# POČÍTAČE A PROGRAMOVÁNÍ

## Podmínka Case, Cykly, Iterační cyklus (For – to – do)

Miroslav Vavroušek

# Opakovaní z minulé přednášky

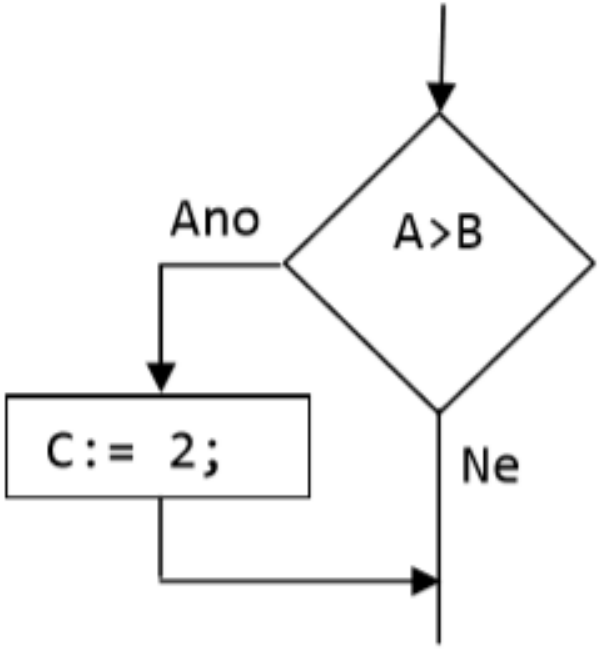
Struktura programu, Proměnné, Konstanty, Datové typy, Podmínka If-then

# Podmíněný příkaz

## Podmínka **If**

**Podmínka if**

```
if Podminka then
begin
  // Pokud je podmínka splněna
end;
```



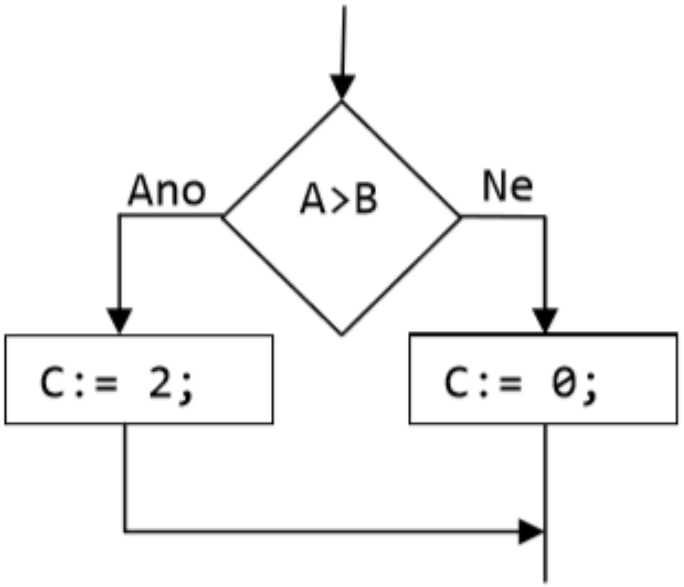

```
{Pokud je A větší než B do C vlož 2}
if A>B then
begin
  C:= 2;
end;
```

# Podmíněný příkaz

## Úplná podmínka **If – else**

**Úplná podmínka if-else**

```
if Podminka then
begin
  // Pokud je podmínka splněna
end else
begin
  // Pokud je podmínka nesplněna
end;
```




```
if A>B then
begin
   C:= 2;{Pokud je A větší
end else            než B do C vlož 2}
begin
  C:= 0;{Pokud A není větší
end;                než B do C vlož 0}
```

# Obsah přednášky

# Podmínka Case, Cykly, Iterační cyklus For

# Podmínka **Case**

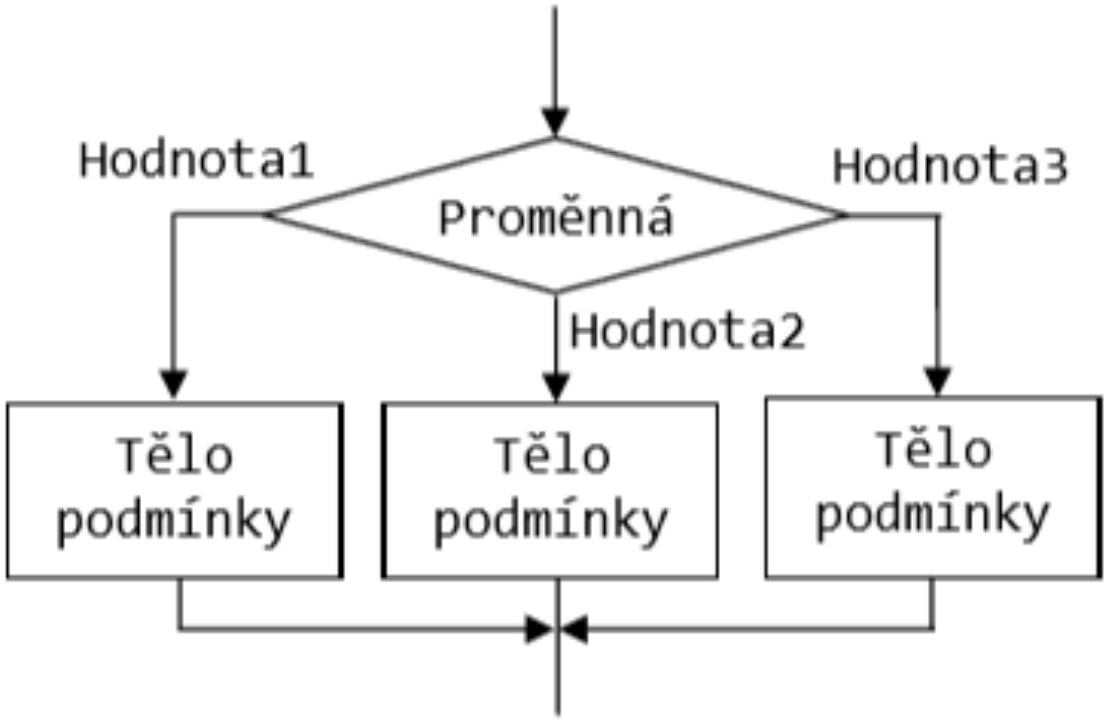
- Umožňuje větvení do více než dvou cest (větví)
- Cesta (větev) je vybrána podle hodnot v návěstí
- Je možné vytvořit větev, která je vykonána pokud hodnota proměnné neodpovídá žádnému navěstí
- Proměnná požita v podmínce case-of musí být ordinálního typu

# Podmínka **Case**

- Konstrukce podmínky **Case**:

**Podmínka case-of**

```
case Proměnná of
  Hodnota1: begin
      //Tělo podmínky 1
    end;
  Hodnota2: begin
      //Tělo podmínky 2
    end;
  Hodnota3: begin
      //Tělo podmínky 3
    end;
end;
```

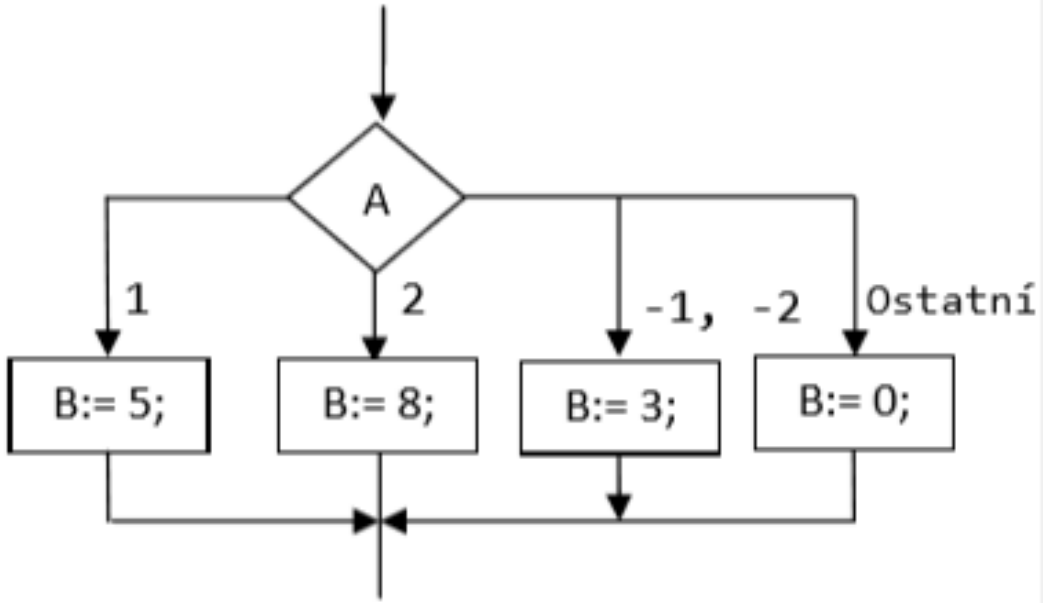


# Podmínka **Case**

- Ukázka konkrétního použití podmínky **Case**

**Podmínka case-of**

```
case A of
  1: begin
       B:= 5; {Tělo podmínky
               pro A = 1}
     end;
  2: begin
       B:= 8; {Tělo podmínky
                 pro A = 2}
     end;
  -1, -2: begin
       B:= 3; {Tělo podmínky
               pro A = -1, -2}
     end;
  else begin
       B:= 0; {Tělo podmínky
               pro A = ostatní}
     end;
end;
```



# Podmínky

## **If (– else)** vs. **Case**

- Podmínku **Case** lze zapsat pomocí jedné nebo více podmínek **If (– else)**
  - Nemusí platit obráceně
- Podmínku **Case** lze použít k efektivnějšímu a přehlednějšímu zápisu kódu
- ☆ Hodnoty návěští lze zapsat i pomocí intervalu

poznámka

```
case A of
  1..10: begin
           {pro A od 1 do 10}
         end;
end;
```

# Cykly

- Cykly se požívají k opakovaní zvolené části kódů
- V jazyku Pascal existují tři typy cyklů:
  - Iterační cyklus (**For – to – do**)
  - Cyklus s podmínkou na začátku (**While – d**o)
  - Cyklus s podmínkou na konci (**Repeat – until**)
- Vhodný typ cyklu je zvolen na základě vlastností vyžadovaných v konkrétní situaci.

# Vlastnosti iteračního cyklu
## (**For – to – do)**

- Nejčastěji používaný cyklus
- Používá se všude tam kde je znám počet opakovaní před započetím cyklu
- Vždy používá některou z proměnných (tzv. řídící proměnná cyklu), která musí být deklarována jako proměnná ordinálního typu

```
for A:=1 to 10 do
begin
  WriteLN(A);
end;
```

**Hlavička cyklu**

**Tělo cyklu**

# Vlastnosti iteračního cyklu (**For – to – do)**

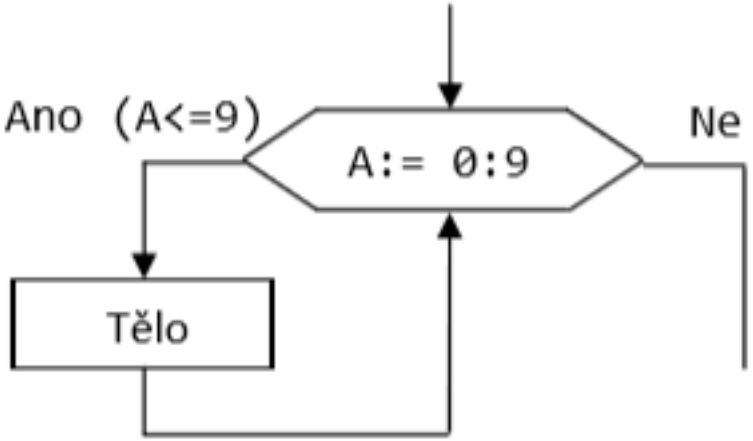
- Vždy má definovánu počáteční a koncovou hodnotu
- Cyklus provádí změnu řídicí proměnné automaticky sám
  - Hodnotu řídící proměnné cyklu je zakázáno v těle cyklu měnit. Je jí možné pouze číst.
  - Nikdy nemůže nastat nekonečný cyklus
  - Krok je vždy o 1 hodnotu ordinálního typu
    - Vzestupný (následovník) vs. Sestupný (předchůdce)
- Tělo cyklu nemusí proběhnou ani jednou

# Vzestupný iterační cyklus
## (**For – to – do)**

**Cyklus for-to-do vzestupný**

```
for Proměnná:=A to B do
begin
  {Hodnota v proměnná stoupá
  v jednotlivých opakováních od A
  do B}
end;
```

```
for A:=0 to 9 do
begin
  {Hodnota v A stoupá v jednotlivých
   opakováních od 0 do 9}
end;
```

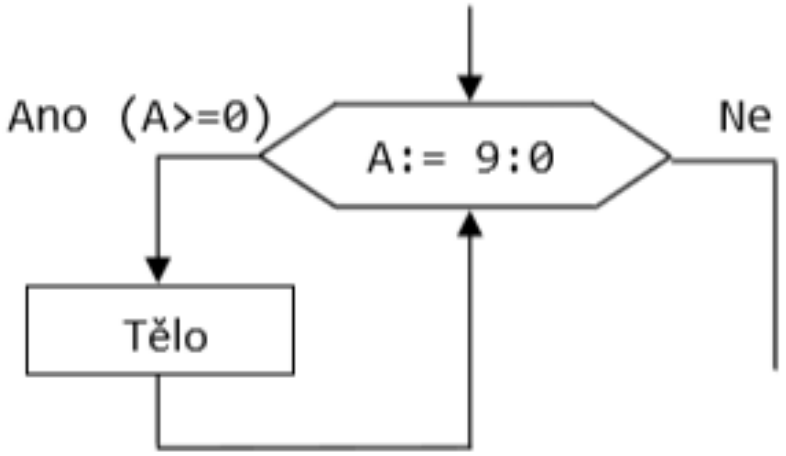


# Sestupný iterační cyklus
## (**For – downto – do)**

**Cyklus for-downto-do sestupný**

```
for Proměnná:=A downto B do
begin
   {Hodnota v proměnná klesá
   v jednotlivých opakováních od A
   do B}
end;
```

```
for A:=9 downto 0 do
begin
  {Hodnota v A klesá v jednotlivých
   opakováních od 9 do 0}
end;
```



# Iterační cyklus
## Příkazy **break** a **continue**

- Příkazy umožňují přerušení vykonávání těla cyklu
- Příkaz **break** přerušuje cely cyklus **for – to – do**
- Příkaz **continue** přerušuje aktuální iteraci cyklu **for – to – do**
- Vztahují se k cyklu, ve kterém jsou zapsány
  - k nejbližší úrovni vnoření

```
break;
```

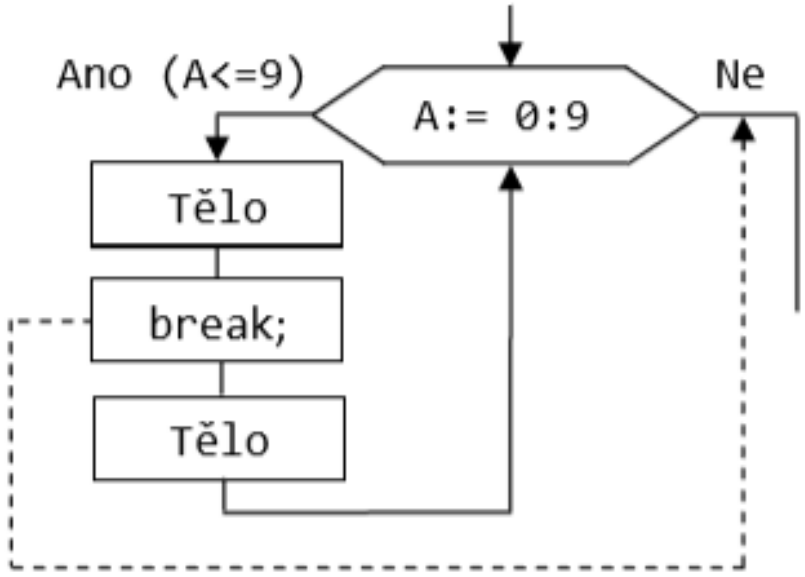
```
continue;
```

# Iterační cyklus

## Příkaz **break**

**Cyklus for-to-do příkaz break**

```
for Proměnná:=A to B do
begin
  //Tělo cyklu
  break;
  //Tělo cyklu
end;
```



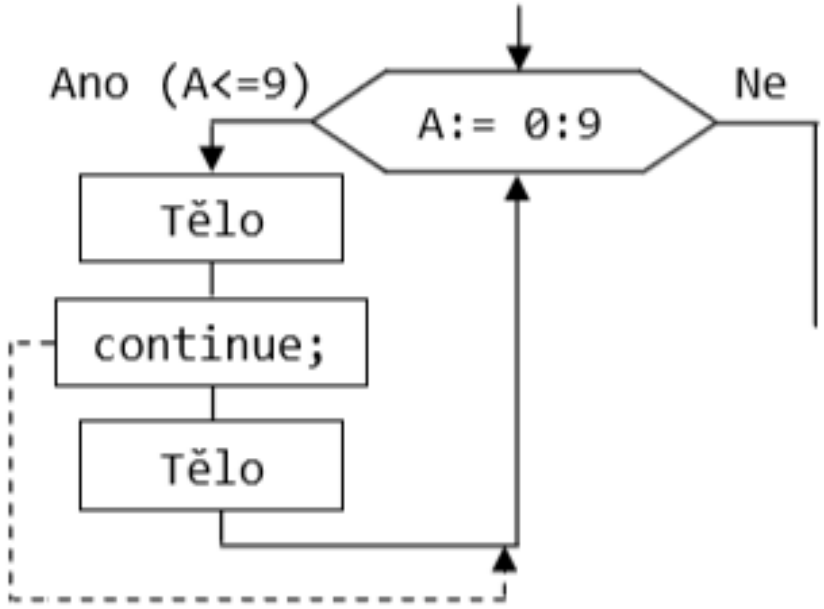

```
for A:=0 to 9 do
begin
  //Tělo cyklu
  if A=5 then
  begin
    break; {Při A = 5 ukončí cyklus.
            Nedojde tedy nikdy na 6-9}
  end;
  {Tělo cyklu (neproběhne pro A = 5)}
end;
```

# Iterační cyklus

## Příkaz **continue**

**Cyklus for-to-do příkaz continue**

```
for Proměnná:=A to B do
begin
  //Tělo cyklu
  continue;
  //Tělo cyklu
end;
```




```
for A:=0 to 9 do
begin
  //Tělo cyklu
  if A=5 then
  begin
    continue; {Při A = 5 ukončí kolo
              cyklu a pokračuje při
              A = 6}
  end;
  {Tělo cyklu (neproběhne pro A = 5)}
end;
```