

# 2. Výstup, reálná čísla

**BI-EP1**

**Efektivní programování 1**

**ZS 2024/2025**

**Ing. Martin Kačer, Ph.D.**

© 2024 Martin Kačer

Katedra teoretické informatiky

Fakulta informačních technologií

České vysoké učení technické v Praze



# Formátování výstupu

A	B	C	D	E	F	G	H	I
A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R
J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	
S	T	U	V	W	X	Y	Z	
a	b	c	d	e	f	g	h	i
a	b	c	d	e	f	g	h	i
j	k	l	m	n	o	p	q	r
j	k	l	m	n	o	p	q	r
s	t	u	v	w	x	y	z	
s	t	u	v	w	x	y	z	

# Desetinná čísla

- Daný počet desetinných míst
- Zaokrouhlování (dle požadavků)
  - 3.1415926535
  - 3.14
  - 3.142
  - 42
  - 42.000

# Reálná čísla – C

## ■ printf

<code>printf("%f", pi);</code>	3.141593
<code>printf("%.2f", pi);</code>	3.14
<code>printf("%0.3f", pi);</code>	3.142
<code>printf("%.3f", 0.25);</code>	0.250
<code>printf("%8.3f", 0.25);</code>	0.250
<code>printf("%08.3f", 0.25);</code>	0000.250

# Reálná čísla – C++

- `cout`

- Manipulátor

- ```
cout << setprecision(N) << num;
```

|                                                      |                       |
|------------------------------------------------------|-----------------------|
| <code>&lt;&lt; std::setprecision(6)</code>           | <code>3.141593</code> |
| <code>&lt;&lt; std::setprecision(2)</code>           | <code>3.14</code>     |
| <code>&lt;&lt; std::setprecision(3)</code>           | <code>3.142</code>    |
| <code>&lt;&lt; std::setw(8)</code>                   | <code>0.250</code>    |
| <code>&lt;&lt; setw(8) &lt;&lt; setfill('0');</code> | <code>0000.250</code> |

# Reálná čísla – Java

- `java.text.*`
- `MessageFormat`, `NumberFormat`, `DecimalFormat`

|                                                              |                   |
|--------------------------------------------------------------|-------------------|
| <code>out.println(new DecimalFormat("0").format(pi));</code> |                   |
| <code>println(pi);</code>                                    | 3.141592653589793 |
| <code>DecimalFormat("0");</code>                             | 3                 |
| <code>DecimalFormat("0.000");</code>                         | 3.142             |
| <code>DecimalFormat("000.000");</code>                       | 003.142           |
| <code>DecimalFormat("###.###");</code>                       | 3.142             |
| <code>DecimalFormat("0.000");</code>                         | 42.250            |
| <code>DecimalFormat("#.###");</code>                         | 42.25             |

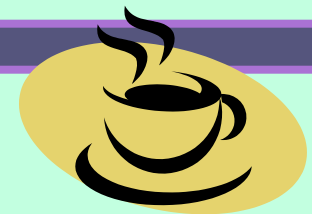
# Výstupní věta s čísly

- Doplnění hodnot do výstupní věty

C / C++

```
printf("Strazny ujde %d kroku.\n", cnt);  
printf("Prumer kruhu je %.3f cm.\n", diam);
```

```
System.out.println("Strazny ujde "  
    + cnt + " kroku.");  
  
System.out.println("Prumer kruhu je "  
    + NUMFORMAT.format(diam) + " cm.");
```



# Složitější formátování

- Požadavky na přesné rozmístění
  - mezery, ASCII-art apod.

```
  $
 $$$ $$
 $$ $ $$
 $$ $
  $$$ $$
   $ $$
 $$ $ $$
 $$$ $$
  $
```

```
#####
#   B   #
#  ###  #
#  #X   #--#
# ##### #
# A     # 00#
#####
```

$$c^2 = a^2 + b^2$$



# Složitější formátování

- Požadavky na přesné rozmístění
  - Většinou se vyplatí zapisovat do pole

```
char map[20][61];
for (r = 0; r < 20; ++r) {
    for (c = 0; c < 60; ++c)
        map[r][c] = ' ';
    map[r][60] = '\\0';
}
while(...)
    map[i][j] = '-';
    map[i+1][j] = '|';
```

**C / C++**

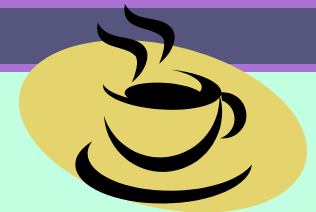
# Složitější formátování

- Požadavky na přesné rozmístění
  - Pole pak vypíšeme celé najednou

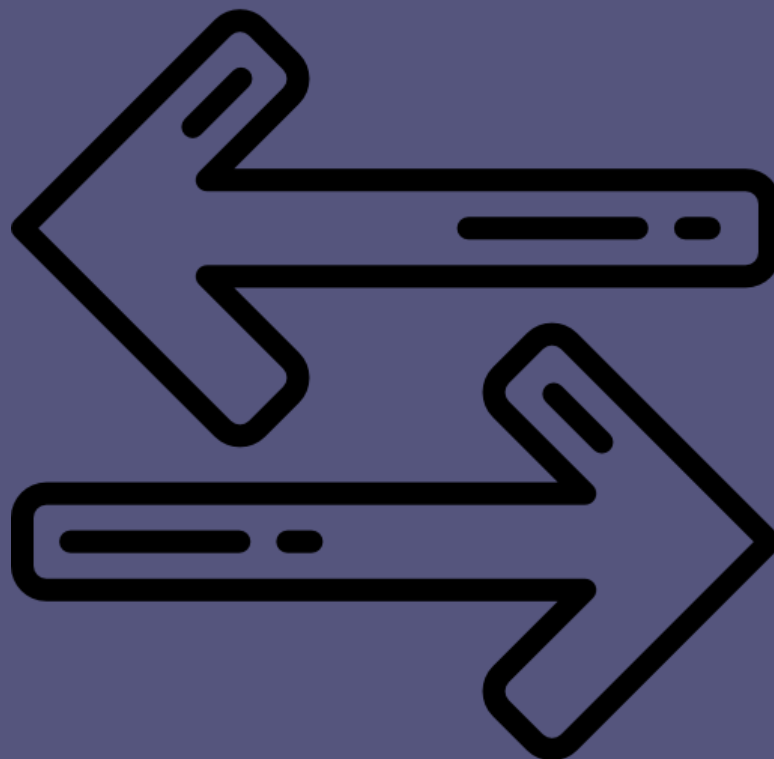
```
char map[20][60];  
.  
.  
.  
for (r = 0; r < 20; ++r)  
    printf("%s\n", map[r]);
```

C / C++

```
char ch[][] = new char[20][30];  
.  
.  
.  
for (int i = 0; i < 20; ++i)  
    System.out.println(ch[i]);
```



# Interaktivní úlohy



# Interaktivní úloha

- **Nepoužívá soubor**
- **Komunikuje s programem**
- **Střídání vstupu a výstupu**

# Interaktivní úloha – příklad

## Hádání čísel

- Úkolem je uhodnout číslo 1-1000000
- Program vypíše na výstup číslo
- Na vstupu se objeví informace, zda je větší nebo menší
- Maximálně 25 pokusů

# Interaktivní úloha – řešení

## Hádání čísel

- **Řešení:**  
**Samozřejmě binární půlení**
- **Nebojte se implementace...**

# Interaktivní úlohy – specifika

- Existence bufferů
- Výstup
  - Po každém výstupu volat flush
  - Knihovny často dělají automaticky, ale bezpečnější je na to nespoléhat

# Interaktivní úlohy – specifika

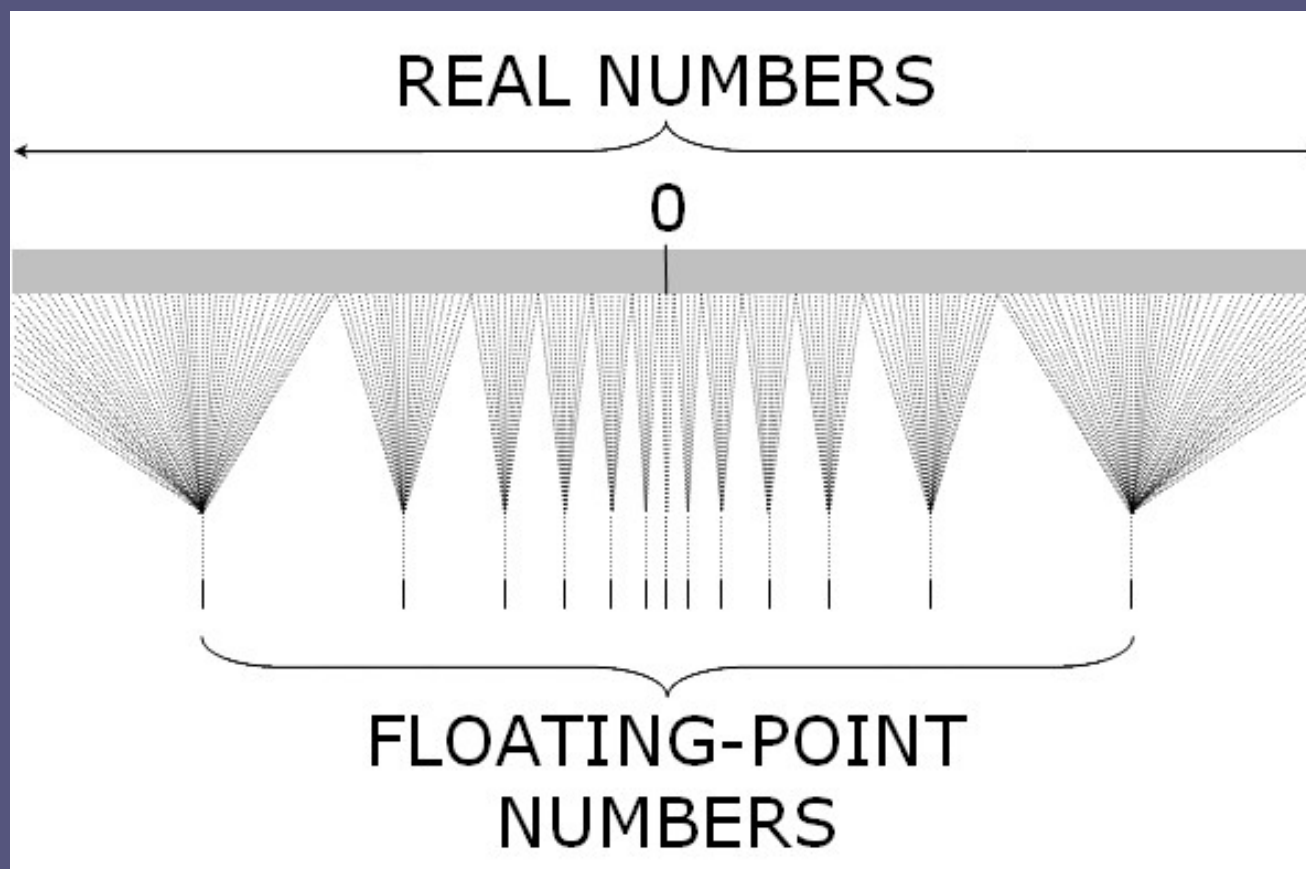
- **Vstup – vzniká postupně**
  - Nečíst (ani nezkoušet) víc, než existuje
- **OK:**
  - `cin >> str`
  - `scanf("%d", &n)`
  - `BufferedReader.readLine()`
- **Špatně:**
  - `scanf("%d\n", &n)`
  - `scanf("%d ", &n)`



# Interaktivní úloha – řešení

```
int l = 1, r = 1000000;  
while (l != r) {  
    int mid = (l + r + 1) / 2;  
    printf("%d\n", mid);  
    fflush(stdout);  
  
    char response[3];  
    scanf("%s", response);  
    if (strcmp(response, "<") == 0)  
        r = mid - 1;  
    else  
        l = mid;  
}  
printf("! %d\n", l);  
fflush(stdout);
```

# Reálná aritmetika







# Reálná čísla – nepřesnosti

- Čísla jsou nepřesná sama o sobě
  - Ne každé reálné číslo jde uložit
  - Jen racionální a ještě ne všechna
  - ... včetně těch s konečným des. rozvojem
  - 0,125 desítkově → 0,001 binárně
  - 0,2 desítkově → 0,001100110011... binárně

# Nepřesnosti – kdy vadí?

- Při výpisu se zaokrouhlí => nevadí
- **Kdy nám nepřesnosti budou vadit??**
- Při operacích se chyby „nabalují“
  - Záleží na okolnostech (příklad s odrazy)
  - Nutno na to pamatovat
- Ale to není vše ... kdy to ještě vadí?

# Nepřesnosti – kdy vadí?

- „I malá nepřesnost může způsobit velké následky ...“
  - Kdy?
- Rychle rostoucí funkce (zejm. do  $\infty$ )
  - Tangens,  $1/x$
- Nespojité funkce
  - Porovnání

# Porovnávání reálných čísel

- Co vypíše následující kód?

```
double x;  
for (x = 0.0; x != 1.0; x += 0.1) {  
    printf("%f\n", x);  
}
```

```
double x;  
for (x = 0.0; x < 1.0; x += 0.02) {  
    printf("%f\n", x);  
}
```



# Porovnávání reálných čísel

- PAMATUJTE: Desetinná čísla jsou nepřesná!!!

```
double x;  
for (x = 0.0;  
     x < 1.0;  
     x += 0.1) {  
    printf("%f\n", x);  
}
```

```
0.000000  
0.100000  
0.200000  
0.300000  
0.400000  
0.500000  
0.600000  
0.700000  
0.800000  
0.900000  
1.000000
```

# Porovnávání reálných čísel

- PAMATUJTE: Desetinná čísla jsou nepřesná!!!

```
double x;  
for (x = 0.0;  
    x <= 1.0;  
    x += 0.02) {  
    printf("%f\n", x);  
}
```

```
. . .  
0.860000  
0.880000  
0.900000  
0.920000  
0.940000  
0.960000  
0.980000
```

# Reálná čísla – příklad

- Zjistěte, zda je bod  $[X,Y]$  uvnitř kruhu o poloměru  $R$

```
if (sqrt(x*x + y*y) <= r)
```

- Odmocnina je zbytečně pomalá a nepřesná

```
if (x*x + y*y <= r*r)
```

- Lepší
- Funguje i pro celá čísla
- ... ale vlastně **JEN** pro celá čísla



# Reálná čísla – příklad

```
if (x*x + y*y <= r*r)
```



- Nevhodné kvůli nepřesnostem reálných čísel
  - Bod ležící na kružnici testem projít může a nemusí

```
static final EPS = 1E-8;
```



```
if (x*x + y*y < r*r + EPS)  
    /* včetně kružnice */
```

```
if (x*x + y*y < r*r - EPS)  
    /* vyjma kružnice */
```

# „Magické epsilon“

- Vždy používejte pro porovnávání!
  - Pokud existuje rozdíl mezi  $< a \leq$
  - Na úplnou rovnost

```
if ( abs (a-b) < EPS )
```



- Jak epsilon vybrat?
  - Dostatečně velké, aby pomohlo
  - Dostatečně malé, aby nevadilo
  - → o několik řádů menší než hodnoty

# Epsilon

- Co když neznáme rozsah hodnot?
  - Relativní epsilon

```
if (Math.abs((a-b)/b) < EPS) . . .
```

- Co je na tomhle špatně?
  - Problém blízko nuly
    - b nebo obě ( $a < 0 \ \&\& \ b > 0 \Rightarrow \text{false}$ )
  - **Není symetrické!**

# Epsilon

- Situace se nám komplikuje...
  - <http://floating-point-gui.de/errors/comparison/>

```
float absA = Math.abs(a);
float absB = Math.abs(b);
float diff = Math.abs(a - b);
if (a == b) { // shortcut, handles infinities
    return true;
} else if (a * b == 0) {
    // a or b or both are zero
    // relative error is not meaningful here
    return diff < (epsilon * epsilon);
} else {
    // use relative error
    return diff / (absA + absB) < epsilon;
}
```



# Porovnávání reálných čísel

- „Úkrok stranou“
- Norma IEEE 754 má další pozitivum:
  - „Po sobě následující“ (co to znamená??) reálná čísla jsou bitově reprezentována stejně jako po sobě jdoucí celá čísla

C / C++

```
if (*(int*)&f1 < *(int*)&f2) . . .
```

```
if (f1 < f2) . . .
```



# Porovnávání reálných čísel

- => Lze porovnat jako int
  - Rovnost  $\Leftrightarrow$  neleží mezi více než X jiných
- Ale pozor!
  - speciální případy (NaN, největší číslo)
  - přesnou délku (float – int, double – long)

```
bool almostEquals(float A, float B) {  
    if (A == B) return true;  
    int intDiff = abs(*(int*)&A - *(int*)&B);  
    return (intDiff <= MAX_ERROR);  
}
```

C++

# Porovnávání čísel – důsledky

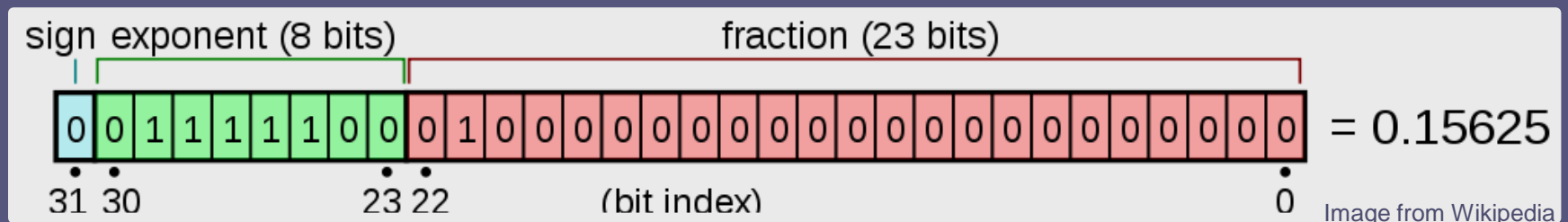
- To se nám to zkomplikovalo, že?

## Praktické důsledky:

- Double místo Float
- Pokud záleží na rovnosti, musí být EPS
  - Známe-li rozsahy čísel, může být pevné
    - Naštěstí pro nás, v těchto úlohách většinou rozsahy známe...
  - Neznáme-li rozsahy, je to složitější

# Speciální hodnoty

- Interpretace:
  - Exponent: +127 („offsetový kód“)
    - 00 i FF je speciální (tj. použito -126 až +127)
  - Mantisa: implicitní 1 na začátku



# Speciální hodnoty

- **+0 / -0 (EXP=x00, M=0)**
  - Tisk
  - Některé operace (1/x)
- **subnormal (EXP=x00, M≠0)**
  - není implicitní 1
  - malá čísla blízka nule
  - => postupná ztráta přesnosti

# Speciální hodnoty

- $+\infty$  /  $-\infty$  (EXP=0xFF, M=0)
  - Lze s nimi počítat
- NaN (EXP=0xFF, M≠0)
  - Nedefinováno / chyba atp.
  - „tichý“ x „signalizační“

# Když to potřebujeme přesně

- **Pevná desetinná čárka**
  - Např. peněžní částky (haléře místo korun)
    - Pozor na rozsah!
- **Čísla s vysokou přesností** (Java: `BigDecimal`)
  - Ukládány číslice (desítkové)
  - Počítá se s nimi „jako ve škole“
  - Rychlost, obzvláště u dlouhých
  - Chyba při nekonečném rozvoji (!)
- **Racionální čísla**
- **Decimal floating point**

# Racionální čísla

- **Reprezentace: čítec + jmenovatel**
- **Operace**
  - **Násobení a dělení – relativně snadné**
  - **Porovnávání – víceméně také**
    - **Pozor na rozsah**
  - **Sčítání a odčítání – společný jmenovatel**
    - **Jak?**

# Největší společný dělitel

- Eukleidův algoritmus
- $\text{gcd}(U, V)$ 
  - $V=0 \Rightarrow \text{return } U$
  - $\text{gcd}(V, U \bmod V)$
- Složitost  $O(\log v)$



# Decimal Floating Point

- IEEE 754-2008
- Mantisa v desítkové soustavě
- Úsporné mapování na bity
- Pozor, tady už neplatí, že větší číslo = větší reprezentace

# Decimal Floating Point

- Různé velikosti

| IEEE 852      | Decimal32  | Decimal64     | Decimal128    |
|---------------|------------|---------------|---------------|
| Bity mantisa  | 23,25      | 53,15         | 112,95        |
| Bity exponent | 7,58       | 9,58          | 13,58         |
| Max mantisa   | $10^7 - 1$ | $10^{16} - 1$ | $10^{34} - 1$ |
| Min exponent  | -95        | -383          | -6143         |
| Max exponent  | +96        | +384          | +6144         |

- Kombinování bitů

# Decimal Floating Point

## ■ Kombinování bitů

(from wikipedia.org)

| Sign  | Combination | Exponent continuation | Coefficient continuation |
|-------|-------------|-----------------------|--------------------------|
| 1 bit | 5 bits      | 6 bits                | 20 bits                  |
| S     | mmmmm       | xxxxxx                | cccccccccccccccccccc     |

| Kombinationsfeld |    |    |    |    | MSBs des |             | Kod. Wert | Beschreibung    |
|------------------|----|----|----|----|----------|-------------|-----------|-----------------|
| m4               | m3 | m2 | m1 | m0 | Exp.     | Signifikant |           |                 |
| 0                | 0  | a  | b  | c  | 00       | 0abc        | (0-7)     | Ziffer bis 7    |
| 0                | 1  | a  | b  | c  | 01       | 0abc        |           |                 |
| 1                | 0  | a  | b  | c  | 10       | 0abc        |           |                 |
| 1                | 1  | 0  | 0  | c  | 00       | 100c        | (8-9)     | Ziffer größer 7 |
| 1                | 1  | 0  | 1  | c  | 01       | 100c        |           |                 |
| 1                | 1  | 1  | 0  | c  | 10       | 100c        |           |                 |
| 1                | 1  | 1  | 1  | 0  |          |             |           | ±Infinity       |
| 1                | 1  | 1  | 1  | 1  |          |             |           | NaN             |

# To je pro dnešek vše

## Úlohy pro další týden:

- Jednoduché
- Už bodované
  
- Reálná čísla
- Formátování výstupu
- Interaktivní úloha
- Další (jak efektivně naprogramovat)

