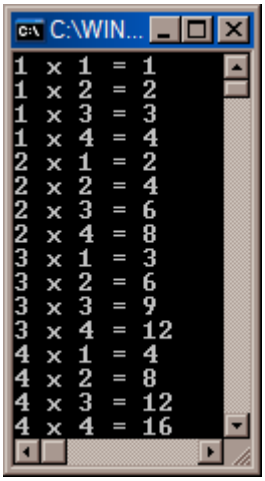
Řídící struktury II

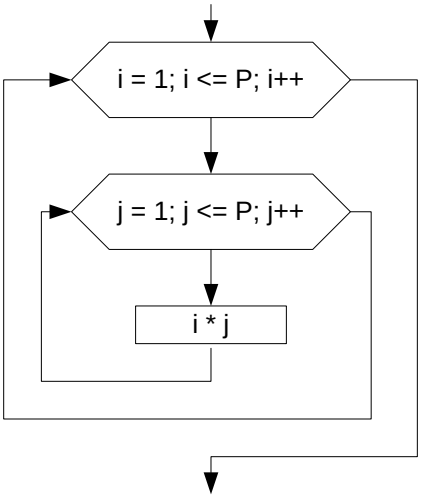
Vnořené cykly

- Uvnitř cyklu → další cyklus
- Příklad: výpis všech kombinací malé násobilky od `1` do `POCET`

```
const int POCET = 4;
for (int i = 1; i <= POCET; i++)
{
        for (int j = 1; j <= POCET; j++)
        {
                Console.WriteLine("{0} x {1} = {2}", i, j, i * j);
        }
}
```

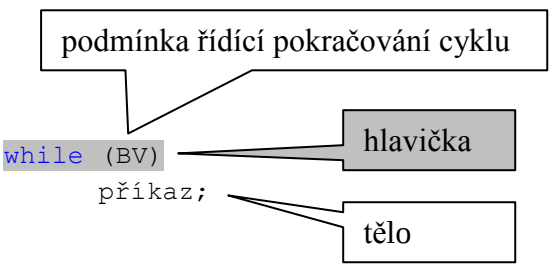
1 příkaz

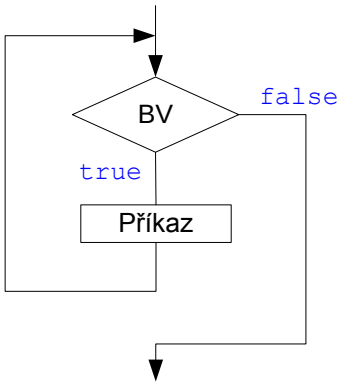




Cyklus while

- pro situace: počet opakování není předem znám
- Syntaxe: sémantika:





- `BV` se testuje před každým průchodem cyklu → cyklus nemusí proběhnout ani jednou

Typické použití

- Součet několika čísel, počet čísel určen tzv. zarážkou – při zadání zarážky se program ukončí

```
double cislo, soucet = 0.0;
cislo = 10; // cokoli, hlavne aby poprve probehlo while
while (cislo != 0.0)
{
      Console.Write("Zadej cislo: ");
      cislo = Double.Parse(Console.ReadLine());
      soucet += cislo;
}
Console.WriteLine("Soucet = {0}", soucet);
```

Poznámky

- nekonečná smyčka:

```
while (true)
{
      příkaz;
}
```

- `while` může nahradit `for` – nedoporučeno, každý cyklus má svoji oblast použití

```
for (int i = 0; i < 10; i++)
{
      příkaz;
}
```

```
int i = 0;
while (i < 10)
{
      příkaz;
      i++;
}
```

Cyklus do while

- syntaxe

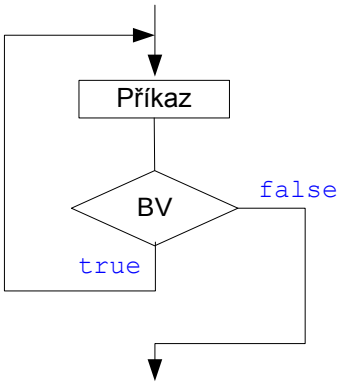
```
do
        příkaz;
while (BV);
```

tělo

hlavička

!!!

sémantika



- pozor:

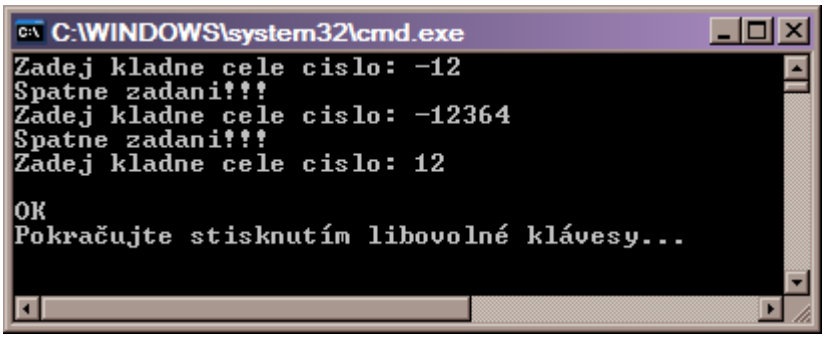
nutné

```
do
{
        příkaz1;
        příkaz2;
        příkazN;
}
while (BV);
```

Příklad

- Ošetření zadání správných hodnot při zadání – jen kladná celá čísla

```
int cislo;
do
{
        Console.Write("Zadej kladne cele cislo: ");
        cislo = Int32.Parse(Console.ReadLine());
        if (cislo < 0)
        {
                Console.WriteLine("Spatne zadani!!!");
        }
} while (cislo < 0);
Console.WriteLine("OK");
```



Příkazy break a continue

- upravují „normální“ běh cyklů
  - `break` – ukončuje (ihned opouští) cyklus
  - `continue` – skočí na konec cyklu = vynutí běh další smyčky cyklu
- lze použít ve všech cyklech (`for`, `while`, `do while`)
- Příklad:

  Načítání čísel. Pokud je kladné, vypíše jeho dvojnásobek, záporných si nevšímá, nula ukončí cyklus.

```
int cislo;
while (true)      // nekonecna smycka
{
      Console.Write("zadej cislo: ");
      cislo = Int32.Parse(Console.ReadLine());
      if (cislo < 0)
      {
            continue;   // dalsi nacitani
      }
      if (cislo == 0)
      {
            break;      // konec cyklu
      }
      Console.WriteLine("vysledek = {0}", 2*cislo);
}
```

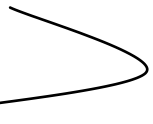
- `continue` mnohem méně často než `break`

- použití `break`:
  - předčasné ukončení cyklu (např. kvůli chybě)
  - řádné ukončení nekonečného cyklu

```
int cislo;
while (true)
{
      Console.Write("Zadej kladne cele cislo: ");
      cislo = Int32.Parse(Console.ReadLine());
      if (cislo >= 0)
            break;
      Console.WriteLine("Spatne zadani!!!");
}
```

- vnořené cykly – působí na cyklus, ve kterém jsou uvedeny

```
while (true)
{
      // kod
      for (int i = 0; i < 10; i++)
      {
            // kod
            break;
            // kod
      }
      // kod
}
```

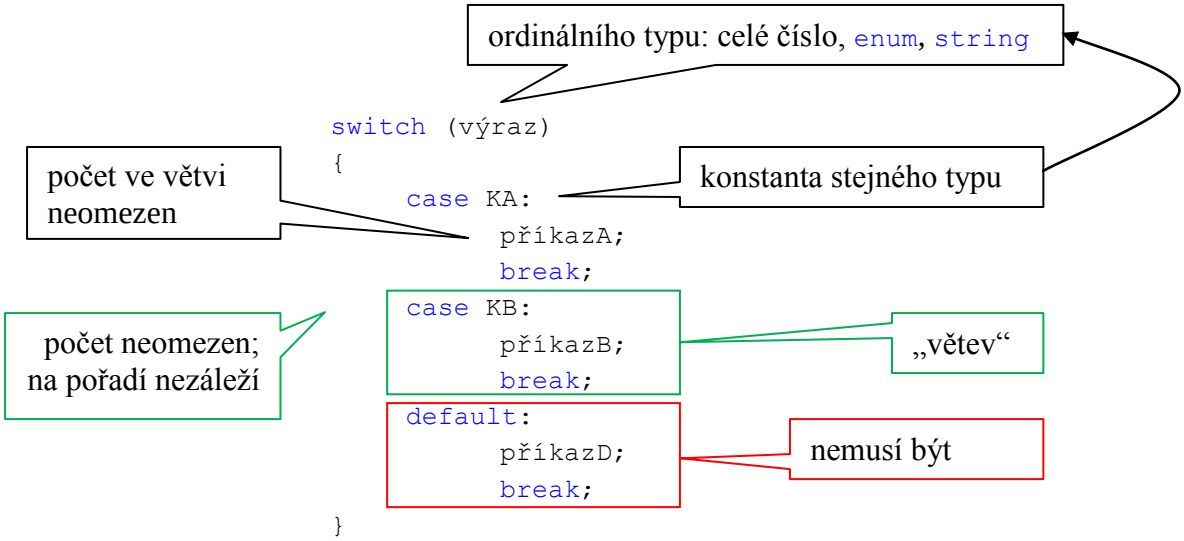


Závěrečné poznámky k cyklům

- špatně (obráceně) postavená ukončující podmínka = porušení konečnosti algoritmů
- Základní pravidla:
  - řídící proměnná se musí při každé otáčce změnit (u `for` relativně snadné, u `do-while` a `while` se o to musí postarat programátor)
  - Do cyklu nutno vstoupit s jasně definovanými počátečními podmínkami
- Shrnutí využití
  - `for` – před spuštěním cyklu je počet opakování předem znám
  - `while` – počet opakování není znám, nemusí proběhnout ani jednou
  - `do while` – počet opakování není znám, min. jeden běh

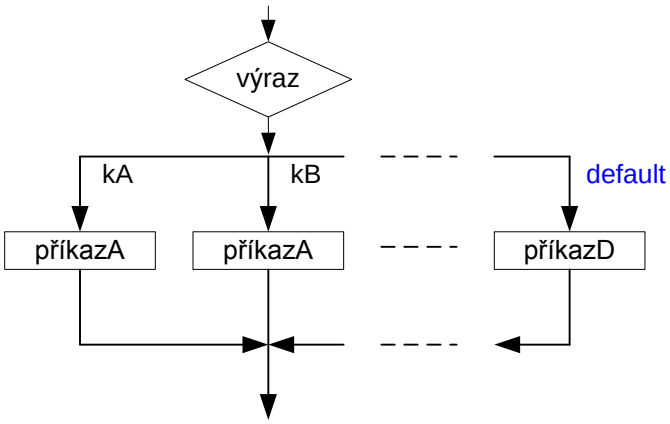
Příkaz switch

- několikanásobné větvení programu
- syntaxe:

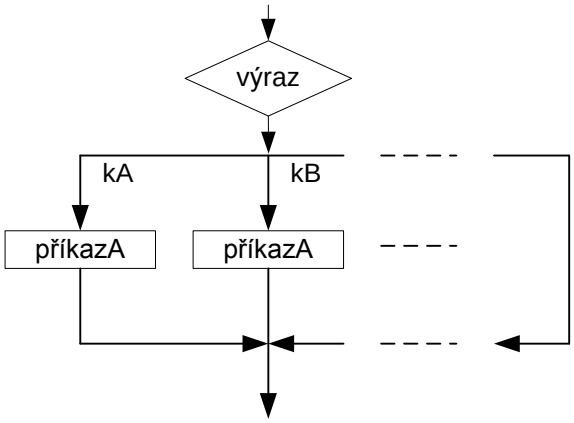


- Sémantika:

  s `default`



  bez `default`



- Příklad:
  Převod zkratky dne na plné znění

```
string den;
switch (den)
{
      case "po":
            Console.WriteLine("Pondělí");
            break;
      case "ut":
            Console.WriteLine("Uterý");
            break;
      // atd.
      default:
            Console.WriteLine("Není zkratka dne");
            break;
}
```

výraz typu `string`

konstanta typu `string`

- Poznámky:
  - Větvení pouze podle konstant, nikoli intervalů (potom nutno vnořené `if else`)
  - `switch` uvnitř cyklů nebo obráceně → `break` se vztahuje ke „svému nejbližšímu“ příkazu

Propadání

- několik `case` sdílí stejné příkazy
- příklad

```
// int pocetPiv;
Console.Write("Vypil jsem ");
switch (pocetPiv)
{
      case 1:
            Console.WriteLine("1 pivo");
            break;
      case 2:
      case 3:
      case 4:
            Console.WriteLine("{0} piva", pocetPiv);
            break;
      default:    // 5 a více
            Console.WriteLine("{0} piv", pocetPiv);
            break;
}
```

`case` bez `break` – nelze žádný příkaz uvnitř

Příkaz goto

- nepodmíněný skok
- syntaxe

- sémantika: skok na `label`
- lze vždy nahradit jinou programovou konstrukcí – slušný programátor nepoužívá
- vhodné využití – ukončení všech vnořených cyklů z vnitřní smyčky

```
for (int i = 0; i <= NEKAM; i++)
{
      for (int j = 0; j <= NEKAM_JINAM; j++)
      {
            if (DOSLO_K_CHYBE)
                  goto konec;
      }
}
konec: příkaz;
```

- řešení bez `goto`

```
bool nastalaChyba = false;
for (int i = 0; i <= NEKAM; i++)
{
      for (int j = 0; j <= NEKAM_JINAM; j++)
      {
            if (DOSLO_K_CHYBE)
            {
                  nastalaChyba = true;
                  break;
            }
      }
      if (nastalaChyba)
            break;
}
příkaz;
```