Úvod

- Předmět = úvod do programování
- menší úpravy oproti min. semestru
- http://webak.upce.cz/~hajek/iajce/

Literatura:

- Vaše poznámky
- Virius M.: C# pro zelenáče, nakladatelství Neocortex
- Slidy z přednášek <> kniha, ale
  - pomůcka přednášejícího
  - ulehčení vašich záznamů

Cíle

- naučit myslet – transformace reálných problémů na funkční programy
- základy programování

Předpoklady

- logické myšlení
- znalost pojmů: HW/SW, operační systém, soubor, ...
- základní obsluha PC (OS Windows)

Zápočet

- viz cvičení

Zkouška

- části
  - praktická – jednoduchý program v C#, 60 min
  - teoretická – 3 otázky z předem daného seznamu, písemná příprava + diskuze
- ukázka příkladu a seznam otázek viz www
- nelze dělat zkoušku po částech
- žádné pomocné materiály
- zkouší se porozumění, učení nazpaměť nic neřeší

Hodnocení

- praktická část:
  - program musí jít přeložit, spustit a musí pracovat dle zadání
  - dokončení s vyžádanou dopomocí zkoušejícího – nelze dostat 1
- známka subjektivně dle předvedeného výkonu

Poznámky

- začátečník = **nutné** průběžné studium
- angličtina
- účel přednášek = vysvětlit, když něčemu nerozumím, okamžitě se zeptám
- účast nepovinná, když to nepotřebuji, na přednášky nechodím

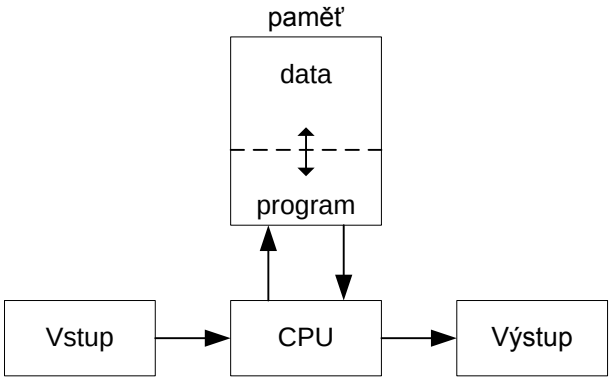
Počítačové jednotky informace

- bit (binary digit)
  - 1 bit = 1 b
  - jedna číslice ve dvojkové soustavě, lze uložit informaci 0/1
- N b – lze uložit 2^N různých hodnot
- Bajt (Byte)
  - 1 B = 8 b
  - 2^8 = 256 hodnot
- násobitelé v digitální technice – základ 2, nikoli 10:

k	$10^3 = 1000$
K	$2^{10} = 1024$
M = K*K	$2^{10} * 2^{10} = 2^{20} = 1\ 048\ 576$
G = K*M	$2^{30} = 1\ 073\ 741\ 824$
T = M*M	2^{40}

Počítač

- = univerzální stroj na automatické zpracování informace.
- Vlastnosti
  - struktura nezávislá od zpracovávaných problémů
  - nutné zvenčí zavést návod na zpracování – program
  - program zpracovává data = užitečná informace jako vstup i výstup (čísla, texty, obrázky, apod.)
- Počítač třídy PC (Personal Computer) – vnitřní uspořádání (architektura)
  - tzv. Von Neumannova architektura – společná paměť pro program i data



        - volitelný poměr program / data
    - HW pohled
        - = elektronické zařízení, pracující pouze ve dvojkové soustavě
        - dvojková soustava = ON/OFF (např. 5V / 0V)

Vstupní a výstupní zařízení

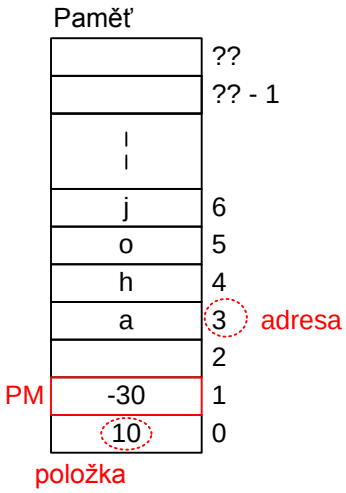
- (I/O = input/output)
- výměna informací mezi počítačem a okolím
- vstupní zařízení = převod informací do číselné (dvojková soustava) podoby
    - klávesnice, scanner, zvuková karta, ...
- výstupní → obráceně do podoby srozumitelné okolí
    - monitor, tiskárna, zvuková karta

Procesor

- (Central Processing Unit)
- „srdce počítače“ – dle pokynů programu zpracovává data
- vykonává pouze jednoduché, elementární operace = instrukce
    - Základní aritmetické operace (+-*/)
    - Logické operace (AND, OR, NOT, XOR)
    - Přesuny dat (CPU ↔ paměť, paměť ↔ paměť).
    - větvení, skoky
- strojový kód SK = program jako posloupnost instrukcí

Paměť

- z uživatelského (programátorského) pohledu
- „schránka na data“; průběžně očíslovaná paměťová místa PM



    - číslo PM = adresa
    - obsah PM = položka

    - velikost PM
        - „z kolika bitů je složeno“
        - skutečnost – dle počítače (8 b, 32 b)
        - IAJCE – „jak budeme potřebovat“ ☺
- základní parametry
    - vybavovací, přístupová doba = doba od vydání příkazu (CPU) do jeho splnění (pamětí)
    - kapacita = množství uložitelné informace (v B)
- paměti v PC
    - vnitřní
        - operační paměť, „RAM“
        - na bázi polovodičů
        - kapacita dnes 100ky MB až 1ky GB
        - vybavovací doba 1-ky ns
        - obsah se po vypnutí napájení ztrácí.
        - obr. Von Neumannova architektura → paměť
    - vnější
        - dlouhodobé uložení dat, obsah zachován i po vypnutí napájení
        - HDD (magnetický záznam), CD-ROM, CD-RW (optický), FLASH Disc (polovodičový)
        - kapacita 100-ky GB, jednotky TB
        - vybavovací doba 1-ky ms
        - obr. Von Neumannova architektura = I/O zařízení

Algoritmus

- další info viz http://cs.wikipedia.org/wiki/Algoritmus

Definice

- = přesný návod či postup, kterým lze vyřešit daný typ úlohy
    - příklad: start motoru automobilu
        1) sešlápnutí spojky
        2) otočení klíčkem do polohy II
        3) čekej, dokud nezhasnou kontrolky
        4) otočení klíčkem do polohy III
        5) jestliže motor do 5 s nenaskočí, čekej 5 s a opakuj vše od bodu 1

        ...
- v programování = teoretický princip řešení problému
    - Příklad: třídící algoritmy = jak seřadit položky (čísla, slova, ...) dle velikosti
        - Bubble sort, Select sort, Quick sort – různý princip, dle toho rychlost třídění, spotřeba paměti, ...

Vlastnosti algoritmů

- Determinismus:
  - Každý krok algoritmu → jednoznačně definován (v rámci „množiny“ příkazů „cíle“ algoritmu)
    Př.: algoritmus pro jízdu autem:
    - řidič: nastartuj → zařaď 1 → ...
    - ne-řidič:
      1) sešlápnutí spojky
      2) otočení klíčkem do polohy II
      ...
    - robot:
      1) levou nohu přesuň na souřadnici X_2,Y_2,Z_2 (spojka)
      2) levou nohou proveď pohyb R, Φ, Θ (sešlápni spojku)
      ...
  - co, jak, kdy se má provést, jak se bude pokračovat.
    Př.: chybný algoritmus
    1) sešlápnutí spojky
    2) otočení klíčkem do jaké polohy
    3) čekej, dokud nezhasnou kontrolky
    4) otočení klíčkem do polohy III
    5) motor naskočí, start dokončen co když ne
  - Elementárnost: každý krok musí být pro „cíl“ algoritmu jednoznačně determinován
- Konečnost: algoritmus musí skončit v konečném počtu kroků.
  - Př.: chybný algoritmus:
    1) sešlápnutí spojky
    ...
    5) jestliže motor do 5 s nenaskočí, čekej 5 s a opakuj vše od bodu 1
    co když nenaskočí nikdy
- Vstupy = data, která jsou mu předány před započetím jeho provádění, nebo v průběhu jeho činnosti.
  - Třídění: data ke třídění
- Výstupy = odpovědi na problém, který algoritmus řeší.
  - Třídění: setříděná data
- Efektivita: posouzení využití daných prostředků algoritmem
  - Př.: Quick Sort je (časově) efektivnější než Bubble Sort (= rychlejší)
  - pojem: složitost algoritmu
- Obecnost: algoritmus řeší obecnou třídu problémů
  - „setřiď 2 položky vs. N položek“

Terminologie v programování

- Programovací jazyk (PJ) = prostředek pro zápis algoritmů, jež mohou být provedeny na počítači
  - jazyk (viz např. http://en.wikipedia.org/wiki/Language) =
    - soubor symbolů, výrazových prostředků a pravidel pro jejich využití
    - prostředek komunikace mezi subjekty (PJ – čitelný pro člověka, zpracovatelný počítačem)
- program = zápis algoritmu ve zvoleném programovacím jazyce
- zdrojový kód ZK (source code)
  - = vlastní text programu
  - posloupnost příkazů v PJ v textových souborech
- překlad (kompilace) programu = proces transformace ZK na strojový kód SK
  - SK na PC s OS fy Microsoft = soubor *.exe (executable), také „program“
- překladač (kompilátor) = program provádějící překlad
- aplikace = „program“ s dalšími podpůrnými soubory (nápověda, DLL knihovny, ...), řešící daný problém
  - často záměna „program“ = aplikace
  - Příklad: Aplikace: MS Word, základ: winword.exe
- spuštění programu (na PC) – OS provede kopii *.exe z vnější paměti do RAM a předá řízení programu
- ladění programu (Debugging) = proces hledání a odstraňování chyb v programech
- instalace programu = proces kopie aplikace do počítače (vnější paměti) z distribučního média (CD, Internet)

Úvod do jazyka C#

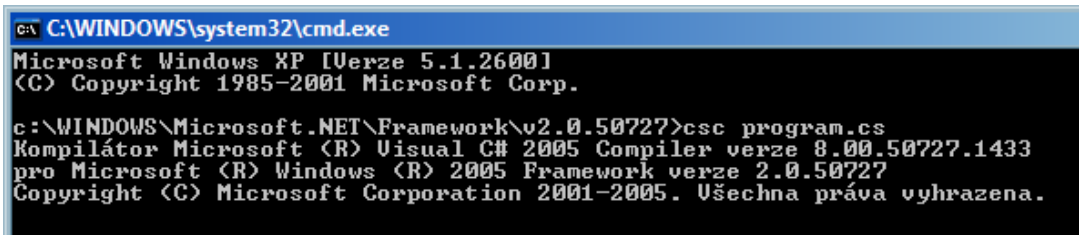
- „Sí šárp“
- vznik cca rok 2000, tvůrce = Microsoft
- standardizován 2003 → ECMA-334 C# Language Specification
- primární PJ pro platformu .NET (.NET Framework, „dot net“)
- .NET =:
  - http://en.wikipedia.org/wiki/.NET_Framework
  - programovací jazyky (C#, C++, Visual Basic.NET]
  - vývojové nástroje (Visual Studio)
  - běhové prostředí (.NET runtime package) = virtuální počítač (nadstavba OS), kde programy pro .NET běží
- Výhody: jednoduchý, bezpečný, plně OOP
- Nevýhody
  - Jen Windows (omezeně Linux → Project Mono, viz http://www.mono-project.com/Main_Page)
  - Pomalý, velká spotřeba paměti

Příklad „Hello world“

- ZK v txt souboru „program.cs“

```
class Program
{
	static void Main(string[] args)
	{
		System.Console.WriteLine("Můj první program!");
	}
}
```

- překladač C#
  c:\WINDOWS\Microsoft.NET\Framework\v???\csc.exe
  - ??? je verze .NET frameworku
- Překlad:
  - csc.exe program.cs



  - výsledek: program.exe
- spuštění
  - program.exe

- „Hello world“ v různých programovacích jazycích
  http://en.wikibooks.org/wiki/List_of_hello_world_programs

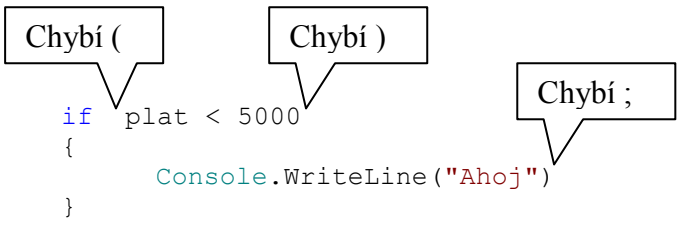
Syntaxe a sémantika PJ

Syntaxe

- = souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu.
- Př.: Neúplná podmínka
  - správná syntaxe

```
if (plat < 5000)
{
	Console.WriteLine("Ahoj");
}
```

- Špatně



- nutno bez výhrady dodržovat – syntakticky nesprávný program nelze zkompilovat

Sémantika

- = význam jednotlivých konstrukcí

```
if (a > 3)
{
      Console.WriteLine("Ahoj");
}
```

„jestliže je a > 3, pak vytiskni `Ahoj`, jinak nedělej nic"

Další prostředky pro zápis algoritmů

- Pseudojazyk
- Vývojový diagram

Pseudojazyk

- (pseudocode)
- PJ s univerzální syntaxí srozumitelnou všem programátorům
- žádná syntaktická pravidla
- Příklad z http://en.wikipedia.org/wiki/Pseudocode
  - jazyk PHP

```
<?php
if (is_valid($cc_number)) {
      execute_transaction($cc_number, $order);
} else {
      show_failure();
}
?>
```

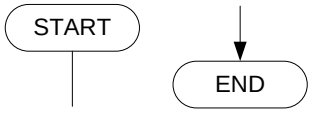
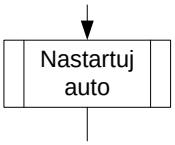
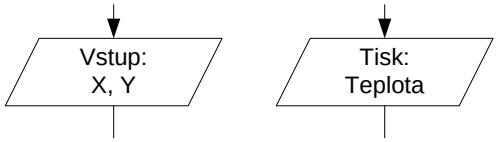
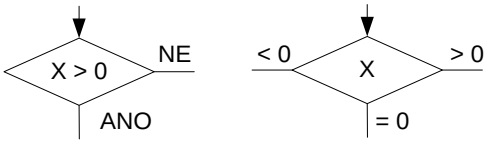
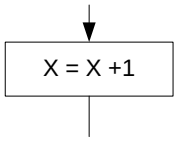
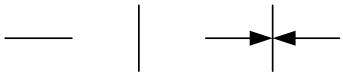
  - pseudojazyk

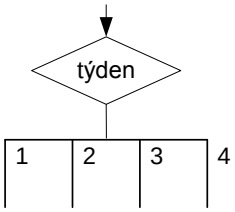
```
if credit card number is valid
    execute transaction based on number and order
else
    show a generic failure message
end if
```

- použití: komunikace mezi programátory napříč jazyky, knihy, ...

Vývojový diagram

- viz ČSN ISO 5807 "Zpracování informací
- = symbolický jazyk pro vizuální znázornění algoritmů
- značky + doplňující textové informace (nemoralizováno)
- výběr nejdůležitějších značek

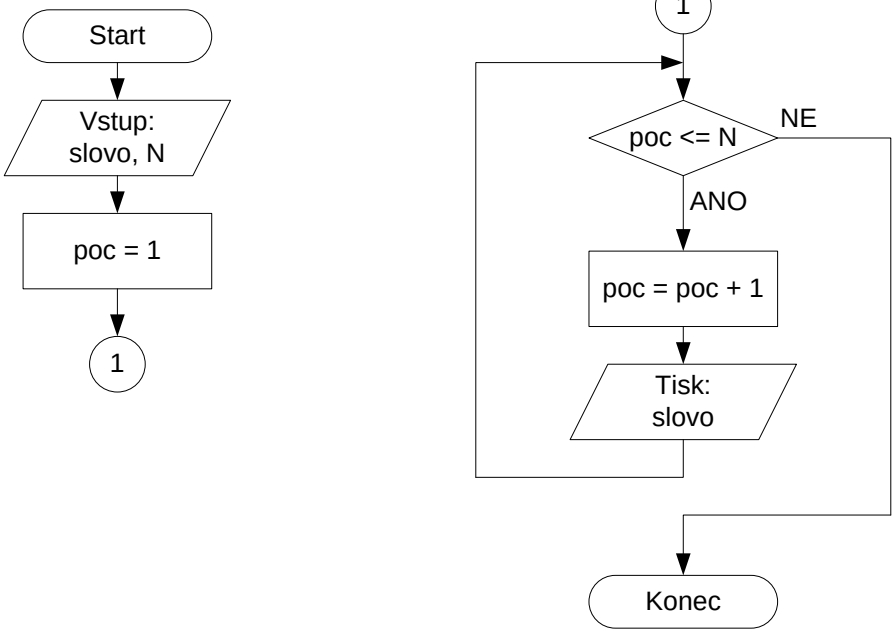
<table>
<tr>
<td>Mezní značka
 
</td>
<td>Předem definovaná činnost
 
</td>
</tr>
<tr>
<td>Vstup Výstup
 
</td>
<td rowspan="2">Rozhodování
 
</td>
</tr>
<tr>
<td>Zpracování
 
</td>
</tr>
<tr>
<td>Spojky
na téže stránce mezi stránkami
 
</td>
<td>spojnice
  </td>
</tr>
</table>



Příklad algoritmizace úlohy

- „Vytiskni N-krát dané slovo
  - `poc` = počítadlo tisků; pomocný záznam nutný pro běh algoritmu

Vývojový diagram



Pseudojazyk

```
pocitadlo = 0
N = 2
slovo = Ahoj
dokud (pocitadlo < N) prováděj
begin
   vypiš slovo
   pocitadlo = pocitadlo + 1
end
```

Programovací jazyk

C#	Pascal

`int pocitadlo = 0;`
`int N = 2;`
`string slovo = "Ahoj";`
`while (pocitadlo < N)`
`{`
`  Console.WriteLine(slovo);`
`  pocitadlo++;`
`}`
	`var pocitadlo, N: integer;`
	`var slovo: string;`
	`pocitadlo := 0;`
	`N := 2;`
	`slovo := "Ahoj";`
	`while (pocitadlo < N) do`
	`begin`
	`  writeln(slovo);`
	`  pocitadlo := pocitadlo + 1;`
	`end`