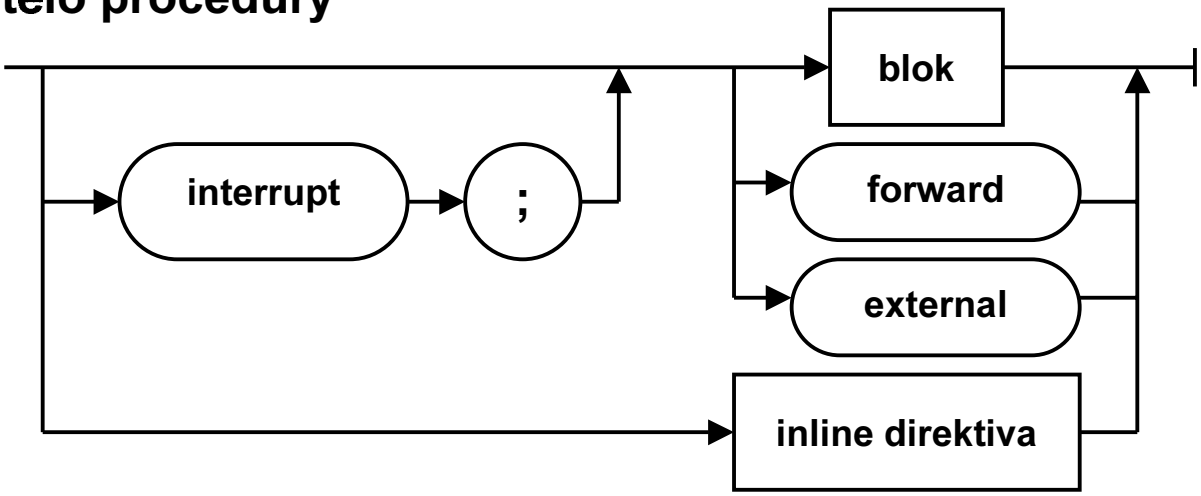
deklarace procedury



hlavička procedury



tělo procedury



Př.

```
 Program procedury;

 procedure P1(……..);
  begin                                  } deklarace procedury
    ….  příkazy procedury                }
   end;
begin
  …
  …
  P1(…… );          <-------------  volání procedury
  …
end.
```

Procedury :

- **bez parametrů**
- **s parametry**

Procedury bez parametrů

Procedury bez parametrů komunikují s hlavním programem prostřednictvím globálních proměnných tj. proměnných deklarovaných v hlavním programu.

Deklarace:

```
Procedure  identifikátor ;
   ……
 lokální deklarace        (nepovinné)
   ……
begin
  ……
 posloupnost příkazů
  ……
end;
```

Volání procedury

```
begin          { začátek hlavního programu }
……..
 identifikátor ;
…….
end.           { konec hlavního programu }
```

Příkazům v těle procedury mohou předcházet tzv. lokální deklarace (konstant, typů, proměnných, procedur a funkcí).
POZOR !!!! Lokální proměnné jsou viditelné pouze uvnitř procedury ve které jsou deklarovány a v procedurách vnořených. Vně procedury nejsou lokální proměnné deklarovány !!!!!!!

```
Program vnoření ;
 var A, B, C: integer ;
    procedure GLOB1;
     var C, D : real;
       procedure LOK1;
         var E: boolean;
       begin
         A^i, B^i, C^r, D^r, E^b
       end;
    begin
     A^i, B^i, C^i, D^r, LOK1
    end;
    procedure GLOB2;
     var F, C: char;
    begin
    A^i, B^i, C^ch, F^ch
    end;
Begin
A^i, B^i, GLOB1, GLOB2, C^i
End.
```

Nelokální objekt (proměnná, konstanta, procedura) je v proceduře (funkci) přístupný pouze tehdy, není-li v proceduře deklarován žádný objekt se stejným jménem identifikátoru !!!

Lokální objekty procedury jsou vytvářeny v okamžiku vyvolání procedury a jsou platné pouze po „dobu života“ procedury. Po ukončení procedury ztrácí platnost (a tedy i hodnotu). Při dalším volání procedury se znovu vytvářejí a mají tedy nedefinovanou hodnotu !!!!!!

```
program posloupnost ;
 var N, S : integer ;
       A     : array [ 1  .. 10 ] of integer ;

 procedure cti_posloupnost ;
  var I : integer;
 begin
  write('Zadej počet prvku');  read(N);
  write('Zadej prvky' );  for i:=1 to N  do read( A[ I ] ) ;
 end ;

procedure soucet;
 var I : integer;
 begin
  S := 0 ;
  for   i := 0 to N  do S := S + A[ I ] ;
 end;

begin
cti_posloupnost;
soucet;
writeln( 'Soucet prvku =' , S );
end.
```

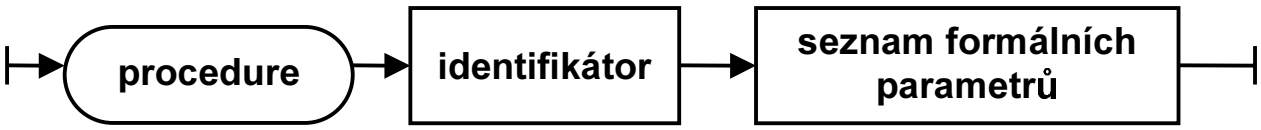
Procedury s parametry

Proměnné používané v proceduře :

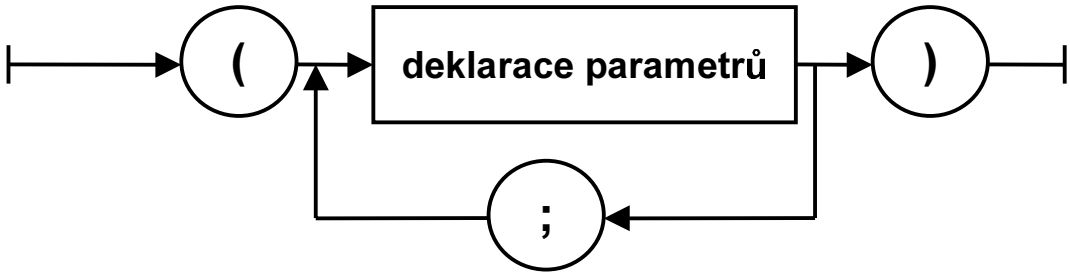
- **proměnné, které slouží pro předávání dat mezi procedurou a místem jejího volání (nelokální proměnné - deklarované vně procedury)**
- **proměnné, které slouží jako pracovní proměnné procedury (lokální proměnné procedury - deklarované uvnitř procedury)**

Komunikace procedury s okolím prostřednictvím nelokálních objektů - nevýhodná (před vyvoláním procedury musí být vhodně nastaveny hodnoty předávaných proměnných) ⟶ použití parametrů procedury pro komunikaci s okolím

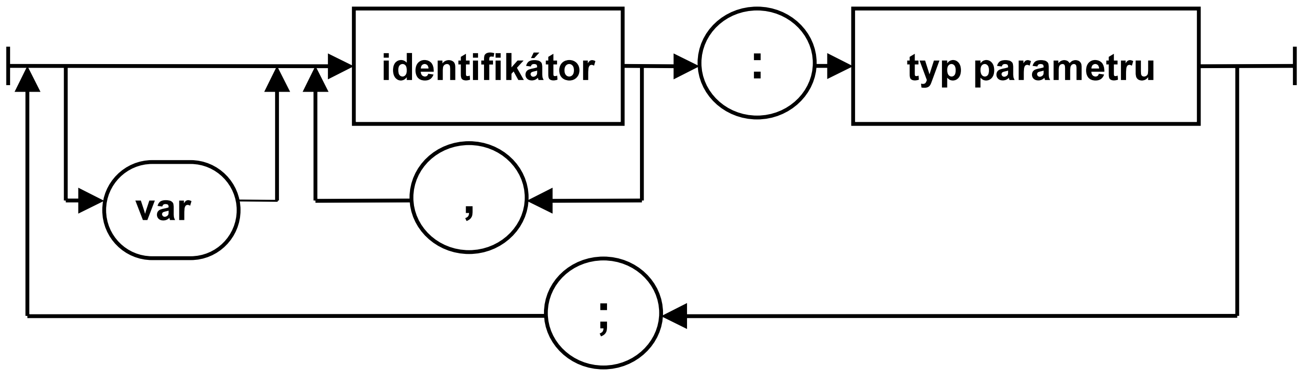
Parametry procedur se navenek chovají jako lokální proměnné, do kterých je v okamžiku vyvolání předávána hodnota (popř. adresa) proměnných z volaného programu.



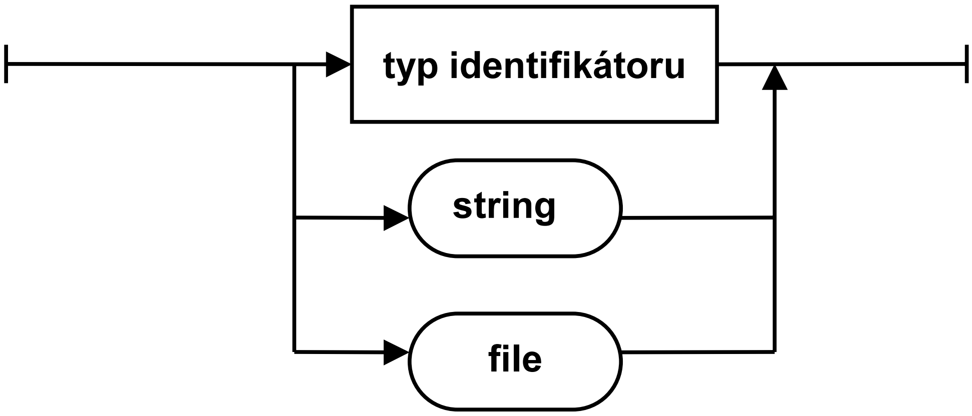
seznam formálních parametrů



deklarace parametrů



typ parametru



Typ indentifikátoru **může být libovolný jednoduchý, popř. strukturovaný typ.**

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! POZOR !!
U strukturovaných datových typů musí být pro deklaraci parametru zaveden uživatelsky definovaný typ (pomocí deklarace ***type*****). Přímý zápis strukturovaného typu není při deklaraci parametrů poolen .**
!!

Př.

```
program proc_s par ;
  type VEKT= array [1..10] of integer;
  var X : integer;
      C : VEKT;
  procedure P1( A,B : integer; C:VEKT);
    begin
      C[ A ] := B;
      Příkazy procedury          formální parametry
      ……
     end;                                          není povoleno
   procedure P2(d : array[1..5] of real)
     begin
       ……
     end;
begin
…       skutečné parametry
P1( 5, X*3 , C) ;
…
end.
```

Parametry procedury :

- **volané hodnotou**
- **volané odkazem**
- **procedurální parametry**
- **funkcionální parametry**

Parametry volané hodnotou

Deklarace : **formální parametr**

procedure P1(*fpar*:typ);

Volání : **skutečné parametry**

P1(*spar*) ;
P1(*výraz*) ;

Volání procedury se skládá z následujících kroků :

1. **V těle procedury se vytvoří lokální proměnná**
   var fpar : typ;

2. **Provede se přiřazení**

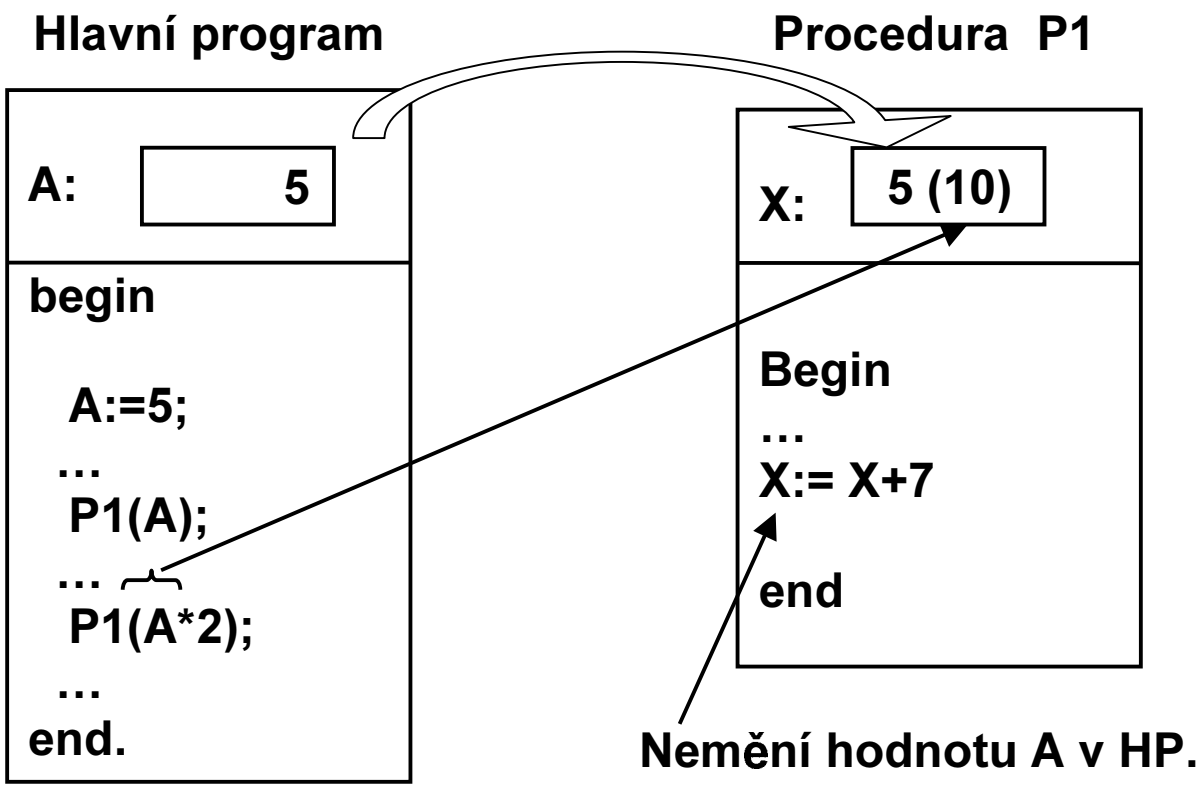
   ***fpar* := *spar*;**

3. **Provede se tělo procedury, každý výskyt výskyt formálního parametru se chápe jako označení pomocné lokální proměnné vztvořené v kroku 1**

```
procedure  P1( X: integer ) ;
 Begin
  ….
  X := X +7;
  …
end;

begin

A:=5;
…
 P1(A);
…
P1(A*2);
…
end;
```



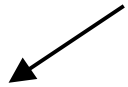
Přípustným skutečným parametrem při volání hodnotou je výraz, jehož hodnota je kompatibilní vzhledem k přiřazení s formálním parametrem.
Změna hodnoty formálního parametru neovlivní hodnotu odpovídajícího skutečného parametru tj. tento způsob volání se využívá pro *vstupní* parametry procedury.

Parametry volané odkazem

Deklarace : **formální parametr**

procedure P1(var *fpar*:typ);

Volání : **skutečný parametr**



P1(*spar*) ;

Volání procedury se skládá z následujících kroků :

1. **V těle procedury se vytvoří lokální proměnná**

   var fpar : *A*typ;

Proměnná fpar obsahuje adresu i když se v programu tváří jako proměnná obsahující hodnotu.

2. **Provede se přiřazení**

   ***fpar* := A*spar*;**

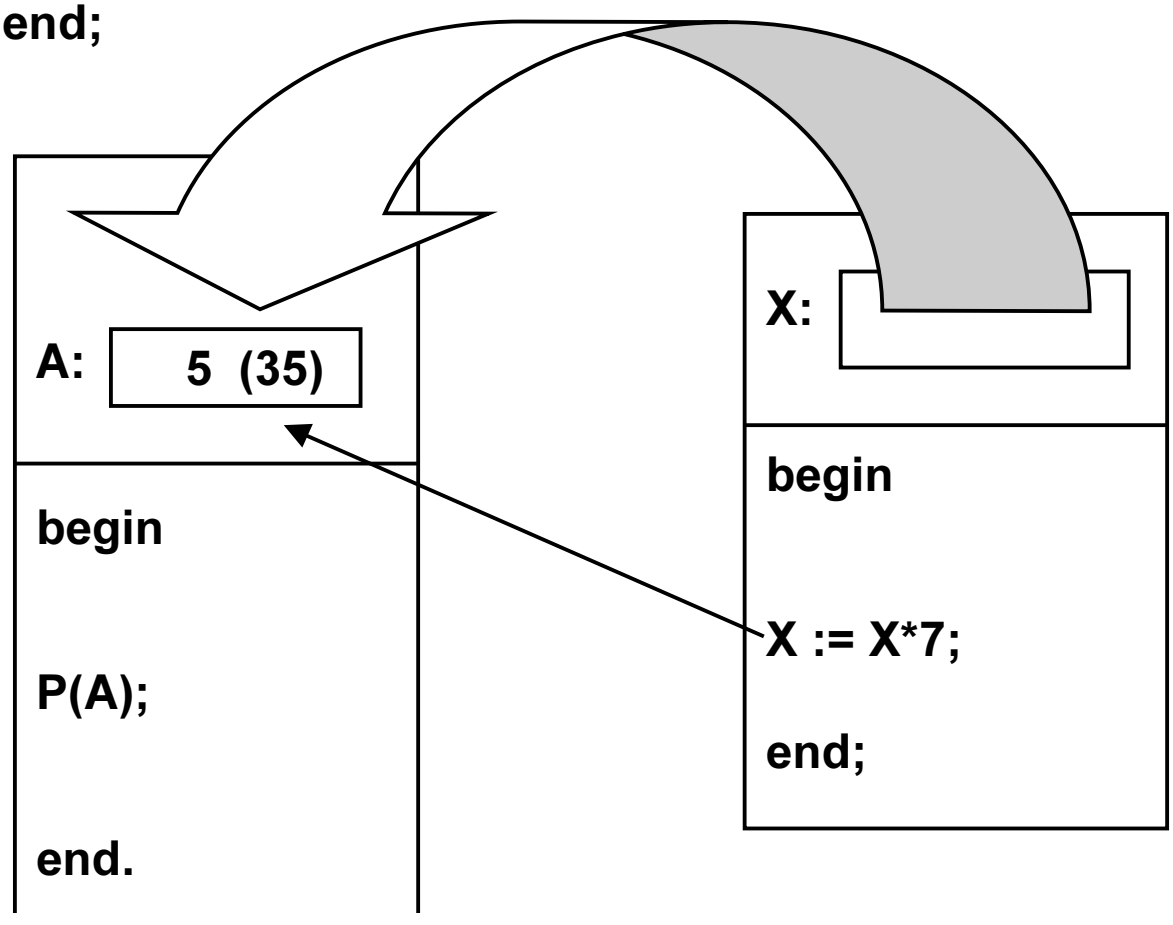
fpar obsahuje odkaz na hodnotu skutečného parametru

3. Provede se tělo procedury, každý výskyt formálního parametru se chápe jako označení pomocné lokální proměnné vytvořené v kroku 1

```
procedure  P1(var  X: integer ) ;
 Begin
  ….
  X := X +7;
  …
end;

begin

A:=5;
…
 P(A);
…
end;
```



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! POZOR !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
U parametrů volaných odkazem smí být jako skutečný parametr použita proměnná, jejíž typ je totožný s typem formálního parametru.
Jako skutečný parametr nesmí být výraz ani konstantní hodnota !!!!
!!

Změna formálního parametru uvnitř procedury mění hodnotu skutečného parametru vně procedury.

Parametry volané odkazem se používají jako výstupní parametry procedury.

V některých případech (strukturované datové typy velkých rozměrů) je výhodnější použít parametry volané odkazem i pro vstupní parametry procedury. Důvodem je úspora času a paměti.