

Základy algoritmizace

13. Přehled prog. jazyků

doc. Ing. Jiří Vokřínek, Ph.D.

Katedra počítačů

Fakulta elektrotechnická

České vysoké učení technické v Praze

Základy algoritmizace

- Dnes:
 - Programovací jazyky
 - Příklad – jazyk C
 - Příklad – jazyk Java

Programovací jazyky

Programovací jazyk

- Existuje množství programovacích jazyků
- Nelze říci, že jeden jazyk je lepší než druhý

V podstatě jsou všechny ekvivalentní

- Můžeme, ale říci, že některé jazyky se hodí na konkrétní úlohy
- Základní dělení:
 - Vysoko-úrovňové a nízko-úrovňové

Liší se mohutností množiny primitiv

- Obecné a speciální (určené pro konkrétní aplikace)
- Interpretované a překládané
- Dle typu: imperativní (procedurální), funkcionální, logické (deklarativní), objektově-orientované

Programovací jazyk

■ Nízko-úrovňové

- Platformě závislé, specifika systému
- Explicitní práce s pamětí
- „náročnější“ vývoj
- Vyšší efektivita programu
- Strojový kód, jazyk symbolických adres (assembler,) jazyk C

■ Vysoko-úrovňové

- Platformě nezávislé
- Využití abstraktních datových typů
- „snadnější“ vývoj
- Nižší efektivita programu
- Jazyk C, Java, Python, ...

Programovací jazyk

- Nízko-úrovňové

- Příklad: funkce ve strojovém kódu x86 pro výpočet n -tého prvku Fibonacciho posloupnosti

```
8B542408 83FA0077
06B80000 0000C383
FA027706 B8010000
00C353BB 01000000
B9010000 008D0419
83FA0376 078BD98B
C84AEBF1 5BC3
```

- Příklad: totéž v jazyku symbolických adres

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

@@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

@@:
    push ebx
    mov ebx, 1
    mov ecx, 1

@@:
    lea eax, [ebx+ecx]
    cmp edx, 3
    jbe @f
    mov ebx, ecx
    mov ecx, eax
    dec edx
    jmp @b

@@:
    pop ebx
    ret
```

Programovací jazyk

■ Příklad: jazyk C

```
#include<stdio.h>
int main() {
    int k,r;
    long int i=0l,j=1,f;
    //Taking maximum numbers form user
    printf("Enter the number range:");
    scanf("%d",&r);

    printf("FIBONACCI SERIES: ");
    printf("%ld %ld",i,j); //printing firts two values.

    for (k=2;k<r;k++) {
        f=i+j;
        i=j;
        j=f;
        printf(" %ld",j);
    }

    return 0;
}
```

Programovací jazyk

- Další třídění:
 - imperativní (procedurální), funkcionální, logické (deklarativní), objektově-orientované
- Příklad: Lisp

```
(defun fibonacci (n)
  (do ((a 1 b)
      (b 1 (print (+ a b)))
      (n n (1- n)))
      ((zerop n) b)))
```

- Příklad: Prolog

```
fib(0, 0).
fib(1, 1).
fib(N, NF) :-
  A is N - 1, B is N - 2,
  fib(A, AF), fib(B, BF),
  NF is AF + BF.
```


Programovací jazyk

- Příklad: jazyk Java

```
public class FibonacciIterative {
    public static int fib(int n) {
        int prev1=0, prev2=1;
        for (int i=0; i<n; i++) {
            int savePrev1 = prev1;
            prev1 = prev2;
            prev2 = savePrev1 + prev2;
        }
        return prev1;
    }
}
```

Programovací jazyk

- (ne) typované
 - Každá operace přijímá jen data daného typu
vs. operace přijímá sekvenci bytů
- Slabě typované
 - Umožňují „konvertovat“ typ proměnné
- Silně typované
 - Špatný typ vyvolá chybu v programu
 - Jsou typově „bezpečné“
- Staticky typované
 - Typ je určen přímo v kódu (C, Java, ...)
- Dynamicky typované
 - Typ se určuje za běhu programu podle hodnoty (Python, ...)

Programovací jazyk

■ Kompilované

- Program se přeloží do strojového kódu pomocí překladače
- Staticky nebo dynamicky se propojí s knihovnamy
- Spustí se na cílové platformě
- Typicky rychlejší, optimalizován pro danou platformu

■ Interpretované

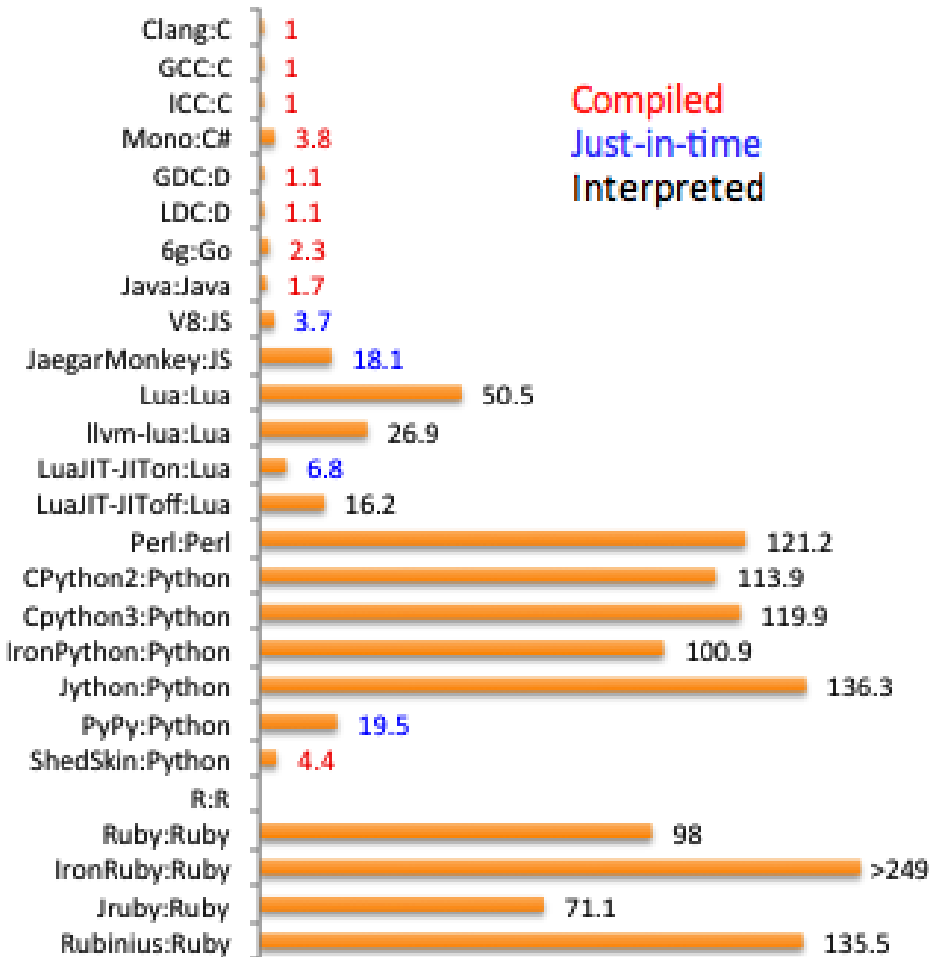
- Program se přímo vykonává pomocí virtuálního počítače – interpretu
- Samostatný program není spustitelný
- Typicky pomalejší, závislý na interpretu
- Vhodné pro skriptování

■ Kompilované „just-in-time“

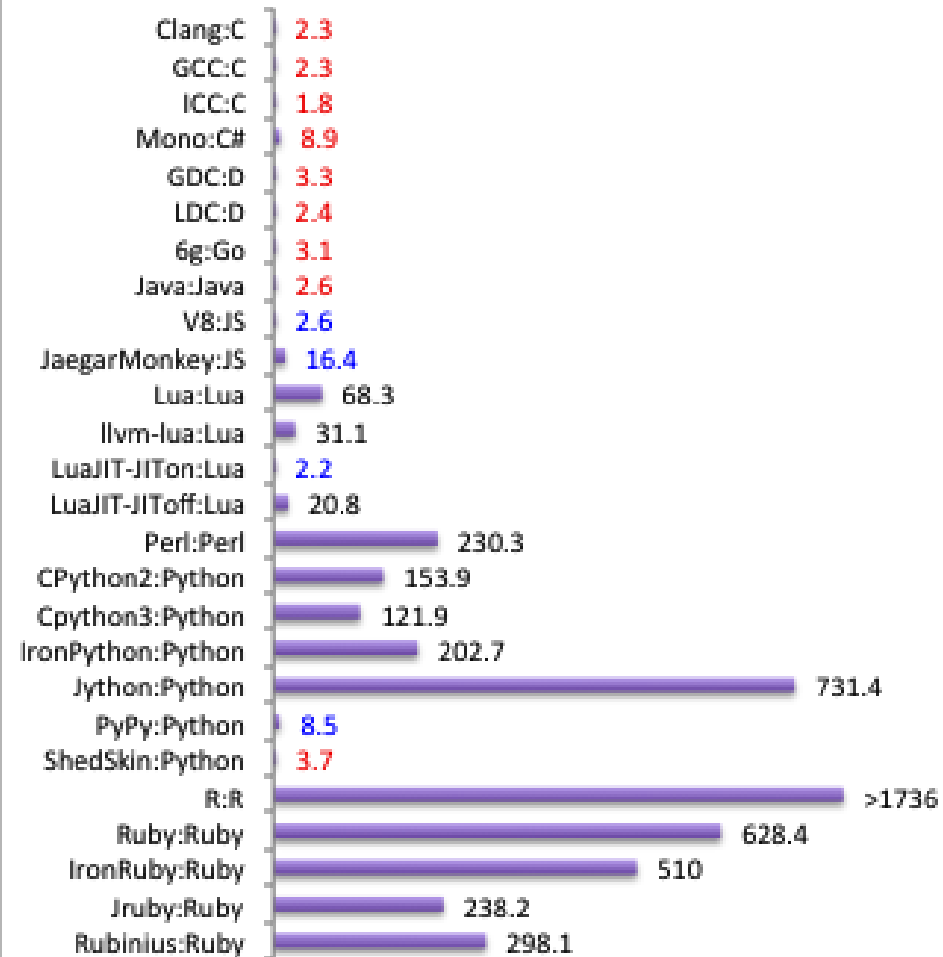
- K překladu dochází při spuštění (příp. v průběhu)

Programovací jazyky

Sudoku solving (CPU sec)



Matrix multiplication (CPU sec)



<https://attractivechaos.github.io/plb/>

Programovací jazyk C

Programovací jazyk C

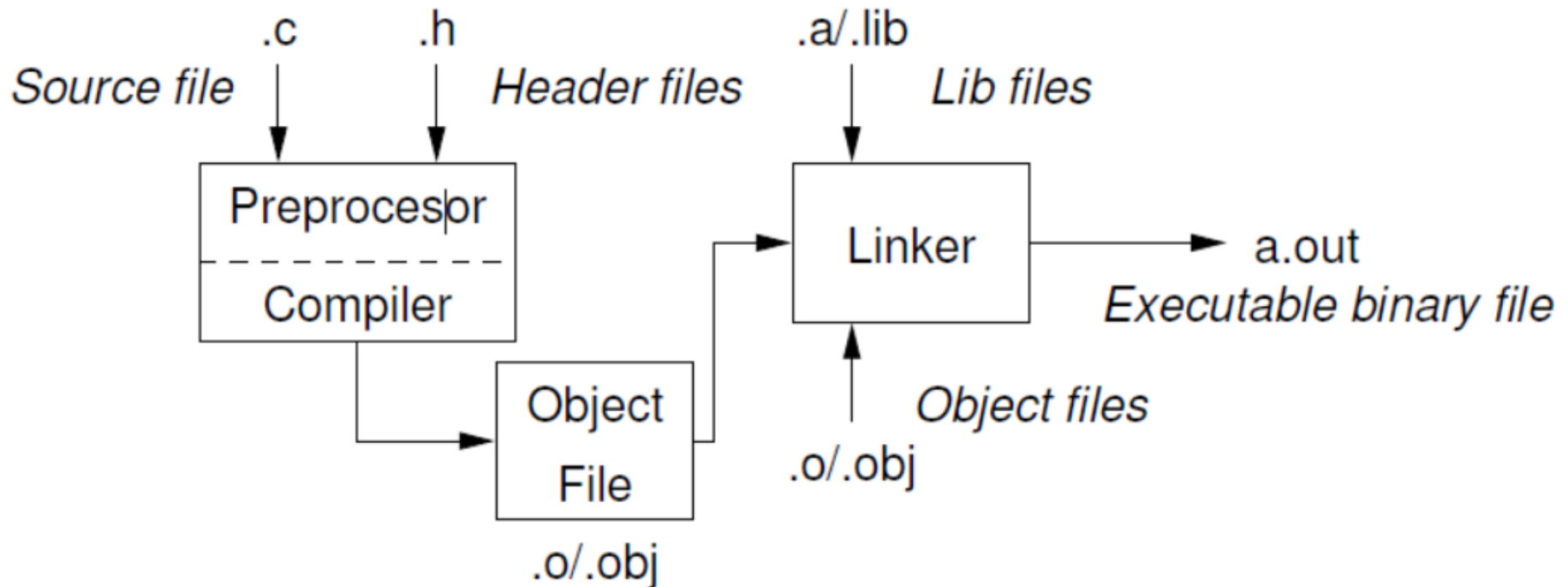
- Nízko-úrovňový programovací jazyk
- Systémový programovací jazyk (operační systém)
- Vhodný pro vestavné systémy
- Téměř vše nechává na programátorovi (inicializace proměnných, správa paměti, ...)
- Má blízko k využití hardwarových zdrojů
- Klíčové pro správné chování je zacházení s pamětí

- Vysoko-úrovňový jazyk
 - Objektivě orientovaná „nadstavba“ C++
 - „Interpretovaná“ varianta C# (.NET)

Programovací jazyk C

■ Překlad

- Preprocesor pro předzpracování zdrojových kódů
- Kompilace do objektového souboru
- Sestavení spustitelného programu z dílčích objektových souborů a odkazovaných knihoven (možno i dynamicky)



Programovací jazyk C

■ Preprocessor

- Umožňuje definovat makra a přizpůsobit překlad aplikace kompilačnímu prostředí
- Výstupem je textový (zdrojový) soubor

■ Compiler

- Překládá zdrojový kód do strojové podoby
- Nativní, platformě závislý kód

■ Linker

- Sestavuje program z přeložených souborů do podoby výsledné aplikace
- Může obsahovat dynamická volání na knihovní funkce, služby OS, atp.

Programovací jazyk Java

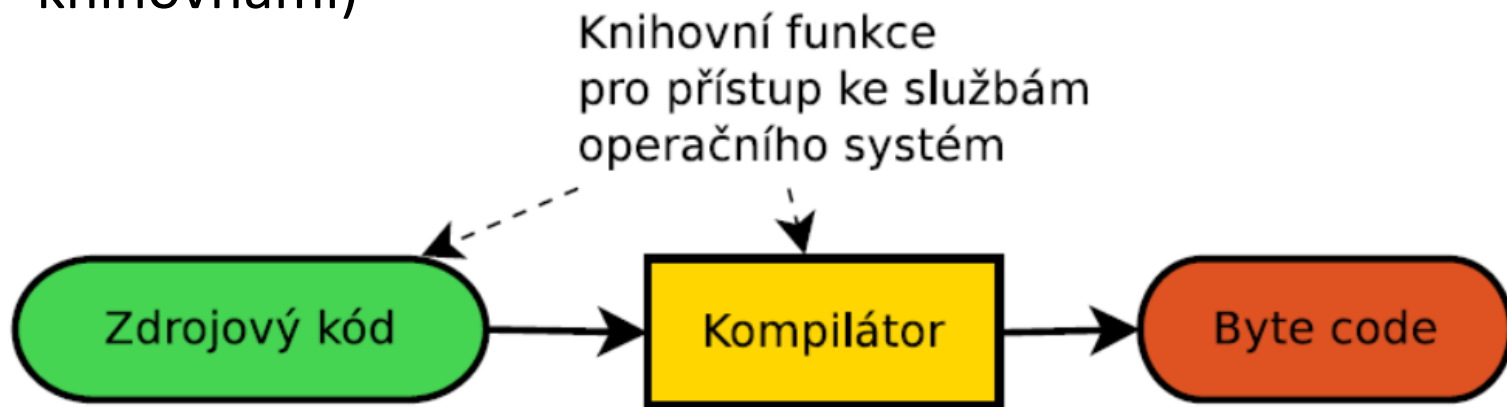
Programovací jazyk Java

- Obecný vyšší imperativní a objekově orientovaný jazyk
- Překládaný, zaměřený na přenositelnost zdrojových kódů i přeložených binárních souborů
- Přeložený binární soubor je interpretován virtuálním strojem
- Součástí základního vývojového prostředí je i bohatý soubor knihovných funkcí
- Zaměřen na efektivní a robustní programování

Programovací jazyk Java

■ Překlad

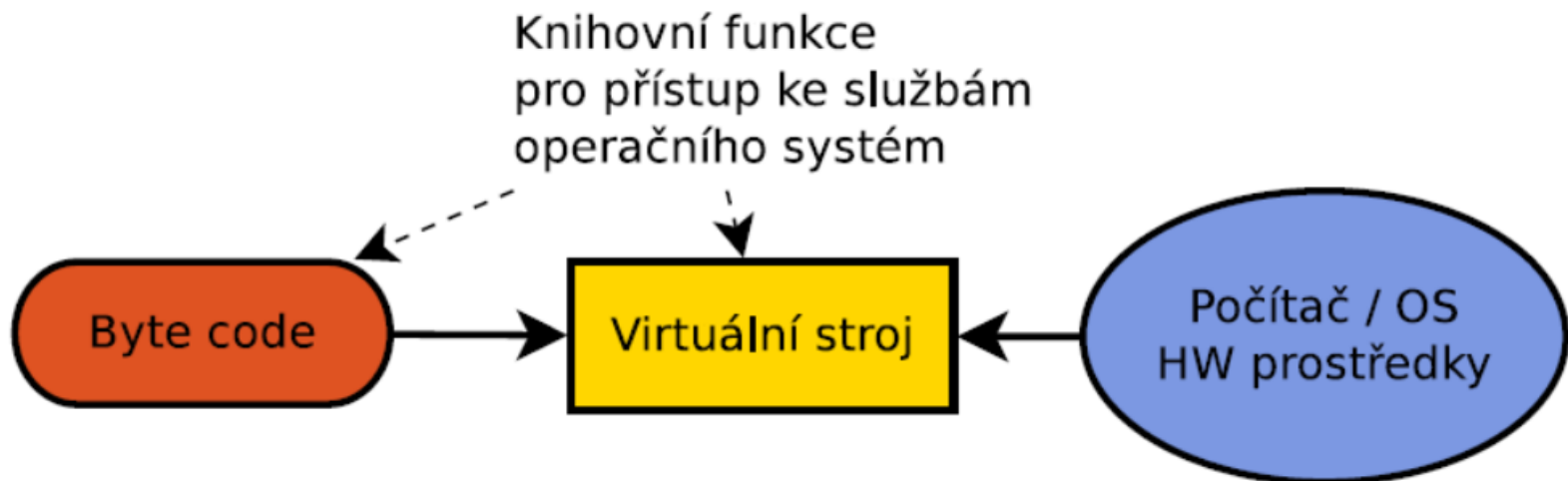
- Zdrojové kódy jsou zapisovány v textových souborech s koncovkou `.java`
- Zdrojové soubory jsou překladačem (`javac`) přeloženy do binárního kódu (byte code) uložených v souborech s koncovkou `.class`
- Byte code je platformě nezávislý binární kód, který je interpretován pomocí Java Virtual Machine (JVM)
- Je možný i přímý překlad do nativního kódu (problém s knihovnami)



Programovací jazyk Java

■ Interpretace

- Virtuální stroj interpretuje přeložený byte code (java)
- JVM je abstraktní virtuální počítač nezávislý na prog. jazyce
- Zajišťuje dynamické linkování a přístup k HW a OS
- Zajišťuje správu paměti pomocí Garbage Collector
- Poskytuje interpretaci nebo just-in-time překlad
- Umožňuje dynamické (runtime) optimalizace
- Potenciálně efektivnější než statická optimalizace/překlad



Programovací jazyk Java

■ Prostředí Java

- **JDK** (Java Development Kit) – základní vývojové prostředí, knihovny funkcí, překladač `javac`, jeho součástí je i JRE
- **JRE** (Java Runtime Environment) – základ prostředí Java pro spouštění programů, obsahuje virtuální stroj
- **JVM** (Java Virtual Machine) – virtuální stroj pro spouštění přeložených programů

- **JAR** (Java ARchive) – archív Java souborů, typicky množiny přeložených tříd (`.class`) doplněných textovým popisem (Manifest), kterou třídu spustit, slouží pro snadnější distribuci a spouštění programů o více souborech

V podstatě je to ZIP archív

Programovací jazyk Java

- V příštím semestru se naučíme
 - Syntaxi a použití jazyka Java
 - Pracovat s JVM
 - Vybrané knihovní funkce prostředí
- A hlavně
 - Osvojíme si objektově orientované programování
- ... a vyvineme (první) složitější aplikaci
- Zároveň se naučíme testovat software a pracovat s databázemi

Základy algoritmizace

- Dnes:
 - Programovací jazyky
 - Příklad – jazyk C
 - Příklad – jazyk Java

Příště zkouška 😊