Programování

Algoritmizace, programovací jazyk

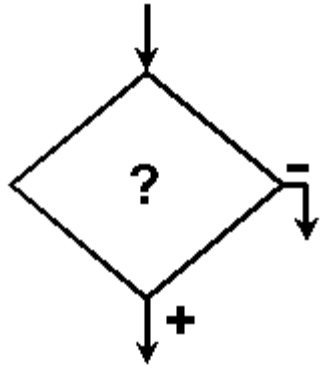
Algoritmizace – postup vytváření algoritmu

- 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

- **Program** = definovaná sekvence složená z jednotlivých příkazů v konkrétním programovacím jazyce

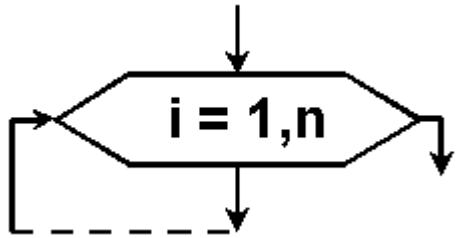
Vývojové diagramy – opakování

Vývojové diagramy



- Rozhodovací blok/podmínka
- Podmínka je umístěna uvnitř symbolu
- Vyhodnocení podmínky se redukuje na možnosti pravda nebo nepravda
  - True/False
- V závislosti zda, je podmínka splněna či ne se dále postupuje v algoritmu
- Pomocí rozhodovacího bloku lze vytvářet cykly s neznámým počtem opakování
  - While – část programu je vykonávaná dokud je podmínka splněna
  - Do While – program se alespoň jednou provede a podmínka je testována až na konci cyklu

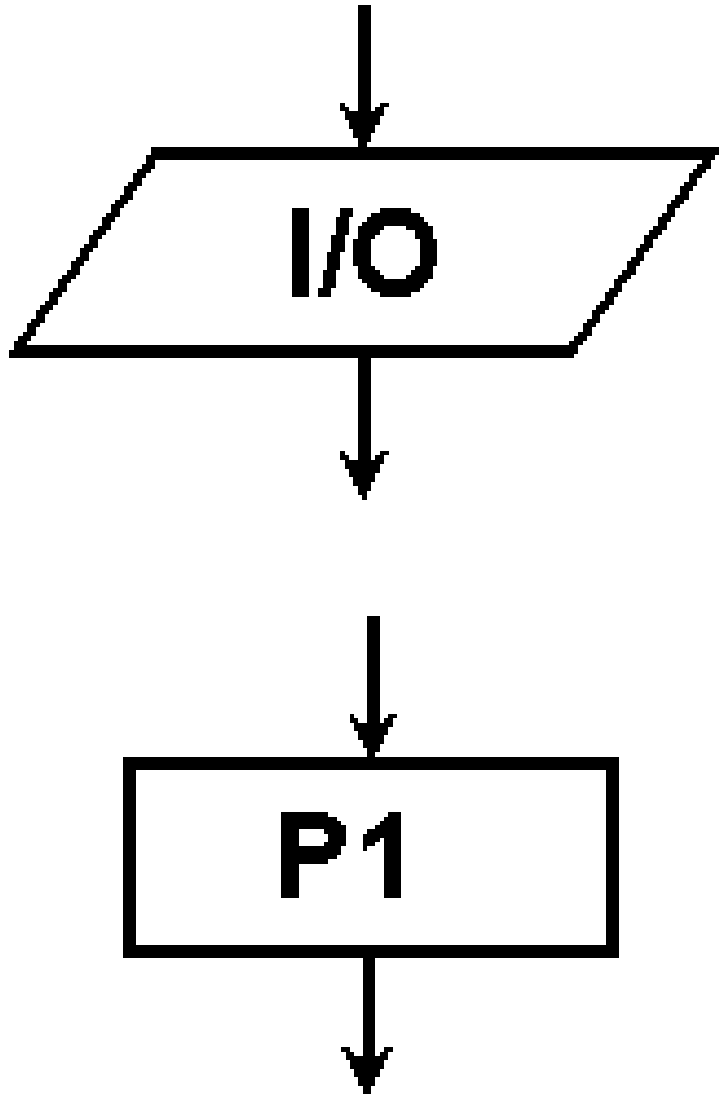
Vývojové diagramy



- Cyklus s pevným počtem opakování
- Počet opakování je hlídaný pomocí proměnné, která je po každém provedení cyklu zvětšena o jedna (lze zvyšovat i o vyšší hodnotu)
- Cykly lze zanořovat – více proměnných hlídající počet průchodů
- V praxi se můžeme setkat i s cykly, kdy řídící proměnnou s každým krokem snižujeme
  - Zvyšování hodnoty – inkrement
  - Snižování hodnoty - dekrement

Vývojové diagramy

- Blok vstup/výstup
- Blok zpracování programu



Vývojové diagramy - procvičení

- Zatlučení hřebíku
- Přechod křižovatky se semaforem
- Zobrazení většího čísla ze dvou hodnot
- Rozhodnutí, zda lze zkonstruovat trojúhelník
- Prohození dvou proměnných
- Součet tří čísel
- Zjištění kolik ze zadaných čísel je sudých a kolik lichých

Programovací jazyk

- = umělý jazyk, který se používá pro definování sekvence programových příkazů, které lze zpracovat na počítači
- Využívá se k implementaci (zhotovení) příslušného algoritmu v konkrétním programovacím jazyku
  - Popis takového algoritmu je ryze konkrétní
- Dělíme podle několika kategorií
  - Míra abstrakce
  - Způsob překladu a spuštění

Dělení programovacích jazyků podle míry abstrakce

- Vyšší programovací jazyky
  - Logická struktura
  - Lépe chápatelné pro člověka
  - Přenositelné na různé platformy OS i HW
  - Do strojového jazyka se překládají za pomocí kompilátoru
  - Java, C#, C++, Kotlin a jiné
- Nižší programovací jazyky
  - Jazyk je blízky strojovému kód
  - Špatně chápatelný pro člověka
  - Často spjatý s konkrétním HW a OS
  - Assembler

Nižší vs. Vyšší programovací jazyky

```
Dump of assembler code for function func4:
=> 0x0000000000400fe2 <+0>:     sub    $0x8,%rsp
   0x0000000000400fe6 <+4>:     mov    %edx,%eax
   0x0000000000400fe8 <+6>:     sub    %esi,%eax
   0x0000000000400fea <+8>:     mov    %eax,%ecx
   0x0000000000400fec <+10>:    shr    $0x1f,%ecx
   0x0000000000400fef <+13>:    add    %ecx,%eax
   0x0000000000400ff1 <+15>:    sar    %eax
   0x0000000000400ff3 <+17>:    lea    (%rax,%rsi,1),%ecx
   0x0000000000400ff6 <+20>:    cmp    %edi,%ecx
   0x0000000000400ff8 <+22>:    jle    0x401006 <func4+36>
   0x0000000000400ffa <+24>:    lea    -0x1(%rcx),%edx
   0x0000000000400ffd <+27>:    callq  0x400fe2 <func4>
   0x0000000000401002 <+32>:    add    %eax,%eax
   0x0000000000401004 <+34>:    jmp    0x40101b <func4+57>
   0x0000000000401006 <+36>:    mov    $0x0,%eax
   0x000000000040100b <+41>:    cmp    %edi,%ecx
   0x000000000040100d <+43>:    jge    0x40101b <func4+57>
   0x000000000040100f <+45>:    lea    0x1(%rcx),%esi
   0x0000000000401012 <+48>:    callq  0x400fe2 <func4>
   0x0000000000401017 <+53>:    lea    0x1(%rax,%rax,1),%eax
   0x000000000040101b <+57>:    add    $0x8,%rsp
   0x000000000040101f <+61>:    retq
End of assembler dump.
(gdb)
```

```
package rentalStore;
import java.util.Enumeration;
import java.util.Vector;

class Customer {
    private String _name;
    private Vector<Rental> _rentals = new Vector<Rental>();

    public Customer(String name) {
        _name = name;
    }
    public String getMovie(Movie movie) {
        Rental rental = new Rental(new Movie("", Movie.NEW_RELEASE), 10);
        Movie m = rental._movie;
        return movie.getTitle();
    }
    public void addRental(Rental arg) {
        _rentals.addElement(arg);
    }
    public String getName() {
        return _name;
    }
```

Další dělení vyšších programovacích jazyků

- **Procedurální** (imperativní) – posloupnost příkazů, která určuje algoritmus
  - Strukturované – dělení algoritmu na menší části tvořící celkovou strukturu
  - Objektově orientované – dodržuje objektové paradigma
- **Neprocedurální** (deklarativní) – vytváření definic, co se má udělat, ne jak
  - Funkcionální – lamba kalkul (Haskell, Lisp, F#)
  - Logické – využití matematické logiky (Prolog, Gödel)

Dělení programovacích jazyků podle způsobu překladu a spuštění

- Kompilované
  - Celé přeloženy kompilátorem a až následně je lze spouštět
  - Vyšší nároky na formální správnost kódu
  - Překlad do strojového nebo do pseudo strojového kódu
- C, C++, C#, Java
  - Interpretované
  - Překlad probíhá za běhu programu
  - Překládají se pomocí interpretu – zajišťuje provádění instrukcí
  - PHP, Python, Perl