

Základní pojmy

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 1

A0B36PR1 – Programování 1

Část 1 – Organizace předmětu

Informace o předmětu

Přednášky

Cvičení a domácí úkoly

Hodnocení předmětu a zkouška

Část 2 – Programování a výpočty

Výpočty a výpočetní prostředky

Program a programovací jazyk

Část 3 – Programovací jazyk Java

Programovací jazyk Java

Příklad jednoduchého programu

Základní datové typy

Část I

Organizace předmětu

Organizace a hodnocení předmětu

■ A0B36PR1 – Programování 1

- Rozsah: 2p+2c; Zakončení: Z,ZK; Kredity: 6;

Z – zápočet, ZK – zkouška

- Průběžná práce v semestru - domácí úkoly a testy

- Implementační a případně ústní zkouška

Schopnost samostatné práce na počítačích v učebnách

- Docházka na cvičení a odevzdání domácích úloh

Samostatná práce

- **Supervize práce v počítačové učebně**

- Pátek od 11:00 až 12:30 místnost T2:H1-131 (9.10.-6.11.2015)

Pro osvojení si základních návyků používání počítačů v učebně a řešení programovacích úloh

- **Alternativní absolvování předmětu pro zkušené**

Předmět A4B36ACM

Základní zdroje a webové stránky

A0B36PR1 - Programování 1

- Webové stránky předmětu

<https://cw.fel.cvut.cz/wiki/courses/a0b36pr1/>

- Odevzdávání domácích úkolů

<https://cw.felk.cvut.cz/upload>

- Přednášející:

- doc. Ing. **Jan Faigl**, Ph.D.



- Garant předmětu:

- doc. Ing. **Ivan Jelínek**, CSc.



Cíle předmětu

Programování 1

Prerekvizita Programování 2 a Algoritmizace

- **Osvojit si** pohled na výpočetní prostředky jako „*počítačový vědec*“ a naučit se je efektivně používat

Computer scientist

- Formulovat problém a jeho řešení počítačovým programem
 - Získat povědomí jaké problémy lze výpočetně řešit

- **Získat zkušenost** s programováním

získání vlastní zkušenosti

- Programování v jazyku Java

cvičení a domácí úkoly

- **Osvojit si** schopnost číst, psát a porozumět malých programům


- **Získat** programovací návyky jak psát

- srozumitelné a přehledné zdrojové kódy;
 - opakovaně použitelné programy.

Zdroje a literatura

- Přednášky – slidy, poznámky a především **vlastní zápisky**
- Cvičení – získání praktických dovedností řešením domácích úkolů a dalších úloh


programovat, programovat, programovat

 Učebnice jazyka Java 5. v., *Pavel Herout* KOPP, 2010, ISBN 978-80-7232-398-2



 Introduction to Java Programming, 9th Edition, *Y. Daniel Liang* Prentice Hall, 2012
<http://www.cs.armstrong.edu/liang/intro9e>



 An Introduction to Object-Oriented Programming with Java, 5th Edition, *C. Thomas Wu*, McGraw-Hill, 2009
<http://it-ebooks.info/book/1908/a>



- On-line kurzy programování (v Java nebo jiném jazyku)
 - <http://www.algorithmy.net/article/21340/Uvod-1>
 - http://www.linuxsoft.cz/article.php?id_article=244
 - <http://www.root.cz/serialy/programovaci-jazyk-java-a-jvm>
g programování java

Cvičení

■ Ing. **Zdeněk Buk**, Ph.D.



■ Ing. **Martin Mudroch**, Ph.D.



■ Ing. **Ondřej Hrstka**



■ Ing. **Jakub Mrva**



■ Ing. **Martin Schaefer**



■ Ing. **Petr Váňa**



Zimní semestr (ZS) akademického roku 2015/2016

- Harmonogram akademického roku 2015/2016
<http://www.fel.cvut.cz/cz/education/harmonogram1516.html>
- Přednášky:
 - **Otevřená informatika (OI)**: středa, 16:15–17:45
Dejvice, místnost T2:D3-309
 - **Kybernetika a robotika (KyR)**: čtvrtek, 15:15–17:00
Karlovo náměstí, místnost KN:E-107
 - Softwarové technologie a management (STM)
7B36ALG - Algoritmizace: čtvrtek, 15:15–17:00
<https://cw.fel.cvut.cz/wiki/courses/a0b36pr1/lectures/harmonogram>
- 14 výukových týdnů
 - 28.10.2015 (středa) a 17.11.2015 (úterý) státní svátek
 - 23.12.2015 (středa) zimní prázdniny
 - 24.12.2015 (čtvrtek) a 25.12.2015 (pátek) státní svátek
 - 11.1.2016 (pondělí) střeční rozvrh

Počítačové laboratoře

- Síťové domovské adresáře NFS v4
Přenos a synchronizace souborů
 - USB media
 - Síťové přenosy (ftp, ssh, unison, rsync)
 - Owncloud – <https://owncloud.cesnet.cz>
služby sdružení CESNET
- Programování ve vývojovém prostředí NetBeans IDE 8.0 a Java verze 8.
<https://netbeans.org>
 - Použití libovolného jiného prostředí je možné, např. IntelliJ IDEA, Eclipse,
<https://download.cvut.cz>
 - či kombinace maven nebo starší ant s textovým editorem, například vim.
- Odevzdávání domácích úkolů
 - Upload System – <https://cw.felk.cvut.cz/upload>

Domácí úkoly a další úlohy

- Samostatná práce s cílem osvojit si praktické zkušenosti
- Odevzdání domácích úkolů prostřednictvím Upload System
 - <https://cw.felk.cvut.cz/upload>
 - Nahrání (upload) archivů s nezbytnými zdrojovými soubory
 - Ověření správnosti implementace automatickými testy

Detekce plagiátů
- Úkoly jsou jednoduché a navrhované tak, aby byly stihnutelné
- Klíčem k úspěšnému dokončení předmětu je samostatná práce a osvojení si technik a znalostí

průběžná práce a řešení úkolů
- Pokud něčemu nerozumíte, ptejte se cvičících

Pokud možno hned a neodkládejte na později
- Pokud vám přijde úkolů málo, ptejte se po dalších úlohách na procvičování.

Kontrola znalostí testy

- 4 testy na cvičení každý se ziskem maximálně 5 bodů
- Implementační testy na počítačích v učebně

Příprava na zkoušku
- 1. (týden 3) - **TEST: implementační test (Karel)**

implementace programu (~ 90 minut)
- 2. (týden 7) - **TEST: písemný test (procedurální programování)**

„programování“ na papír (~ 20 minut)
- 3. (týden 9) - **TEST: implementační test (procedurální prog.)**

implementace programu (~ 60 minut)
- 4. (týden 13) - **TEST: implementační test (objekty)**

implementace programu (~ 60 minut)

Uvedené časy jsou orientační a spíše odpovídají očekávané náročnosti testu.

Přehled domácích úkolů

- 10 domácích úkolů po 5 bodech (+1 testovací)
 0. (týden 1) - (Lab00) První program s robotem Karel

Testovací úkol za 0 bodů
 1. (týden 1) - (Lab01) Robot Karel - cykly a vlastní příkaz
 2. (týden 2) - (Lab02) Robot Karel - hledání objektu v bludišti
 3. (týden 4) - (Lab04) Řídící struktury - jednoduchá kalkulačka

Kontrola stylu
 4. (týden 5) - (Lab05) Zpracování vstupu a výstupu programu
 5. (týden 6) - (Lab06) Výpočet statistik
 6. (týden 7) - (Lab07) Automatické zpracování souboru hodnot
 7. (týden 8) - (Lab08) Implementace kruhové fronty
 8. (týden 9) - (Lab09) Řešení problému rekurzí
 9. (týden 11) - (Lab11) Zatřídování spojových seznamů
 10. (týden 12) - (Lab12) Implementace prioritní fronty haldou

Kontrola stylu
- Podmínkou zápočtu je úspěšné odevzdání všech domácích úkolů
- Bodová ztráta za pozdní odevzdání úkolu (týden)

Hodnocení předmětu

Zdroj bodů	Maximum bodů	Přípustné minimum bodů
Domácí úkoly (10×5 bodů)	50	30
Testy na cvičení (4×5 bodů)	20	10
Implementační zkouška	20	10
Ústní zkouška	20	-10

- Minimální počet bodů pro zápočet **40**
- Při rozhodnutí k ústní zkoušce je odečteno 10 bodů
- Pro úspěšné absolvování předmětu je nutné získat **zápočet** a vykonat **zkoušku**
- Získání **zápočtu** je podmíněno odevzdáním všech domácích úkolů a úspěšně složenými testy

Podmínka nutná nikoliv však postačující

Klasifikace předmětu

Klasifikace	Bodové rozmezí	Hodnocení	Slovní hodnocení
A	90–100	1	výborně
B	80–89	1,5	velmi dobře
C	70–79	2	dobře
D	60–69	2,5	uspokojivě
E	50–59	3	dostatečně
F	<50	4	nedostatečně

- Minimální přípustné body:

30 (úkoly) + 10 (testy) + 10 (zkouška) = 50 bodů

Případ A: „Na cvičení se nudím.“

- Zápis předmětu **A4B36ACM - Seminář ACM z algoritmizace**

- RNDr. Marko Genyk-Berezovskyj (berezovs@fel.cvut.cz)

- Podmínky a pravidla:

- Projít vstupním testem A4B36ACM
- Odevzdat všechny úkoly z PR1 do 10. týdne
- Napsat všechny testy z PR1
- Jít na zkoušku z PR1

- Benefity:

- Lze získat +4 kredity z ACM předmětu
- **Rozšíření znalostí pokročilého programování**
- Kromě testů není nutné chodit na cvičení z PR1
- Úlohy lze odevzdat později (do 10. týdne) a všechny najednou

Dá se zvládnout za odpoledne / den

Obtížnost předmětu

- V předmětu **nepředpokládáme** znalosti z programování

- **Předpokládáme** však základní dovednosti ovládání počítače

Principiální, nikoliv pouze memorované

- Přístup na středních školách je různý a tím také úroveň vstupních znalostí a dovedností

Kurz je koncipovaný tak, aby zapsaní studenti kurz programování zvládli a získali odpovídající znalosti a dovednosti. S ohledem na různorodost je to však pro někoho rychlejší a pro někoho pomalejší. V průběhu semestru můžeme přednášky přizpůsobit konkrétním potřebám – zpětná vazba je důležitá, nebojte se ozvat.

- Mohou proto nastat dva extrémní případy:

A – Pokud je PR1 příliš snadné

Alternativní průchod

B – Počáteční nápor znalostí je značný

Pomoc při startu a přechodu na jiný styl výuky na VŠ.

Případ B: „Na cvičení nestíhám!“

- **Letní prázdninové soustředění –**

<http://cs.fel.cvut.cz/en/news/detail/1148>

- Úlohy <https://cw.fel.cvut.cz/wiki/courses/a0b36pr1/tutorials/bootcamp/start>

- Konzultace u cvičícího

- Konzultace u přednášejícího

- Tutoriály na stránkách předmětu

<https://cw.fel.cvut.cz/wiki/courses/a0b36pr1>

- Prvních pět týdnů lze pracovat na řešení problému

- v T2:H1-131 (pátek od 11:00-12:30)
- s možností konzultovat řešené problémy s dozorem
- **Není** to však náhrada cvičení ani seminární cvičení zaměřené na řešení domácích úloh.

Spíše je to prostor pracovat na problémech a při pochybnostech interaktivně požádat o radu.

Část II

Programování a výpočetní prostředky

Test pochopení principu přiřazení

- Zápis programu pro přiřazení hodnot do proměnných a a b a následné přiřazení proměnné b do a .

Přiřazení hodnoty proměnné

```
1 int a = 10;
2 int b = 20;
3
4 a = b;
```

- Jaké jsou hodnoty proměnných a a b ?

a. $a = 20$ $b = 0$ f. $a = 30$ $b = 0$ b. $a = 20$ $b = 20$ g. $a = 10$ $b = 30$ c. $a = 0$ $b = 10$ h. $a = 0$ $b = 30$ d. $a = 10$ $b = 10$ i. $a = 10$ $b = 20$ e. $a = 30$ $b = 20$ j. $a = 20$ $b = 10$

Základní koncept programování

„Separating Programming Sheep from Non-Programming Goats”

[http://blog.codinghorror.com/](http://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats)

[separating-programming-sheep-from-non-programming-goats](http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf)

<http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf>

- Efektivní metody výuky programování se hledají již od dob prvních počítačů
tj. přes více než 50 let
- Přesto se zdá, že je každý základní kurz programování obtížný a 30% až 60% studentů jej na poprvé nezvládne
V PR1 je průchodnost výrazně vyšší.
- Základní koncept je pochopení principu přiřazení hodnoty proměnné

Skupiny počítačových uživatelů

„Uživatel”

- Spouštěč programů
- Zadává vstup
- Čeká na výstup
- Čte výstup

Píše, kliká

- **Relativně omezená množina vstupů**

Pouze to co je dovoleno

„Programátor”

- Spouští programy
- Dává počítači příkazy
- Vytváří nové programy
- Kombinuje příkazy

Řadí je do posloupnosti

- **Rozmanitější možnosti použití**

Omezen pouze limity počítače

Způsob reprezentace znalostí

- Z hlediska výpočtu můžeme rozlišit dva základní typy znalostí:

Způsob popisu problému

Deklarativní

- Tvrzení popisující stav
- Axiomatické
- Umožňuje jednoduše ověřovat (testovat) pravdivost tvrzení
- Neposkytuje návod jak vypočítat hodnotu

Příklad:

$$\sqrt{x} = y, y^2 = x, x \geq 0, y \geq 0$$

Imperativní

- Popis jak něco vypočítat
- Posloupnost výpočtu
- Test jak ovlivnit průběh výpočtu

Příklad:

1. If $y^2 \approx x$

2. Then

return y

3. Else

$$y \leftarrow \frac{y+x}{2}$$

Go to Step 1

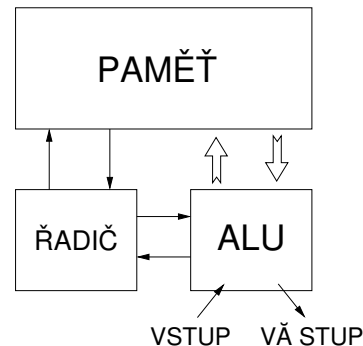
Von Neumannova architektura

- ALU - Aritmeticko logická jednotka (Arithmetic Logic Unit)

Základní matematické a logické instrukce

- PC - Čítač instrukcí (Program Counter)

„Ukazuje“ na místo v paměti s instrukcemi pro vykonání



V drtivě většině případů je program posloupnost instrukcí zpracovávající jednu nebo dvě hodnoty (uložené v nějakém paměťovém místě) jako vstup a generování nějaké výstupní hodnoty, kterou ukládá někam do paměti nebo modifikuje hodnotu PC (podmíněně řízení běhu programu).

Výpočetní prostředky (počítače)

- Jednoúčelové přístroje s předepsaným chováním
 - program / posloupnost kroků (instrukcí) je vestavěná a neměnná
Kalkulačka, pračka, první telefony
- Počítač s uloženým programem v paměti
 - Posloupnost instrukcí čtena z paměti
 - Flexibilita ve tvorbě posloupnosti
Program lze libovolně měnit
 - Architektura počítače se společnou pamětí pro data a program
 - Von Neumannova architektura počítače
John Louis von Neumann (1903–1957)

Program je „recept“

- Program je posloupnost kroků (výpočtů) popisující průběh výpočtu pro řešení problému
(je to „recept“ na řešení problému)
- Pro zápis receptu potřebujeme **jazyk**
Způsob zápisu programu
- Jazyk definuje základní sadu primitiv (operací/příkazů), které můžeme použít pro zápis receptu
- S konečnou množinou primitiv dobrý programátor naprogramuje „cokoliv“.
Co může být vyjádřeno.
- Touring Machine – obecný model počítačícího stroje
Alan Turing, 1936
- V předmětu A0B36PR1 používáme programovací jazyk **Java**
Programování není o znalosti konkrétního programovacího jazyka, je to o způsobu uvažování a řešení problému.

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 typy 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

Správnost programu

- Syntakticky i staticky sémanticky správný program neznamena, že dělá to co od něj požadujeme
- Správnost a smysluplnost programu je dána očekávaným chováním při řešení požadovaného problému
- V zásadě při spuštění programu mohou nastat tyto události:
 - Program havaruje a dojde k chybovému výpisu

Mrzuté, ale výpis (report) je dobrý start řešení chyby (bug)
 - Program běží, ale nezastaví se a počítá v nekonečné smyčce.

Zpravidla velmi obtížné detekovat a program ukončujeme po nějaké době.
 - Program včas dává odpověď

Je však dobré vědět, že odpověď je korektní.

Správnost programu je plně v režii programátora, proto je důležité pro snadnější ověření správnosti, ladění a hledání chyby používat **dobry programovací styl.**

Definice programovacího jazyka

- **Syntax** – definice povolených výrazů a konstrukcí programu

Plná kontrola a podpora vývojových prostředí

- Příklad popisu výrazu gramatikou v **Backus-Naurově formě**.

```
<exp> ::= <exp> + <exp> | <exp> * <exp> | <exp> |
<number> <number> ::= <number> <digit> | <digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

- **Statická sémantika** – definuje jak jsou konstrukty používány

Částečná kontrola a podpora prostředí

- Příklad axiomatické specifikace: $\{P\} S \{Q\}$, P - precondition, Q -postcondition, S - konstrukce jazyka.

- **Plná sémantika** – co program znamená a dělá, jeho smysluplnost

Kontrola a ověření správnosti je kompletně v režii programátora.

Program a jeho zápis

- Program – popis činnosti prováděné počítačem.
- Programovací jazyk – notační systém pro zápis programu.

Program zpravidla zapisujeme ve zdrojových (textových) souborech.

- Čitelnost programu:

- strojová – efektivnost kódu
 - lidská – srozumitelnost, udržitelnost, **kódovací konvence**,

Vývojová prostředí (editor, debugger, nástroje pro správu verzí, analýza, testování → softwarové inženýrství.

- Abstrakce

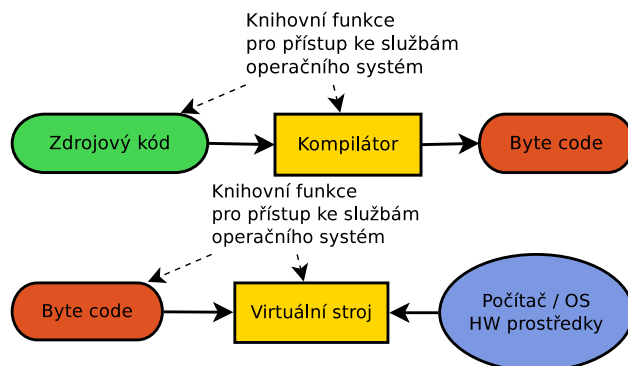
- Datová – základní typy, struktury, modulární.
 - Řídící – základní, strukturální, modulární.

Část III

Úvod do programovacího jazyku Java

Zdrojové kódy, překlad a spuštění Java programu

- Zdrojové kódy jsou zapisovány v textových souborech s koncovkou **.java**
- Zdrojové soubory jsou překládány překladačem (javac) do binárního kódu („byte code“) uložených v souborech s koncovkou **.class**
- Spuštění programu je realizováno virtuálním strojem, který poskytuje abstrakci nad operačním systémem počítače



Java

- Obecný, vyšší, imperativní (procedurální) a objektově orientovaný jazyk
- Překládaný jazyk zaměřený na přenositelnost (portabilitu) zdrojových kódů i přeložených binárních souborů
- Historie:
 - 1991 – nejdříve jako jazyk Oak
 - 1995 – Java JDK 1 (první veřejná verze)
 - 1998 – Java 2 (ver. 1.2)
 - 2002 – Java 2 (ver. 1.4) a J2EE
 - 2004 – Java 2 (ver. 1.5) – J2SE5.0
 - 2011 – Java 7 (vydává Oracle po akvizici Sun Microsystems)
 - 2014 – Java 8 (18. března, 2014)
- Součástí základního vývojového prostředí je bohatý soubor knihovnicích funkcí.

Java je relativně jednoduchý jazyk (v základní verzi) a jeho efektivní používání je spíše o znalosti knihovnicích funkcí.

Java prostředí – JDK, JRE, JVM

- **JDK** (Java Development Kit) – základní vývojové prostředí, knihovny funkcí, překladač zdrojových souborů javac. Jeho součástí je i JRE.
- **JRE** (Java Runtime Environment) – základ prostředí Java pro spuštění programů, obsahuje virtuální stroj java.
- **JVM** (Java Virtual Machine) – virtuální stroj pro spuštění Java programů (java).
- **JAR** (Java ARchive) – archiv Java souborů, typicky množiny zkompileovaných .class (tříd) doplněných textovým popisem (Manifest), kterou třídu spustit. Slouží pro snadnější spuštění programů o více souborech.

V podstatě ZIP archiv

Příklad

Výpočet druhé odmocniny

```

1 double x = 13.0;
2 double y = 1.0;
3 int i = 1;
4
5 while(Math.abs(y*y - x) > 1e-3) {
6     System.out.println("Step " + i + " y = " + y);
7     y = (y+(x/y))/2;
8     i += 1;
9 }
10 System.out.println("sqrt(" + x + ") found in " + i + "
    steps as " + y);

```

lec01/Sqrt.java

Kompilace a spuštění programu

```

javac sqrt.java
java sqrt

```

Primitivní typy

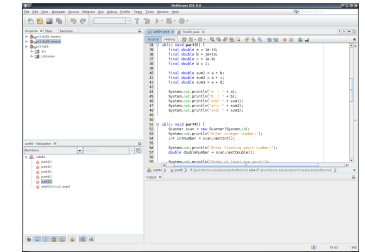
- Vyhrazují v paměti místo pro uložení číselné hodnoty
Velikost alokovaného prostoru odpovídá možnému rozsahu a přesnosti reprezentace čísla
- Celočíselné typy reprezentují čísla v definovaném rozsahu
- Neceločíselné typy reprezentují reálná, racionální a iracionální čísla.
Aproximace do pevného počtu bitů
 - Mantisa + exponent

s	e	e	...	M	M	...	M
---	---	---	-----	---	---	-----	---

 kde s - znaménkový bit, M - mantisa, e - exponent,
 $s \times M \times B^{e-E}$, B - báze, E - konstanta.
 - double - 64 bitů podle IEEE 754, $\pm 4.9E-324$ - $\pm 1.7E+308$.

Integrovaná vývojová prostředí (IDE)

- Nadstavba základních příkazů javac a java
- Integrují (mimo jiné) systém pro řízení překladu
Např. ant nebo maven
- Zvýrazňují syntax, doplňují jména a provádějí základní kontrolu kódu
- Mezi nejznámější patří Netbeans, Eclipse a IntelliJ IDEA
<https://download.cvut.cz>
- Na cvičení je používáno prostředí Netbeans
šťábní kultura
- Adresářová struktura projektu



Možnosti zvýšení přesnosti

- Reprezentace racionálních čísel - podíl dvou celočíselných hodnot, např. *Homogenní souřadnice*.
- „Libovolná přesnost“ - speciální knihovny, např. gmp až do výše volné paměti.
souřadnice x,y - 7511164176768 346868669952 3739567104 ~ 2008.57, 92.76

Základní typy v Javě

Celočíselné typy

- **byte** – 8 bitů (1 byte), -128 až 127
- **short** – 16 bitů (2 byte), -32 768 až 32 767
- **int** – 32 bitů (4 bytes), -2^{31} až $2^{31}-1$
základní celočíselný typ
- **long** – 64-bitů (8 bajtů), -2^{63} až $2^{63}-1$

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

Neceločíselné typy

- **float** – 32-bit IEEE 754
- **double** – 64-bit IEEE 754

Logický a znakový typ

- **boolean** – true / false
- **char** – jeden 16-bit Unicode znak

Přesnost výpočtu 1/2

- Ztráta přesnosti při aritmetických operacích.

Příklad sčítání dvou čísel

```

1 public static void main(String[] args) {
2     double a = 1E+10;
3     double b = 1E-10;
4     System.out.println("a : " + a);
5     System.out.println("b : " + b);
6     System.out.println("a+b: " + (a+b));
7 }
8
9 javac Sum.javac
10 java Sum
11 a : 1.0E10
12 b : 1.0E-10
13 a+b: 1.0E10

```

lec01/Sum.java

Zápis číselné hodnoty v programu (literál)

- Základní zápis je pro typ **int** a **double**

```

int decI = 173;
int hexI = 0xad;
int binI = 0b10101101;

double d1 = 105.67;
double d2 = 1.0567e2;

```

lec01/Literals1.java

- Zápis hodnoty typu **long** a **float** je nutné specifikovat

```

long l1 = 171; //specifikujeme znakem l
long l2 = 13313514L; //nebo znakem L

float f1 = 105.67f; //nutne uvest f nebo F
float f2 = 1.0567e2f; //jinak pri kompilaci chyba

```

lec01/Literals2.java

Přesnost výpočtu 2/2

Příklad dělení dvou čísel

```

1 public static void main(String[] args) {
2     final int number = 100;
3     double dV = 0.0;
4     float fV = (float)0.0;
5     for (int i = 0; i < number; i++) {
6         dV += 1.0 / 10.0;
7         fV += 1.0 / 10.0;
8     }
9     System.out.println("double value: " + dV);
10    System.out.println("float value: " + fV);
11 }
12
13 javac Division.java
14 java Division
15 double value: 9.999999999999998
16 float value: 10.000002

```

lec01/Division.java

Přesnost výpočtu - strojová přesnost

- Strojová přesnost ϵ_m - nejmenší desetinné číslo, které přičtením k 1.0 dává výsledek různý od 1, pro $|v| < \epsilon_m$, platí

$$v + 1.0 \neq 1.0.$$

Symbol `==` odpovídá porovnání dvou hodnot v Javě (test na ekvivalenci).

- Zaokrouhlovací chyba - nejméně ϵ_m .
- Přesnost výpočtu - aditivní chyba roste s počtem operací v řádu $\sqrt{N} \cdot \epsilon_m$.
 - Často se však kumuluje preferabilně v jedno směru v řádu $N \cdot \epsilon_m$.

Podmíněnost numerických úloh

- Podmíněnost úlohy $C_p = \frac{\text{relativní chyba výstupních údajů}}{\text{relativní chyba vstupních údajů}}$
- Dobře podmíněná úloha $C_p \approx 1$.
- Výpočet je dobře podmíněný, je-li málo citlivý na poruchy ve vstupních datech.
- Numericky stabilní výpočet - vliv zaokrouhlovacích chyb na výsledek je malý.
- Výpočet je stabilní, je-li dobře podmíněný a numericky stabilní.

Zdroje a typy chyby

- Chyby matematického modelu - matematická aproximace fyzikální situace.
- Chyby vstupních dat.
- Chyby numerické metody.
- Chyby zaokrouhlovací.
- Absolutní chyba aproximace $E(x) = \hat{x} - x$, \hat{x} přesná hodnota, x aproximace.
- Relativní chyba $RE(x) = \frac{\hat{x} - x}{x}$.

Příklady chyb

- Ariane 5 - 4.6.1996
40 sekund po startu explodovala. Datová konverze z 64-bitového desetinné reprezentace na 16-ti bitový znaménkový integer.
http://www.esa.int/esaCP/Pr_33_1996_p_EN.html
- Systém Patriot - 25.2.1991
Systémový čas v desetinách sekundy, převod na sekundy realizován dělením 10, registry pouze 24 bitů.
<http://www.ima.umn.edu/~arnold/disasters/patriot.html>
<http://www5.informatik.tu-muenchen.de/~huckle/bugse.html>

Shrnutí přednášky

Diskutovaná témata

- Informace o předmětu
- Programování, reprezentace znalostí, program a programovací jazyk
- Úvod do jazyku Java
- Základní datové typy a přesnost výpočtu

- **Příště: Výrazy, operátory a řídicí konstrukce**