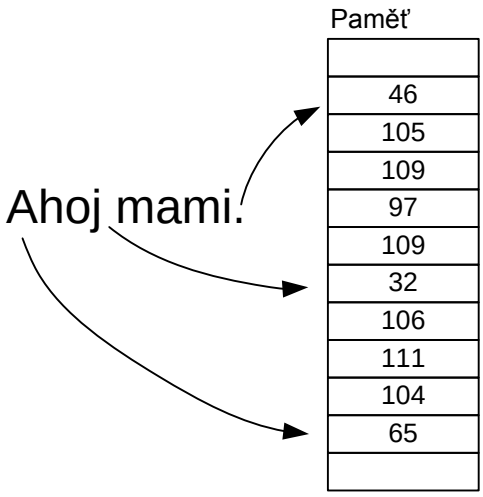
Uložení dat v počítači

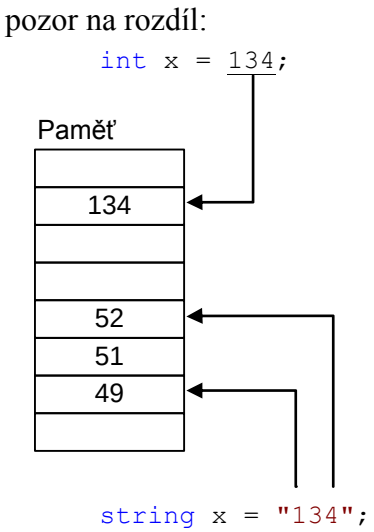
- „Data“ = užitečné, zpracovávané informace
- Kódování (formát) dat = způsob uložení v počítači (nutno vše převést na čísla ve dvojkové soustavě)

Příklady kódování dat

Text

- každému znaku přiřazeno číslo (ASCII tabulka, UNICODE, ...)





Obraz

- RGB (Red Green Blue) barevný model

	R	G	B
červená	255	0	0
zelená	0	255	0
modrá	0	0	255
bílá	255	255	255
černá	0	0	0
obecná	244	115	155

  - pokud barva ⟨0; 255⟩ (8 b) → 255 × 255 × 255 = 16 581 375 kombinací (24 b barvy; „true color“)

Uložení čísel v paměti počítače

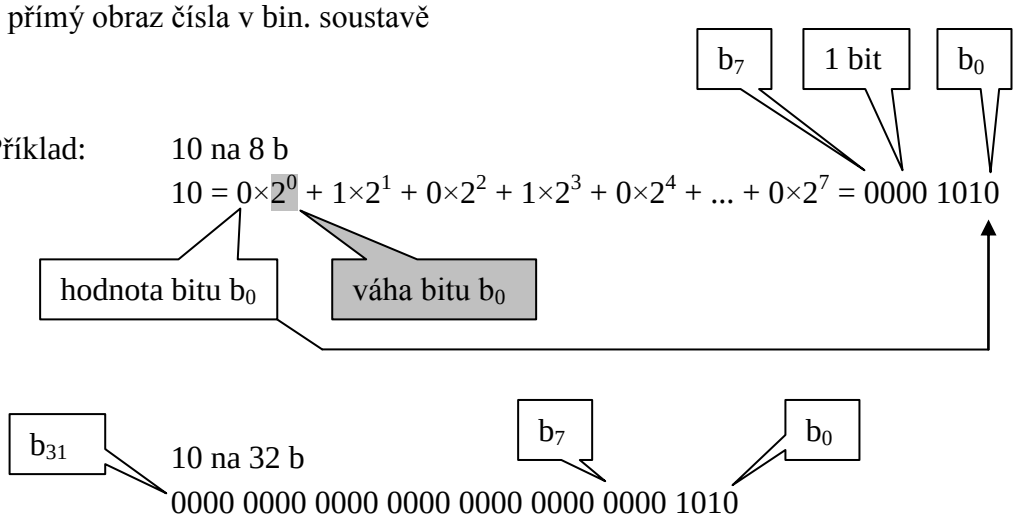
- převody soustav dec ↔ bin viz příloha

Celá čísla

- (fixed point numbers); čísla s pevnou řádovou čárkou

Bez znaménka

- (unsigned); sign = znaménko
- v paměti přímý obraz čísla v bin. soustavě



  - Příklad: 10 na 8 b
    $10 = 0\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^3 + 0\times2^4 + \ldots + 0\times2^7 = 0000\ 1010$

    10 na 32 b
    0000 0000 0000 0000 0000 0000 0000 1010

- kapacita čísla K = počet různých čísel (kombinací) uložitelných do *n* bitů

$$K = 2^n$$

  - příklady
    8 b = 1 B = 2^8 = 256
    32 b = 4 B = 2^{32} = 4 294 967 296

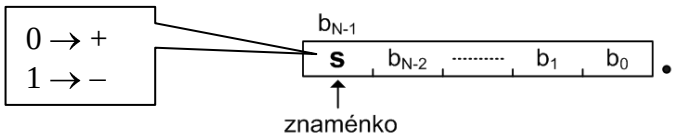
- nejmenší zobrazitelné dekadické číslo: 0
- největší zobrazitelné dekadické číslo:

$$A_{max} = 2^n - 1 = K - 1$$

- příklady:
  $8\ b = 2^8 = 255$
  $32\ b = 2^{32} = 4\ 294\ 967\ 295$

Se znaménkem

- (signed)
- jeden bit → uložení znaménka



- vlastní vyjádření záporného čísla – několik možností (přímý kód, jedničkový doplněk, dvojkový doplněk, kód s posunutou nulou, ...)
- Dvojkový doplněk
  - na PC všechny celočíselné datové typy
  - obraz čísla *A* v doplňkovém kódu D(*A*):

    $D(A) = A$ pro $A >= 0$
    $D(A) = 2^n + A$ pro $A < 0$

  - Příklad ($n = 3$): $A = -1$; $D(A) = 2^3 + (-1) = 7$
  - čísla pro $n = 3$

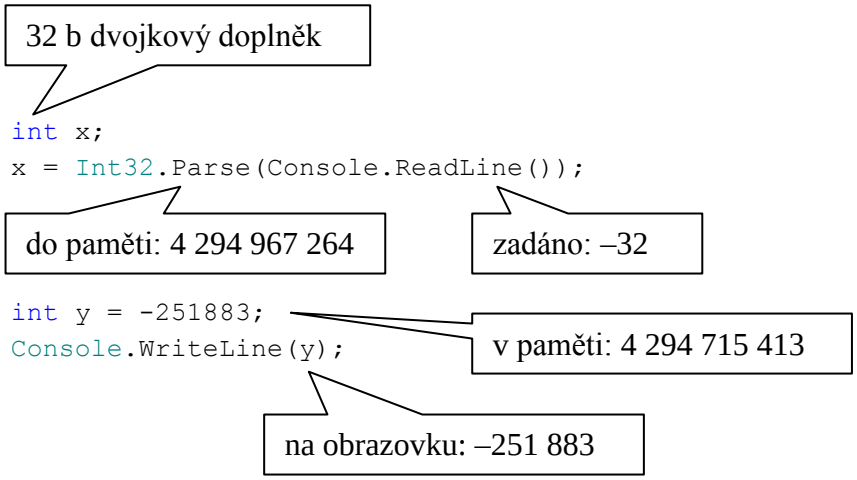
kladná čísla

záporná čísla

v paměti		D(*A*)
bin	dec	
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

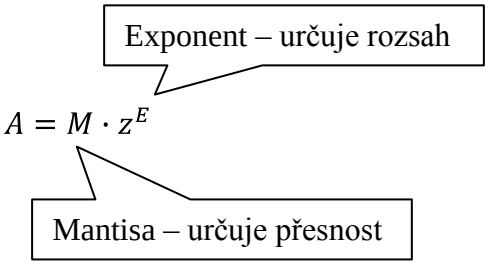
  - nejmenší zobrazitelné dekadické číslo: -2^{n-1}
  - největší zobrazitelné dekadické číslo: $2^{n-1} - 1$

- princip práce I/O metod

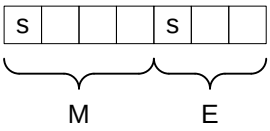


Reálná čísla

- (floating point number); čísla v pohyblivé řádové čárce
- obraz čísla A



- výukový formát (desítková soustava)



$M \in \langle -999; 999 \rangle$

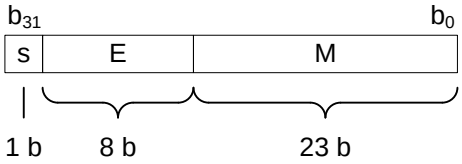
$E \in \langle -99; 99 \rangle$

  - Příklady:

číslo	obraz	poznámka
999×10^{99}	+999+99	max. kladné číslo
1×10^{-99}	+001–99	min. kladné číslo
0	+000+00	
1,2	+012–01 +120–02	několik obrazů
12,345	+123–01	ztráta přesnosti, sk. hodnota 12,3

- standard IEEE 754
  - více na http://en.wikipedia.org/wiki/IEEE754
  - převodník http://babbage.cs.qc.edu/IEEE-754/

- čísla s jednoduchou přesností (32 b, single precision)



  - hodnota čísla:

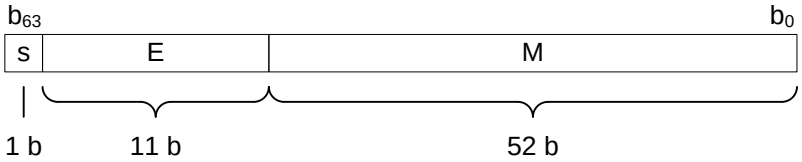
$$A = (-1)^S \cdot 2^{E-127} \cdot 1,M$$

  - přesnost: 7-8 platných číslic
  - příklady

hodnota	single obraz	single hodnota
3	0x40400000	1.5×2^1
3,1	0x 40466666	$1.55\times2^1 = 3.0999999$

uložena nepřesně!!!

- čísla s dvojnásobnou přesností (64 b, double precision)



  - hodnota čísla:

$$A = (-1)^S \cdot 2^{E-1023} \cdot 1,M$$

  - přesnost: 15-16 platných číslic
- lze uložit i „nečísla“
  - NaN (Not a Number) např. SQRT(-1)
  - +∞ např. 10/0
  - –∞

Číselné datové typy v C#

Celočíselné

datový typ	třída CTS	bitů	znam.	rozsah
`sbyte`	`SByte`	8	s	–128 ÷127
`byte`	`Byte`	8	bez	0 ÷ 255
`short`	`Int16`	16	s	–32 768 ÷ 32 767
`ushort`	`UInt16`	16	bez	0 ÷ 65 535
`int`	`Int32`	32	s	–2 147 483 648 ÷ 2 147 483 647
`uint`	`UInt32`	32	bez	0 ÷ 4 294 967 295
`long`	`Int64`	64	s	$-9\ 233\times10^{15} \div 9\ 233\times10^{15}$
`ulong`	`UInt64`	64	bez	$0 \div 18\ 466\times10^{15}$

- signed typy → dvojkový doplněk
- přednostně používat `int`

Reálné

datový typ	třída CTS	bitů	Rozsah	platných číslic
`float`	`Float`	32	+(-) 1.5×10^{-45} až +(-)3.4×10^{38}	7
`double`	`Double`	64	+(-) 5.0×10^{-324} až +(-)1.7×10^{308}	15-16
`decimal`	`Decimal`	128		

- `float` nepoužívat (nepřesné), standardně `double`
- `decimal`
  - „pseudo reálné číslo“, pro finanční výpočty
  - oproti `double` zvýšená přesnost, avšak menší rozsah

Poznámky

- reálné vs. celé číslo

```
int y = 3;          00000000000000000000000000000011
float x = 3.0f;     01000000010000000000000000000000
```

jako `float` 4.203895392974451e-45

jako `int` 1 077 936 128

- aritmetické operace s reálnými typy
  - nepřesné

Typová konverze

- (cast expression), přetypování
- = změna datového typu při operacích
- = přepočet mezi obrazy téhož čísla v různých formátech (datových typech)
- princip I

```
int a = 3;
int b = a;
```

prostá bitová kopie

- princip II

```
int a = 3;
float b = a;
```

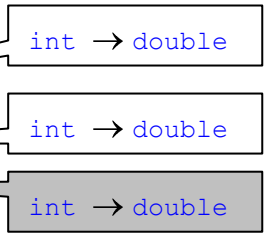
překladač musí vložit kód po přepočet `3` na `3.0f`

Implicitní

- automatická konverze z „nižších“ typů na „vyšší“ – nehrozí ztráta informace (v rozsahu čísel), např.: `byte` → `int` → `double`

- příklady

```
int a = 2;
double x, y;

x = a;
y = 10;
x = y + a;
```



Explicitní

- = konverze z „vyšších“ typů na „nižší“
  - reálné typy → celočíselné (oseknutím zlomkové části čísla)
    např. `double → int`
  - „větší“ celočíselné → „menší“ celočíselné
    např. `int → byte`
- potenciální riziko ztráty informace
- syntaxe:

```
(novy_typ)vyraz
```

- příklad:

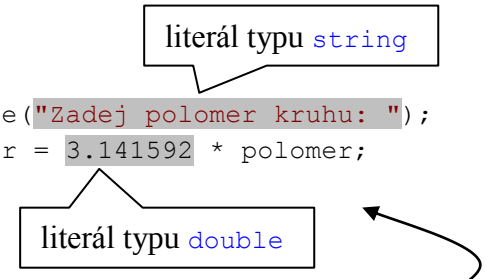
```
int a;
float b=2.87;

a = (int)b;
double x = (double)30;
a = 10 * (int)(a + b);
```

Konstanty a literály

Literál

- = jakýkoli neproměnný objekt datové povahy

```
Console.Write("Zadej polomer kruhu: ");
double prumer = 3.141592 * polomer;
```



- datový typ určen:
  - implicitně – dle povahy literálu
  - explicitně (viz dále)

Celočíselné

- dle soustavy
  - desítková (decadic) – první číslice nesmí být 0 `12` `-47992`
  - šestnáctková (hexadecimal) – `0X12` `0xAAAA`
  - osmičková (octal) `012`
- dle typu
  - automaticky – dle velikosti
    - neuveden → implicitně `int`

```
int x = 32;
```

literál typu `int`

    - větší než `int` (`uint`) → `long` (`ulong`)

```
Console.WriteLine(3148608214);
```

`uint`

  - explicitně: sufix `u` (`U`) = unsigned, `l` (`L`) = long

```
prom1 = 124LU * prom2;
prom1 = 124U * prom2;
```

Reálné

- dle formy zápisu:
  - dekadicky: `1.234`
  - exp. tvar: `13e+23`, `-10.3E-23`
- dle typu
  - neuveden → implicitně `double`
  - explicitně
    - `float` → sufix `f` (`F`)
    - `double` → sufix `d` (`D`)
    - `decimal` → sufix `m` (`M`)

```
Console.WriteLine(14e+3F);
```

`float`

Konstanta

- (konstatní proměnná)
- odstranění „magických čísel“

literál !

```
const double PI = 3.14592;
double obsah = PI * prumer * prumer / 4;
double obvod = PI * prumer;
```

- nutno inicializovat přímo v definici (literálem), dál se nesmí měnit

```
const double PI;  // chyba
PI = 3.14592;     // chyba
```

- konstanta → jakéhokoli datového typu

```
const int MAX = 150;
const string ERROR_MESSAGE = "V programu nastala chyba";
const float K3 = 0.1245f;

Console.WriteLine(ERROR_MESSAGE);
```

CTS třídy

- CTS (Common type specification) – datové typy .NET
- čtení dat z klávesnice

```
datovyTyp id = CTSTrida.Parse(Console...
```

  - Příklad:

```
long promenna= Int64.Parse(Console...
```

- mezní hodnoty rozsahů

```
CTSTrida.MaxValue
CTSTrida.MinValue
```

  - Příklad:

```
Console.WriteLine("max. hodnota v int je {0}", Int32.MaxValue);
```

- „nečísla“ v reálných typech

```
Double.NaN
Double.PositiveInfinity
Double.NegativeInfinity
```

  - Příklad:

```
double a = 10, b = 0;
Console.WriteLine(a/b);
Console.WriteLine(Math.Sqrt(-1));
```