Paměť počítače

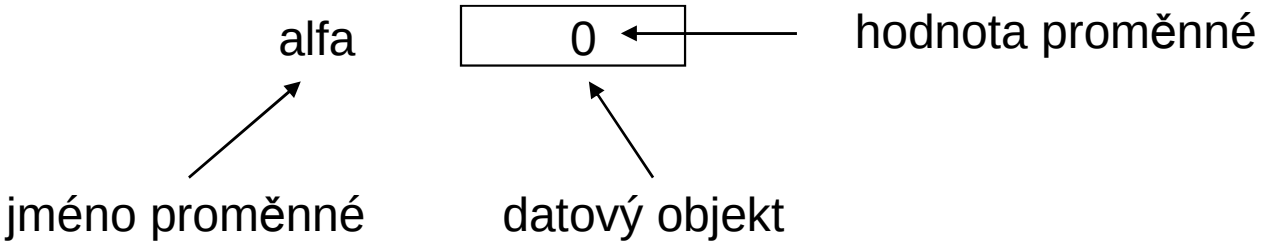
- Výpočetní proces je posloupnost akcí nad daty uloženými v paměti počítače
- Data jsou v paměti reprezentována posloupnostmi bitů (bit = 0 nebo 1)
- Připomeňme:
  - paměť je tvořena řadou 8-mi bitových paměťových míst nazývaných bajt (z angl. byte, česky též slabika)
  - rozlišujeme vnitřní (operační) paměť a vnější paměť (např. disk)
  - každé paměťové místo vnitřní paměti má svou adresu (nezáporné celé číslo), která slouží pro jeho identifikaci
  - kapacita paměti se udává v MB (1 MB = 2^{20}B) nebo GB (1 MB = 2^{30}B)
- Instrukce strojového jazyka předepisují aritmetické, logické a jiné operace s posloupnostmi bitů bez ohledu na to, jaká data posloupnost bitů reprezentuje

Datové typy

- Při návrhu algoritmů a psaní programů ve vyšších programovacích jazycích abstrahujeme od binární podoby paměti počítače
- S daty pracujeme jako s hodnotami různých datových typů, které jsou uloženy v datových objektech
- Datový typ (zkráceně jen typ) specifikuje:
  - množinu hodnot
  - množinu operací, které lze s hodnotami daného typu provádět
- Příklad typu: celočíselný typ *int* v jazyku Java:
  - množinou hodnot jsou celá čísla z intervalu -2147483648 .. 2147483647
  - množinu operací tvoří
    - aritmetické operace +, -, *, /, jejichž výsledkem je hodnota typu *int*
    - relační operace ==, !=, >, >=, <, <=, jejichž výsledkem je hodnota typu *boolean*
    - a další
- Typ *int* je jednoduchý typ, jehož hodnoty jsou atomické (z hlediska operací dále nedělitelné)

Proměnné a přiřazení

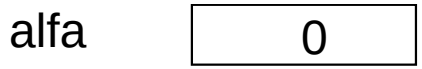
- Proměnná je datový objekt, který je označen jménem a je v něm uložena hodnota nějakého typu, která se může měnit



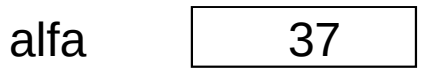
- V jazyku Java zavedeme výše uvedenou proměnnou deklarací

```
int alfa = 0;
```

- Hodnotu proměnné lze změnit přiřazovacím příkazem




```
alfa = 37;
```



Jednoduché datové typy

- V dalších příkladech budeme používat tyto jednoduché (primitivní) typy jazyka Java:
  - *byte* celá čísla z intervalu –128 .. 127
  - *int* celá čísla z intervalu -2147483648 .. 2147483647
  - *double* aproximace reálných čísel v absolutní hodnotě od 4.9406545841246544E-324 do 1.79769313486231579E+308
  - *boolean* *false*, *true*
  - *char* znaky
- Reprezentace hodnot jednoduchých typů v paměti počítače:
  - *byte* 8 bitů v doplňkovém kódu
  - *int* 32 bitů v doplňkovém kódu
  - *double* 64 bitů v pohyblivé řádové čárce
  - *boolean* 8 bitů, 0 nebo 1
  - *char* 16 bitů v kódu Unicode
- Podrobněji na cvičeních

Deklarace proměnných

- Proměnné se zavádějí deklaracemi
- Příklady deklarací proměnných:

```
int i;          // deklarace proměnné i typu int
double x;       // deklarace proměnné x typu double
```

- Proměnná deklarovaná uvnitř funkce (lokální proměnná) nemá deklarací definovanou hodnotu
- Použití proměnné s nedefinovanou hodnotou v jazyku Java je chyba při překladu
- Příklad:

```
int x, y;
x = y + 2;      // chyba při překladu
```

- Deklaraci proměnné lze doplnit o inicializaci proměnné:

```
int x = 10;     // deklarovaná proměnná má hodnotu 10
```

- Deklarací lze zavést několik proměnných stejného typu:

```
int x, z;
```

Literály a pojmenované konstanty

- Přímý zápis hodnoty v programu se nazývá literál
  - *int* 34 45
  - *double* 25.3 1.2E-3
  - *boolean* *false* *true*
  - *char* ‘a’ ‘1’ ‘+’
- Dalším často potřebným literálem je literál typu *String* (není to jednoduchý typ):
  “abcd” “Nazdar”
- Kromě literálů lze v jazyku Java používat pojmenované konstanty, které se deklarují podobně jako inicializované proměnné, v deklaraci je navíc klíčové slovo *final*
- Příklad deklarace konstant:

```
final int MAX = 100;
final String NAZEV_PREDMETU = “Algoritmizace”;
```

- Konvence: Jména konstant píšeme velkými písmeny
- Konstantě nelze změnit hodnotu přiřazovacím příkazem

```
MAX = 20;     // chyba při překladu
```

Přiřazovací příkaz

- Slouží pro přiřazení hodnoty proměnné
- Tvar přiřazovacího příkazu:

  proměnná = výraz;

- Příklad:

  `x = y + z;`

  proměnné *x* se přiřadí součet hodnot proměnných *y* a *z*

  `x = x + 1;`

  hodnota proměnné *x* se zvětší o 1

- Proměnné v jazyku Java lze přiřadit pouze hodnotu jejího typu
- Příklady nedovolených přiřazovacích příkazů:

```
boolean b; int i; double d;
b = 1;
i = 1.4;
d = true;
```

Typové konverze

- Typová konverze je operace, která hodnotu nějakého typu převede na hodnotu jiného typu
- Typová konverze může být implicitní (vyvolá se automaticky) nebo explicitní (v programu je třeba ji explicitně předepsat)
- Konverze typu *int* na *double* je v jazyku Java implicitní:
  - kde se očekává hodnota typu *double*, může být uvedena hodnota typu *int*, která se automaticky převede na hodnotu typu *double*
- Příklad:

```
double x; int i = 1;
x = i;  // hodnota 1 typu int se automaticky převede na
        // hodnotu 1.0 typu double
```

- Převod hodnoty typu *double* na *int* (odseknutím necelé části) je třeba explicitně předepsat
- Příklad:

```
double x = 1.2; int i;
i = (int)x;   // hodnota 1.2 typu double se odseknutím
              // necelé části převede na hodnotu 1 typu
              // int
```

Příkazy výstupu

- Pro výpis dat na obrazovku se v Javě nejčastěji používá příkaz
  `System.out.println(`*`parametr`*`);`
- Data daná parametrem se vypíší na aktuální řádek a přejde se na další řádek
- Pro výpis dat na aktuální řádek bez přechodu na další řádek lze použít příkaz
  `System.out.print(`*`parametr`*`);`
- Parametrem musí být výraz typu *String* (řetězec)
- Příklad výpisu textu:
  `System.out.println("Nazdar");`
- Pro každý jednoduchý typ existuje implicitní konverze na typ *String*, tzn. parametrem příkazu výstupu může být proměnná (výraz) jednoduchého typu
- Příklad výpisu hodnoty proměnné *n* typu *int*:
  `System.out.println(n);`
- Řetězce lze spojovat pomocí operátoru +; je tedy dovolen např. tento příkaz:
  `System.out.println("hodnota proměnné n = " + n);`

Druhý program

```
package alg2;

public class DruhyProgram {
  public static void main(String[] args) {
    int x = 10, y;
    y = x + 20;
    System.out.println("hodnota proměnné x je "+x);
    System.out.println("hodnota proměnné y je "+y);
  }
}
```

- Program po překladu a spuštění vypíše:

```
hodnota proměnné x je 10
hodnota proměnné y je 30
```

Poznámka k příkazu výstupu

- Co vypíší následující příkazy:

příkazy	výstup
`int x = 1, y = 2;` `System.out.println(x+y);`	`3`
`int x = 1, y = 2;` `System.out.println("součet je "+(x+y));`	`součet je 3`
`int x = 1, y = 2;` `System.out.println("součet je "+x+y);`	`součet je 12`

- Příčinou „podivného“ výstupu posledního programu je, že operátor + (a většina dalších) je asociativní zleva, tzn. výraz **`"součet je " + x + y`**
  se vyhodnotí takto: **`("součet je " + x) + y`**

Vstup a výstup pomocí třídy Sys

- V dalších příkladech budeme potřebovat příkazy pro vstup číselných dat zadaných na klávesnici
- Vstup dat (např. čísel) pomocí předdefinovaných tříd jazyka Java je složitější, než v jiných jazycích
- Použijeme proto vlastní třídu *Sys*, která vstup i výstup zjednodušuje:

```
package alg2;

import sugar.Sys;

public class TretiProgram {
  public static void main(String[] args) {
    int x, y, z;
    Sys.pln("zadejte dvě celá čísla");
    x = Sys.readInt();
    y = Sys.readInt();
    z = x + y;
    Sys.pln("součet čísel je "+z);
  }
}
```

Vstup a výstup pomocí třídy Sys

- Využití služeb třídy *Sys* je třeba deklarovat příkazem
  `import sugar.Sys;`
- Třída *Sys* poskytuje tyto služby:
  `Sys.readInt()`
  - přečte celé číslo z řádku zadaného klávesnicí (řádek je zakončen klávesou *Enter,* číslo je zakončeno mezerou nebo *Enter*) a vrátí je jako funkční hodnotu typu *int*

  `Sys.readDouble()`
  - přečte číslo z řádku zadaného klávesnicí a vráti je jako funkční hodnotu typu *double*

  `Sys.p(x)`
  - na obrazovku vypíše hodnotu danou parametrem *x*, který může být libovolného jednoduchého typu nebo typu *String*

  `Sys.pln(x)`
  - na obrazovku vypíše hodnotu danou parametrem *x*, který může být libovolného jednoduchého typu nebo typu *String*, a přejde na nový řádek

  a další ...

Výrazy

- Výraz předepisuje výpočet hodnoty určitého typu
- Výraz může obsahovat:
  - proměnné
  - konstanty
  - volání funkcí
  - binární operátory
  - unární operátory
  - závorky
- Pořadí operací předepsaných výrazem je dáno:
  - prioritou operátorů
  - asociativitou operátorů
- Příklad:

výraz	pořadí operací	zdůvodnění
`x + y * z`	`x + (y * z)`	* má vyšší prioritu než
`x + y + z`	`(x + y) + z`	+ je asociativní zleva

Aritmetické operátory

- Pro operandy typu *int* a *double* budeme požívat tyto aritmetické operátory (seřazeno sestupně podle priority):

  unární – (změna znaménka)

  binární *, / a % (násobení, dělení a zbytek po dělení)

  binární + a – (sčítání a odčítání)
- Jsou-li oba operandy stejného typu, výsledek aritmetické operace je téhož typu
- Jsou-li operandy různého typu, operand typu *int* se implicitní konverzí převede na hodnotu typu *double* a výsledkem operace je hodnota typu *double*
- Výsledkem dělení operandů typu *int* je celá část podílu
  např. 7 / 3 je 2 -7 / 3 je -2
- Pro zbytek po dělení platí: $x \% y = x - (x / y) * y$
  např: 7 % 3 je 1 -7 % 3 je -1 7 % -3 je 1 -7 % -3 je -1

Relační operátory

- Hodnoty všech jednoduchých typů jsou uspořádané a lze je porovnávat relačními operátory
- Budeme používat tyto relační operátory (priorita je menší než priorita aritmetických operátorů):

  >, <, >=, <= (větší než, menší než, větší nebo rovno, menší nebo rovno)

  ==, != (rovná se, nerovná se)
- Výsledek relační operace je typu *boolean* (*true*, když relace označená operátorem platí, *false* v opačném případě)
- Jestliže při porovnávání číselných hodnot jsou operandy různého typu, operand typu *int* se implicitní konverzí převede na hodnotu typu *double*
- Relační operátory mají menší prioritu, než aritmetické operátory
- Příklady relačních výrazů:

```
int i=10; double x=12.3; boolean b;
Sys.pln(i==10);     // vypíše true
Sys.pln(i+1==10);   // vypíše false
b = i>x;            // proměnné b se přiřadí false
```

Logické operátory

- Logické operátory jsou definovány pro hodnoty typu *boolean*
  - unární ! (negace)
  - binární && (konjunkce, logický součin)
  - binární || (disjunkce, logický součet)

x	y	!x	x && y	x || y
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

- Negace má stejnou prioritu, jako změna znaménka, logický součin má nižší prioritu než relační operátory
- Operace && a || se vyhodnocují zkráceným způsobem, tj. druhý operand se nevyhodnocuje, jestliže lze výsledek určit již z prvního operandu
- Příklady:

```
int n = 10; boolean b1 = false, b2 = true;
Sys.pln(1<=n && n<=20);  // vypíše se true
Sys.pln(b1 || !b2);      // vypíše se false
```

Matematické funkce

- Při číselných výpočtech často potřebujeme matematické funkce jako *abs, sin, cos, sqrt* (druhá odmocnina), *log* (přirozený logaritmus) atd.
- Tyto funkce poskytuje knihovní třída *Math*
- Příklad:

```
package alg2;
import sugar.Sys;

public class Prepona {
  public static void main(String[] args) {
    Sys.pln("zadejte délku odvěsen prav. trojúhelníka");
    double x = Sys.readDouble();
    double y = Sys.readDouble();
    double z = Math.sqrt(x*x+y*y);
    Sys.pln("délka přepony je "+z);
  }
}
```

- Knihovní třídu *Math* není třeba (na rozdíl od třídy *sugar.Sys*) importovat příkazem *import*

Knihovní třída Math

- Některé z funkcí poskytovaných třídou *Math*

```
public static int abs(int a)
public static double abs(double a)
public static int max(int a, int b)
public static double max(double a, double b)
public static int min(int a, int b)
public static double min(double a, double b)
public static double sqrt(double a)
public static double sin(double a)
public static double random()
```

  – vrátí pseudonáhodné číslo větší nebo rovno 0.0 a menší než 1.0

- Třída poskytuje též dvě konstanty: E a PI
- Příklad:

```
public class ObvodKruhu {
  public static void main(String[] args) {
    Sys.pln("zadejte poloměr kruhu");
    double r = Sys.readDouble();
    Sys.pln("obvod kruhu je "+2*Math.PI*r);
  }
}
```