

1. Úvod

Obsah předmětu =komparativní studie program. jazyků

- Vývoj programovacích jazyků, styly a vlastnosti
- Logické programování - Prolog
- Funkcionální programování - Lisp
- Datové abstrakce a modulární programování
- Objektově orientované programování - Java, Obj.Pascal, Ada95, C++
- Vizuální programování Eclipse-visual editor
- Zpracování výjimečných situací
- Paralelní programování
- Porovnání vlastností imperativních jazyků
- Úvod do překladačů

Podmínky absolvování

Podmínky udělení zápočtu:

- Předvedení programu v Prologu
 - Předvedení programu v Lispu
 - Zpracování referátu z Delphi nebo
 - Zpracování referátu s použitím vizuálních neklasických konstrukcí - archivace

Min 20b. Max 40b

Forma zkoušky písemný test s příklady - archivace

Min 30b. Max 60b

Termíny zkoušky:

řádný

1.opravný

2. opravný (poslední)

Vývoj programovacích jazyků

Konec 40. let – odklon od strojových kódů

Pseudokódy:

- Pseudooperace aritm. a matem. funkcí
- Podmíněné a nepodmíněné skoky
- Autoinkrement. registry pro přístup k polím

Vývoj programovacích jazyků

50. léta

- první definice vyššího programovacího jazyka (efektivita návrhu programu)
 - FORTRAN (formula translation - Backus), vědeckotechnické výpočty, komplexní výpočty na jednoduchých datech, pole, cykly, podmínky
 - COBOL (common business lang.), jednoduché výpočty, velká množství dat, záznamy, soubory, formátování výstupů
 - ALGOL (algorithmic language - Backus, Naur), předek všech imperativních jazyků, bloková struktura, rekurze, volání param. hodnotou, deklarace typů
 - nové idee - strukturování na podprogramy, přístup ke globálním datům (Fortan), bloková struktura, soubory, ...
 - stále živé - Fortran90, Cobol
-

Vývoj programovacích jazyků

první polovina 60. let

- začátek rozvoje neimperativních jazyků
- LISP (McCarthy) - založen na teorii rekurzivních funkcí, první funkcionální jazyk, použití v UI (symbolické manipulace)
- APL - manipulace s vektory a s maticemi
- SNOBOL (Griswold) - manipulace s řetězci a vyhledávání v textech, podporuje deklarativní programování
- vzniká
- potřeba dynamického ovládání zdrojů,
- potřeba symbolických výpočtů
- -----

Vývoj programovacích jazyků

pozdní 60. léta

- **IBM snaha integrovat úspěšné koncepty všech jazyků - vznik PL/1 (moduly, bloky, dynamické struktury)**
 - **nové prvky PL/1 - zpracování výjimek, multitasking. Nedostatečná jednotnost konstrukcí, komplikovanost**
 - **ALGOL68 - ortogonální konstrukce, první jazyk s formální specifikací (VDL), uživatelsky málo přívětivý, typ reference, dynamická pole**
 - **SIMULA67 (Nygaard, Dahl) - zavádí pojem tříd, hierarchie ke strukturování dat a procedur, ovlivnila všechny moderní jazyky, corutiny**
 - **BASIC (Kemeny) - žádné nové konstrukce, určen začátečníkům, obliba pro efektivnost a jednoduchost, interaktivní styl programování**
 - **Pascal (Wirth) - k výuce strukturovaného programování, jednoduchost a použitelnost na PC zaručily úspěch**
-

Vývoj programovacích jazyků

70. léta

- **důraz na bezpečnost a spolehlivost**
 - **abstraktní datové typy, moduly, typování, dokazování správnosti, práce s výjimkami**
 - **CLU (datové abstrakce), Mesa (rozšíření Pascalu o moduly), Concurrent Pascal, Euclid (rozšíření Pascalu o abstraktní datové typy)**
 - **C (Ritchie) - efektivní pro systémové programování, efektivní implementace na různých počítačích, slabé typování**
 - **Scheme - rozšířený dialekt LISPU**
 - **PROLOG (Colmerauer) - první logicky orientovaný jazyk, používaný v UI a znalostních systémech, neprocedurální, „inteligentní DBS odvozující pravdivost dotazu“**
-

Vývoj programovacích jazyků

80. léta

- **Modula2 (Wirth)** - specifické konstrukce pro modulární programování
 - další rozvoj funkcionálních jazyků - **Scheme (Sussman, Steele, MIT)**, **Miranda (Turner)**, **ML (Milner)** - typová kontrola
 - **ADA (US DOD)** syntéza vlastností všech konvenčních jazyků, moduly, procesy, zpracování výjimek
 - průlom objektově orientovaného programování - **Smalltalk (Key, Ingalls, Xerox:** Datová abstrakce, dědičnost, dynamická vazba typů), **C++ (Stroustrup 85- C a Simula)**
 - další OO jazyky - **Eiffel (Mayer)**, **Modula3**, **Oberon (Wirth)**
 - **OPS5, CLIPS** - pro zpracování znalostí
-

Vývoj programovacích jazyků

- 90. léta
- jazyky 4.generace, QBE, SQL - databázové jazyky
- Java (SUN) - mobilita kódu na webu, nezávislost na platformě
- vizuální programování (programování ve windows) - Delphi, Visual Basic, Visual C++
- jazyky pro psaní CGI skriptů:
 - ✓ Perl (Larry Wall - Pathologically Eclectic Rubbish Lister) - nástroj pro webmastery a administrátory systémů,
 - ✓ JavaScript - podporován v Netscape i v Explorer,
 - ✓ VBScript,
 - ✓ PHP – freeware
- 2000 C Sharp

Vývoj programovacích jazyků (Fortran)

Fortran 0 – 1954 neimplementován

Fortran 1 – 1957 (hlavní kriterium=efektivita)

- Bez dynamické paměti
- Rychlý přístup k polím (posttest for cyklus)
- Jména max 6 znaková
- Formátovaný I/O
- Uživatelské podprogramy
- Třícestná selekce (aritmetický IF)
- 18 člověkoroků implementace

Vývoj programovacích jazyků (Fortran)

Fortran II – 1958

- Separátní překlad

Fortran IV – 1960-1962

- Explicitní deklarace typů
- Logický IF příkaz
- Podprogramy s parametry
- ANSI standard 1966

Vývoj programovacích jazyků (Fortran)

Fortran 77 - 1978

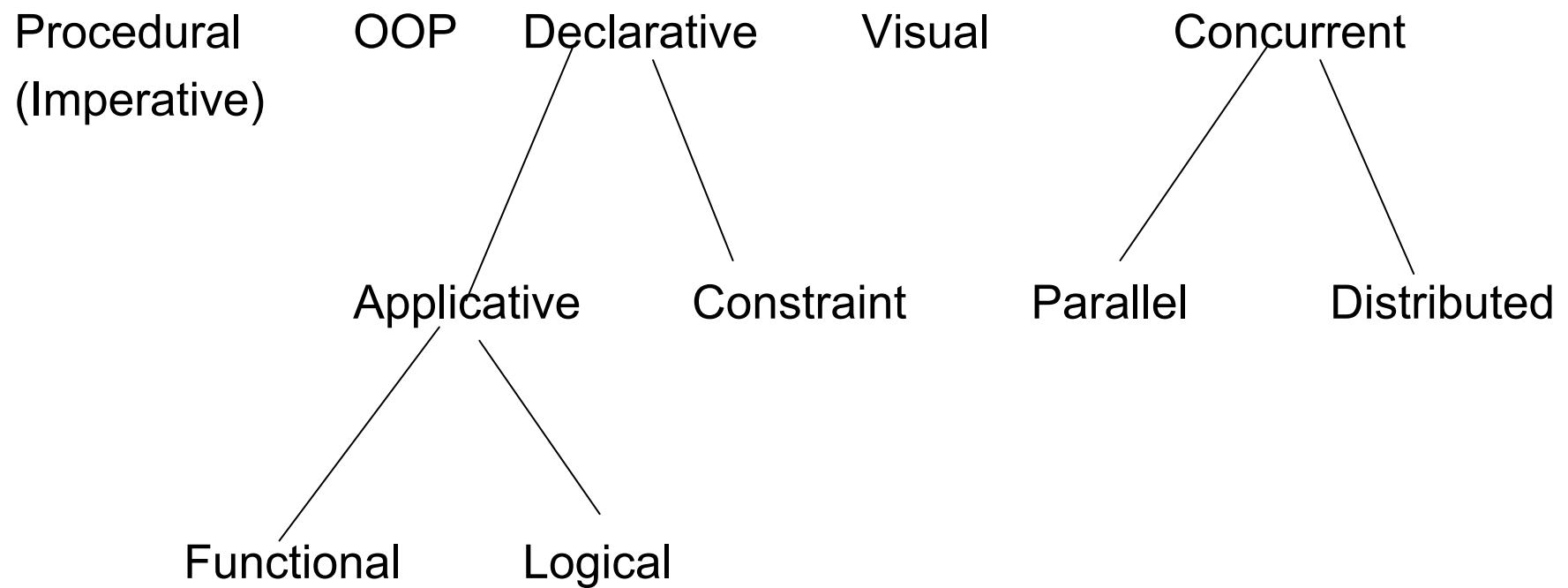
- Manipulace s řetězci znaků
- IF THEN ELSE příkaz
- Log. hodnotou řízený cyklus

Fortran 90 - 1990

- Moduly, Kontrola typů parametů
- Dynamická pole, Ukazatele
- Case příkaz
- Rekurze

Paradigmata programování

- Paradigma = souhrn způsobů formulace problémů, metodologických prostředků řešení, metodik zpracování a pod.
- Procedurální (imperativní) programování
- Objektově orientované programování
- Deklarativní programování
- Aplikativní programování
- Funkcionální programování
- Logické programování
- Programování ohraničeními (constraint progr.)
- Vizuální programování
- Souběžné programování
 - paralelní
 - distribuované



Koexistence paradigmat

```
#include <stdio.h>
```

```
int nsd(int u, int v)
{ if (v == 0) return u;
  else return nsd (v, u % v);
}
```

```
main()
{ int x, y;
  printf("vstup dvou celych cisel:\n");
  scanf("%d%d",&x,&y);
  printf("nsd z %d a %d je %d\n",x,y,nsd(x,y));
  getchar(); getchar();
}
```

```

import java.io.*;
class IntWithNsd
{ public IntWithNsd( int val ) { value = val; }
  public int getValue() { return value; }
  public int nsd ( int v )
  { int z = value;
    int y = v;
    while ( y != 0 )
    { int t = y;
      y = z % y;
      z = t;
    }
    return z;
  }
  private int value;
}
class NsdProg
{ public static void main (String args[])
  { System.out.println(" vstup dvou celych cisel:");
    BufferedReader in =
      new BufferedReader(new InputStreamReader(System.in));
    try
    { IntWithNsd x = new IntWithNsd(Integer.parseInt(in.readLine()));
      int y = Integer.parseInt(in.readLine());
      System.out.print(" nsd z " + x.getValue() + " a " + y + " je ");
      System.out.println(x.nsd(y));
    }
    catch ( Exception e )
    { System.out.println(e);
      System.exit(1);
    } } }

```

```
nsd(U, V, U) :- V = 0 .  
nsd(U, V, X) :- not(V = 0),  
    Y is U mod V,  
    nsd(V, Y, X).
```

```
(defun nsd (u v)
  (if (= v 0) u
    (nsd v (mod u v)))))
```

```

with TEXT_IO; use TEXT_IO;
procedure nsd_prog is
    function nsd (u, v: in integer) return integer is
        y, t, z: integer;
    begin
        z := u;
        y := v;
        loop
            exit when y = 0;
            t := y;
            y := z mod y;
            z := t;
        end loop;
        return z;
    end nsd;
    package INT_IO is new INTEGER_IO(Integer);
    use INT_IO;
    x: Integer;
    y: Integer;
begin
    put_line("vstup dvou celych cisel:"); get(x); get(y);
    put("nsd z "); put(x); put(" and "); put(y);
    put(" je "); put(nsd(x,y));
    new_line;
end nsd_prog;

```

Globální kriteria na programovací jazyk

1. Spolehlivost
2. Efektivita
3. Strojová nezávislost
4. Čitelnost a vyjadřovací schopnosti
5. Řádně definovaná syntax a sémantika
6. Úplnost v Turingově smyslu

Globální kriteria na programovací jazyk

Ad 1

Typová kontrola

Zpracování výjimečných situací

Ad 2

Ef. Překladu

Ef. výpočtu

Globální kriteria na programovací jazyk

Ad 4

- jednoduchost (kontra př.: $C=C+1$; $C+=1$; $C++$; $++C$)
- Ortogonalita (malá množina primitivních konstrukcí, z té lze kombinovat další konstrukce. Všechny kombinace jsou legální) kontra př. v C:
 - struktury mohou být funkční hodnotou
 - Pole nemohou
- Strukturované příkazy
- Strukturované datové typy
- Podpora abstrakčních prostředků

Globální kriteria na programovací jazyk

Ad 4

- Strojová čitelnost
 - = existence algoritmu překladu s lineární časovou složitostí
 - = bezkontextová syntax
 - Humánní čitelnost
 - silné odvisí od způsobu abstrakcí
 - abstrakce dat
 - abstrakce řízení
- čitelnost kontra zapisovatelnost

Globální kriteria na programovací jazyk

Ad 5

Syntax = forma či struktura výrazů, příkazů a programových jednotek

Sémantika = význam výrazů, příkazů a programových jednotek

Definici jazyka potřebují
návrháři jazyka
implementátoři
uživatelé

Globální kriteria na programovací jazyk

Ad 6

- Turingův stroj = jednoduchý ale neefektivní počítač použitelný jako formální prostředek k popisu algoritmu
- Programovací jazyk je úplný v Turingově smyslu, jestliže je schopný popsat libovolný výpočet (algoritmus)
- Co je potřebné pro Turingovu úplnost?

Téměř nic: Stačí

- ✓ celočíselná aritmetika a
- ✓ celočíselné proměnné spolu se sekvenčně prováděnými příkazy zahrnujícími
- ✓ přiřazení a
- ✓ cyklus (While)

Syntax

Formálně je jazyk množinou vět
Věta je řetězcem lexémů (terminálních symbolů)

Syntax lze popsat:

- Rozpoznávacím mechanismem - automatem (užívá jej překladač)
- Generačním mechanismem – gramatikou (to probereme)

Syntax

Gramatika je čtverice (N, T, P, S)

Bezkontextová gramatika

Backus Naurova forma (BNF) – používá metajazyk

<program> → <seznam deklaraci> ; <prikazy>

<seznam deklaraci> →

<deklarace> |

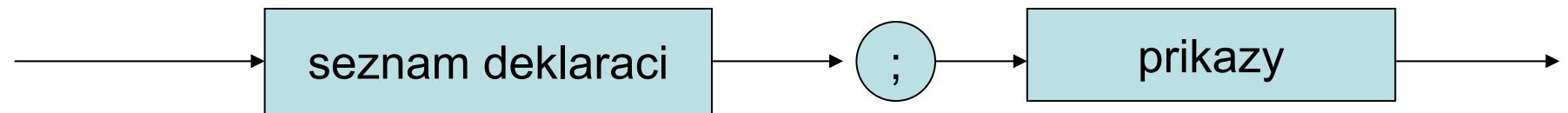
<deklarace>;<seznam deklaraci>

<deklarace> → <spec. typu> <sez. promennych>

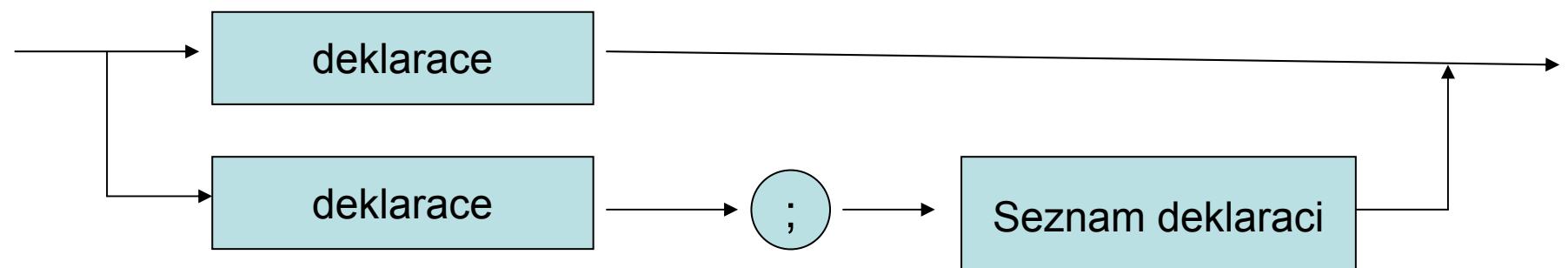
Syntax

Syntaktické diagramy

program



Seznam deklarací



Syntax

- Derivace

<program> => <seznam deklaraci> ; <prikazy> =>
⇒ <deklarace> > ; <prikazy> => int x ; <prikazy> =>
⇒ int x; <prikaz> ; <prikazy> => int x ; read(x); <prikazy> =>
⇒ int x ; read(x) ; <prikazy> => ...

- Derivační strom

Sémantika

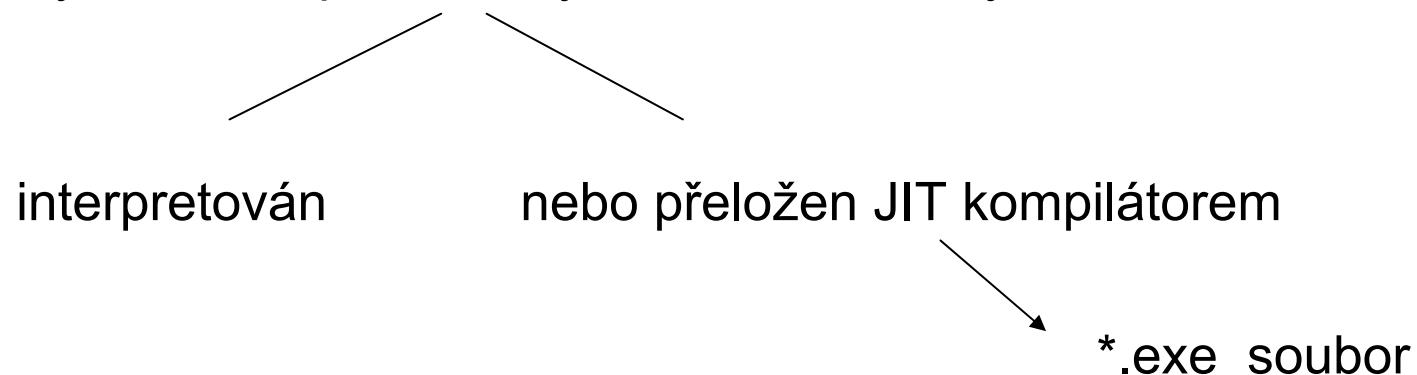
- A. Statická sémantika
- B. Dynamická sémantika

ad B

- Operační sémantiky
- Axiomatické sémantiky
- Denotační sémantiky

Překlad jazyka

- Kompilátor: dvoukrokový proces překládá zdrojový kód do cílového kódu. Následně uživatel sestaví a spustí cílový kód
- Interpret: jednokrokový proces, „zdrojový kód je rovnou prováděn“
- Hybridní: např. Java Byte-code – soubory *.class



Klasifikace chyb

- Lexikální - např nedovolený znak
 - Syntaktické -chyba ve struktuře
 - Statické sémantiky –např ne definovaná proměnná, chyba v typech. Způsobené kontextovými vztahy. Nejsou syntaktické
 - Dynamické sémantiky – např dělení 0. Nastávají při výpočtu, neodhalitelné při překladu. Nejsou syntaktické
 - Logické – chyba v algoritmu
-
- ❖ Kompilátor schopen nalézt lexikální a syntaktické při překladu
 - ❖ Statické sémantiky chyby může nalézt až před výpočtem
 - ❖ Nemůže při překladu nalézt chyby v dynamické sémantice, projeví se až při výpočtu
 - ❖ Žádný překladač nemůže hlásit logické chyby
 - ❖ Interpret obvykle hlásí jen lexikální a syntaktické chyby když zavádí program

Klasifikace chyb př. Java

```
public int nsd ( int v# ) {  
    int a = value  
    b = v;  
    while ( b != 0 ) {  
        int p = b;  
        b = a % b;  
        a = p;  
    }  
    return b;  
}
```

Zdroje

- Catalog of Free Compilers and Interpreters. <http://www.idiom.com/free-compilers/>
- Free electronic book <http://www.mindview.net/Books>
- <http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/user/dst/www/LispBook/index.html>
- <http://www.lisp.org/alu/res-lisp-education.clp>
- <http://clisp.cons.org>
- Raphael Finkel: Advanced Programming Language Design
- <http://cseng.aw.com/book/0,3828,0805311912,00.html>
- R.W.Sebesta: Concepts of Programming Languages
- M.L.Scott: Programming Languages Pragmatics
- Ježek, Racek, Matějovič: Paralelní architektury a programy
- Herout: Učebnice jazyka Java
- Návrat, Bieliková: Funkcionálne a logické programovanie
- Kolář: Jazyky pro umělou inteligenci
- <http://www.hitmill.com/programming/dotNET/csharp.html#courses>
- ...