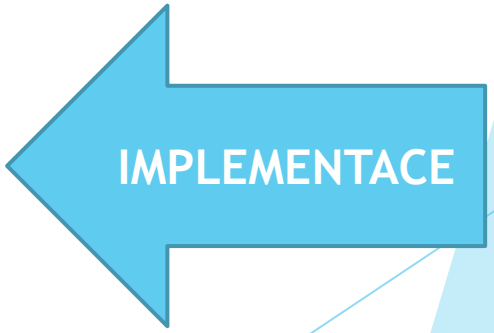
Programování

Základní konstrukce v programování a jejich použití

Postup vytváření algoritmu - opakování

- Formulace problému
  - Formulace požadavků, určení vstupů/výstupu a požadavků na přesnost
- Analýza úlohy
  - Ověření řešitelnosti a počtu řešení dle zadaných vstupů
- Vytvoření algoritmu
  - Sestavení sledu operací, které vedou k požadovanému výsledku
- **Sestavení programu**
  - Vytvoření zdrojového kódu v příslušném programovacím jazyce
- Odladění programu
  - Odstraňování logických a syntaktických chyb v programu



Jak vytvořit kód z vývojového diagramu?

- Jazyk C# je imperativní (přikazovací) programovací jazyk
  - Vykonává instrukce, které mu jsou zadány
- Sled vykonávaných kroků odpovídá algoritmu znázorněné ve vývojovém diagramu (VD)
- Pořadí jednotlivých operací VD ⇔ pořadí instrukcí ve zdrojovém kódu
- Jednotlivé symboly VD odpovídají konkrétním programovacím konstrukcím

Blok programu

Zápis ve VD

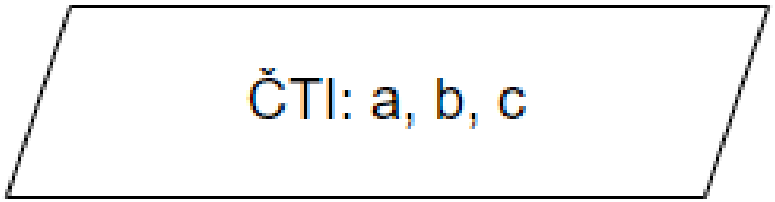
Zápis pomocí C#

```
{
    // VYKONEJ
}
```

Přiřazení hodnoty do proměnné provádíme pomocí operandu =
Např.: suma = 1 + 2 + 3;

Vstupní operace – čtení z konzole

Zápis ve VD



Předpokládáme, že budeme načítat celá čísla

Zápis pomocí C#

```
int a, b, c;
```

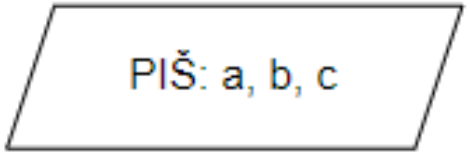
Deklarace proměnných

```
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
c = int.Parse(Console.ReadLine());
```

Přiřazení přetypované hodnoty, kterou jsme načetli

Výstupní operace – výpis na konzoli

Zápis ve VD



Předpokládáme, že proměnné jsou čísla

Zápis pomocí C#

```
Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

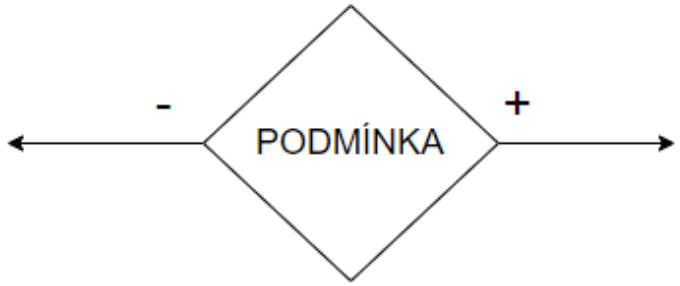
Hodnota v a
Hodnota v b
Hodnota v c

```
Console.WriteLine(a + ", " + b
+ ", " + c);
```

Hodnota v a, hodnota v b, hodnota v C

Rozhodovací blok – podmínky (úplná)

Zápis ve VD

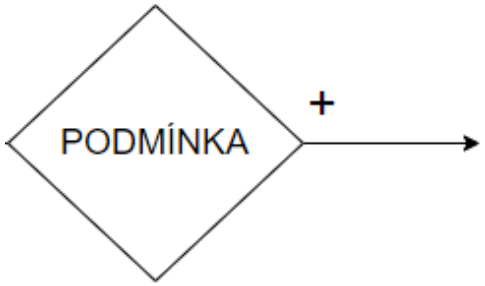


Zápis pomocí C#

```
if (PODMÍNKA)
   {
    // POKUD JE PODMÍNKA SPLNĚNÁ
   }
else
   {
   // POKUD PODMÍNKA NENÍ SPLNĚNA
   }
```

Rozhodovací blok – podmínky (neúplná)

Zápis ve VD



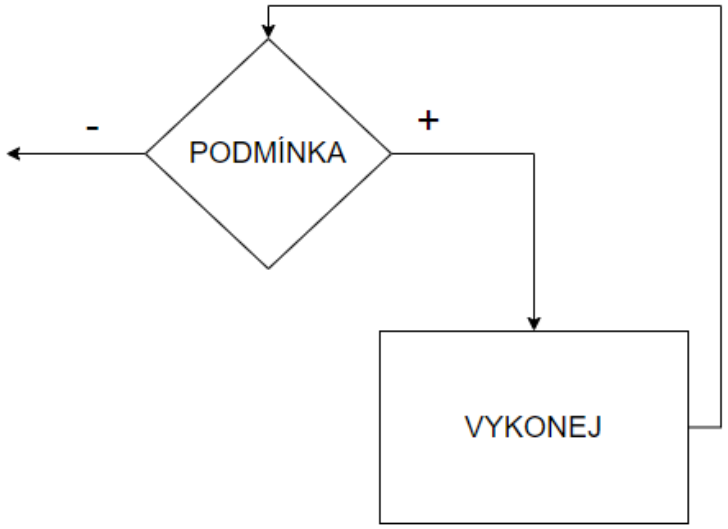
Zápis pomocí C#

```
if (PODMÍNKA)
   {
    // POKUD JE PODMÍNKA SPLNĚNÁ
   }
```

- Podmínka musí mít vždy alespoň kladnou větev

Cyklus s neznámým počtem opakování – while

Zápis ve VD



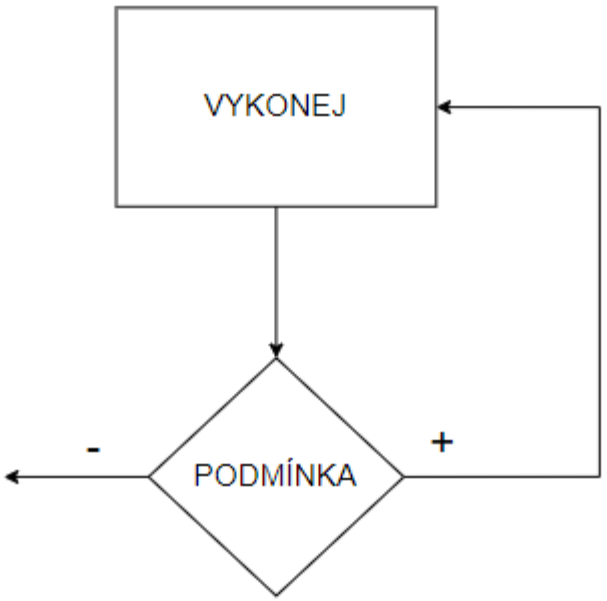
Zápis pomocí C#

```
while(PODMÍNKA)
    {
      // POKUD JE PODMÍNKA SPLNĚNÁ
    }
```

- Pokud podmínka není splněná, nebude kód uvnitř bloku vykonán

Cyklus s neznámým počtem opakování – do while

Zápis ve VD



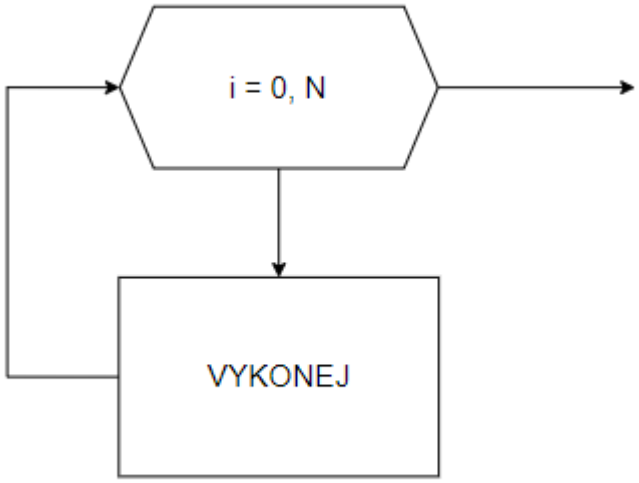
Zápis pomocí C#

```
do
{
    // POKUD JE PODMÍNKA SPLNĚNÁ
} while(PODMÍNKA);
```

- Kód se alespoň jednou vykoná a až poté se kontroluje podmínka

Cyklus s pevným počtem opakování - for

Zápis ve VD



Zápis pomocí C#

Deklarace řídící proměnné

Zvětšení řídící proměnné na konci cyklu

```
for (int i=0; i < N; i++)
{
    // VYKONEJ
}
```

Testovací podmínka zda má cyklus pokračovat

Čtení kódu a pochopení jeho funkcionality

- Program v ideálním případě kopíruje navržený diagram včetně deklarace proměnných
  - Některé proměnné (obvykle dočasné) se deklarují až v průběhu programu
  - Proměnnou lze používat až od řádku, kdy byla deklarována
- Každý řádek kódu tak odpovídá nějaké části diagramu
- Prozatím se setkáváme z programy, které načtou vstupní hodnoty, zpracují je a následně nás informují o výsledku
- Velká část aplikací používaných v praxi se chová stejně
- Pokud si nejste jisti, co se v danou chvíli v kódu odehrává načrtněte si současný stav případně jej zkuste zapsat pomocí diagramu

Co provádí následující kód?

```
int a = int.Parse(Console.ReadLine());

if (a > 0)
{
    for (int i = 0; i < a; i++)
    {
        Console.WriteLine("*");
    }
}
else
{
    int b = a * (-2);
    for (int j = b; j > 0; j--)
    {
        Console.WriteLine("-");
    }
}
```

Konstrukce switch – case

```
string day = Console.ReadLine();

switch (day)
{
    case "Pondělí":
        Console.WriteLine("Pondělí je 1. den v týdnu");
        break;
    case "Úterý":
        Console.WriteLine("Úterý je 2. den v týdnu");
        break;
    case "Středa":
        Console.WriteLine("Středa, programování je třeba");
        break;
    case "Čtvrtek":
        Console.WriteLine("Čtvrtek je malý pátek");
        break;
    case "Pátek":
        Console.WriteLine("Pátek je ideálná na párty");
        break;
    case "Sobota":
        Console.WriteLine("Sobota je klidová");
        break;
    case "Neděle":
        Console.WriteLine("V neděli se prostě nepracuje");
        break;
    default:
        Console.WriteLine("Takový den neexistuje!");
        break;
}
```

Rozhodovací proměnná

Co se má stát pokud day == Středa
Break; říká, že poté máme blok opustit

Default: určuje, co se má stát v ostatních případech, mimo definovaných

Brake vs Continue

- Modifikátory běhu programu
- Při dosažení příkazu **brake;** je ukončena obsluha kódu v konkrétním bloku a pokračuje se až za blokem
- Při dosažení příkazu **continue;** je přeskočen zbytek kódu v bloku a pokračuje se v následující iteraci
  - Continue se nejčastěji objevují právě v cyklech