Uvod u programiranje
- predavanja -

listopad 2025.

9. Tipovi podataka u programskom jeziku C

Vodič „08 prije devetog predavanja”

- Očekuje se da ste savladali (na razini na kojoj je obrađeno u vodiču):
  - osnovni tipovi podataka - _Bool

- Pitanja?

Realni tipovi podataka

Tip podatka float

Tip podatka float

- Primjer definicije varijable

```
float temperatura;
```

- Primjeri konstanti tipa float

`2f`	= 2.0
`-2.34F`	= -2.34
`-1.34e5F`	= $-1.34 \cdot 10^{5}$
`9.1093E-31f`	= $9.1093 \cdot 10^{-31}$

- Na koji način su u računalu pohranjene vrijednosti tipa float?

Binarni razlomci

- racionalni broj **q** može se zapisati kao *decimalni* razlomak u obliku

$$\mathbf{q} = \pm (c_n \cdot 10^n + c_{n-1} \cdot 10^{n-1} + \ldots c_1 \cdot 10^1 + c_0 \cdot 10^0$$
$$+ r_1 \cdot 10^{-1} + r_2 \cdot 10^{-2} + \ldots + r_m \cdot 10^{-m})$$

$c_i \in \{0, 1, 2, \ldots, 9\}, i = 0, \ldots, n$

$r_j \in \{0, 1, 2, \ldots, 9\}, j = 1, \ldots, m$

$13.75_{10} = 1 \cdot 10^1 + 3 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$

- slično, *binarni* razlomak **q** može se zapisati u obliku

$$\mathbf{q} = \pm (c_n \cdot 2^n + c_{n-1} \cdot 2^{n-1} + \ldots c_1 \cdot 2^1 + c_0 \cdot 2^0$$
$$+ r_1 \cdot 2^{-1} + r_2 \cdot 2^{-2} + \ldots + r_m \cdot 2^{-m})$$

$c_i \in \{0, 1\}, i = 0, \ldots, n$

$r_j \in \{0, 1\}, j = 1, \ldots, m$

$1101.11_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$

Pretvaranje decimalnog u binarni razlomak

- cijeli dio (dio ispred decimalne točke) pretvara se u cijeli dio (dio ispred binarne točke) binarnog razlomka uzastopnim dijeljenjem brojem 2
- razlomljeni dio (dio iza decimalne točke) pretvara se u razlomljeni dio binarnog razlomka uzastopnim množenjem brojem 2. Prekida se kad se za razlomljeni dio dobije točno nula
  - $9.625_{10} = ?_2$

    $9_{10} = 1001_2$

    . 1 0 1

    $0.625 \cdot 2 = \mathbf{1}.250$

    $0.250 \cdot 2 = \mathbf{0}.5$

    $0.5 \cdot 2 = \mathbf{1}.0$ ← kraj
  - $9.625_{10} = 1001.101_2$

Pretvaranje decimalnog u binarni razlomak

- ako nazivnik nekog racionalnog broja sadrži pribrojnik koji nije potencija broja 2, tada se decimalni razlomak neće moći prikazati kao binarni broj s konačnim brojem znamenaka. Npr.
  - konačnim brojem binarnih znamenaka mogu se prikazati brojevi
    - 1/2, 1/4, 1/8, 1/16, ..., 3/2, 3/4, 3/8, ..., 5/2, 5/4, 5/8, ...
  - konačnim brojem binarnih znamenaka ne mogu se prikazati brojevi
    - 1/3, 1/5, 1/6, 1/7, 1/9, 1/10, ..., 2/3, 2/5, 2/7, ...
- također očito: ako realni broj nije decimalni razlomak (ne može se prikazati s konačnim brojem znamenaka iza decimalne točke) tada se neće moći prikazati niti kao binarni broj s konačnim brojem znamenaka

Pretvaranje decimalnog u binarni razlomak

- $3/10 = 0.3_{10} = ?_2$

. 0 1 0 0 1 1 0 0 1

$0.3 \cdot 2 = \mathbf{0}.6$

$0.6 \cdot 2 = \mathbf{1}.2$

$0.2 \cdot 2 = \mathbf{0}.4$

$0.4 \cdot 2 = \mathbf{0}.8$

$0.8 \cdot 2 = \mathbf{1}.6$

$0.6 \cdot 2 = \mathbf{1}.2$

$0.2 \cdot 2 = \mathbf{0}.4$

...

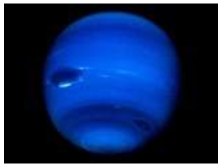
dok se ne dosegne zadovoljavajuća ili moguća preciznost

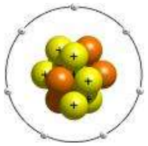
Množenje binarnog broja s potencijama broja 2

- binarni broj se množi s potencijama baze 2 tako da se binarna točka pomakne odgovarajući broj mjesta desno ili lijevo, ovisno o tome je li predznak potencije pozitivan ili negativan
- Primjer:
  - $111.000 \cdot 2^{2} = 11100.0$
  - $0001.101 \cdot 2^{-3} = 0.001101$

Prikaz vrlo velikih i vrlo malih brojeva

- Kako inženjeri i znanstvenici prikazuju vrlo velike i vrlo male brojeve?
  - Kolika je prosječna udaljenost Neptuna i Sunca?
    - 450393**0000000** m
  - Koliko iznosi masa elektrona?
    - 0.**000000000000000000000000000000**910938188 kg
  - previše znamenaka (nula) troši se samo za prikaz magnitude broja





- zato se koristi znanstvena notacija: decimalni broj s jednom znamenkom ispred decimalne točke (zareza), pomnožen odgovarajućom potencijom broja 10 (jer je baza brojanja = 10)
  - $4.50393 \cdot 10^{12}$

    eksponent

    mantisa
  - $9.10938188 \cdot 10^{-31}$

Prikaz vrlo velikih i vrlo malih binarnih brojeva

- Slično, binarni razlomak se može prikazati kao binarni broj s jednom binarnom znamenkom ispred binarne točke, pomnožen odgovarajućom potencijom broja 2 (jer je baza brojanja = 2)
- Za broj u takvom obliku kaže se da je normaliziran. Npr.

$$101.11 = 1.0111 \cdot 2^{2}$$

$$0.000000000000010011 = \underbrace{1.0011}_{\text{binarna mantisa}} \cdot 2^{-14}$$

binarni eksponent

- Normalizacija broja omogućava prikaz vrlo velikih i vrlo malih binarnih brojeva, uvijek u istoj formi, bez korištenja velikog broja nula

Prikaz realnih brojeva u računalu

- kako se podatak tipa float pohranjuje u registru računala?
- realni brojevi se u računalu pohranjuju u normaliziranom binarnom obliku
  - naravno, samo realni brojevi koji se mogu prikazati kao binarni razlomci s prihvatljivim brojem binarnih znamenaka
- Kako točno normalizirani binarni broj pohraniti u registar računala?
  - Standardom IEEE 754 propisan je način pohrane realnih brojeva
    - IEEE 754 standardna preciznost - *single precision*
    - IEEE 754 dvostruka preciznost - *double precision*

IEEE - Institute of Electrical and Electronics Engineers

Realni brojevi standardne preciznosti

- IEEE 754 - *single precision*
  - najčešće se koristi za prikaz vrijednosti tipa float
  - binarni razlomak, u normaliziranom obliku, pohranjuje se u ukupno 32 bita (4 bajta) u sljedećem obliku:

31	30 23	22 0
P	**K - karakteristika**	**M - mantisa bez vodećeg bita**

  - 1 bit za pohranu predznaka broja
    - sam sadržaj bita određuje predznak broja
  - 8 bitova za pohranu karakteristike
    - pozitivni ili negativni binarni eksponent (BE) preslikan u uvijek pozitivnu karakteristiku (K)
  - 23 bita za pohranu mantise *bez vodećeg bita*
    - pohrana prvog bita (koji je uvijek jedinica) je nepotrebna

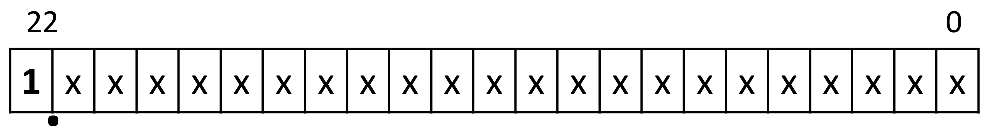
Realni brojevi standardne preciznosti

31	30 … 23	22 … 0
P	**K - karakteristika**	**M - mantisa bez vodećeg bita**

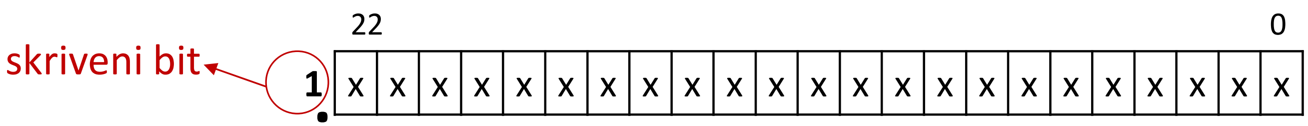
- predznak P: na mjestu najznačajnijeg bita upisuje se
  - 1 ako se radi o negativnom broju
  - 0 ako se radi o pozitivnom broju
- karakteristika K:
  - raspon binarnog eksponenta BE $\in$ [-126, 127]
    - radi izbjegavanja dvojnog komplementa za prikaz negativnih eksponenata, umjesto BE pohranjuje se karakteristika K
    - raspon karakteristike: K $\in$ [0, 255], K = BE + 127 $\Rightarrow$ BE $\in$ [-126, 127]
    - K = 0 i K = 255 se koriste za posebne slučajeve (objašnjeno kasnije)
- mantisa M:
  - ne pohranjuje se vodeći bit mantise - jedinica ispred binarne točke

Prvi bit mantise je implicitno određen

- ako se u 23 bita mora pohraniti *cijela* mantisa
  - iza binarne točke mogu se pohraniti najviše **22** binarne znamenke



  - uočiti: jedina znamenka koja se u normaliziranom broju može pojaviti ispred binarne točke je 1 (osim za prikaz broja nula). Jedan bit se troši nepotrebno jer je njegova vrijednost implicitno poznata
- stoga, vodeći bit (*leading bit*, *hidden bit*) se neće pohranjivati



  - iza binarne točke mogu se pohraniti najviše **23** binarne znamenke
  - povećava se preciznost prikaza broja

Primjer

- Odrediti sadržaj registra u kojeg je pohranjen broj 5.75 prema standardu IEEE 754, standardnoj preciznosti. Sadržaj registra izraziti u binarnom i heksadekadskom brojevnom sustavu
  1. Odrediti bit za predznak: broj je pozitivan, stoga je P = 0
  2. Na temelju decimalnog razlomka izračunati binarni razlomak i normalizirati broj

     $5.75_{10} = 101.11_2 = 1.0111_2 \cdot 2^2$
  3. Izračunati karakteristiku i izraziti je u binarnom obliku, u 8 bitova

     $K = 2_{10} + 127_{10} = 129_{10} = 1000\ 0001_2$
  4. U registar prepisati bit za predznak, karakteristiku i mantisu <u>bez vodećeg bita</u>

31	30 … 23	22 … 0
0	10000001	01110000000000000000000
P	K - karakteristika	M - mantisa bez vodećeg bita

$40\ B8\ 00\ 00_{16}$

Primjer

- 2.0

  $P = 0$

  $10_2 \cdot 2^0 = 1.0_2 \cdot 2^1$

  $BE = 1,\ K = 1 + 127 = 128_{10} = 10000000_2$

  $M = 1.000\ 0000\ \ldots\ 0000_2$

  $0100\ 0000\ 0000\ 0000\ \ldots\ 0000_2 = 4000\ 0000_{16}$

- \- 2.0

  $P = 1$

  sve ostalo je jednako kao za 2.0

  $1100\ 0000\ 0000\ 0000\ \ldots\ 0000_2 = C000\ 0000_{16}$

Primjer

- 4.0

  P = 0

  $100_2 \cdot 2^0 = 1.0_2 \cdot 2^2$

  BE = 2, K = 2 + 127 = 129_{10} = 10000001_2

  M = 1.000 0000 ... 0000_2

  0100 0000 1000 0000 ... 0000_2 = 4080 0000_{16}

- 6.0

  P = 0

  $110_2 \cdot 2^0 = 1.10_2 \cdot 2^2$

  BE = 2, K = 2 + 127 = 129_{10} = 10000001_2

  M = 1.100 0000 ... 0000_2

  0100 0000 1100 0000 ... 0000_2 = 40C0 0000_{16}

Primjer

- 1.0

  P = 0

  $1_2 \cdot 2^0 = 1.0_2 \cdot 2^0$

  BE = 0, K = 0 + 127 = 127_{10} = 01111111_2

  M = 1.000 0000 ... 0000_2

  0011 1111 1000 0000 ... 0000_2 = 3F80 0000_{16}

- 0.75

  P = 0

  $0.11_2 \cdot 2^0 = 1.1_2 \cdot 2^{-1}$

  BE = -1, K = -1 + 127 = 126_{10} = 01111110_2

  M = 1.100 0000 ... 0000_2

  0011 1111 0100 0000 ... 0000_2 = 3F40 0000_{16}

Kalkulator za vježbanje

- Internet stranica na kojoj se nalazi dobar kalkulator za uvježbavanje zadataka s prikazivanjem realnih brojeva

  https://christophervickery.com/IEEE-754/

Posebni slučajevi: prikaz realnog broja 0

- vodeća znamenka (bit) normaliziranog broja u binarnom obliku je uvijek 1, osim u slučaju realnog broja 0. Zato se u IEEE 754 standardu koristi posebno pravilo:
  - kada je K=0 i svi bitovi mantise su postavljeni na 0, radi se o prikazu realnog broja 0 (tada se ne smatra da je vodeći bit implicitno 1)
  - s obzirom da se pri tome bit za predznak može postaviti na 0 ili 1, moguće je prikazati "pozitivnu" i "negativnu" nulu, tj. +0 i -0
    - $00\ 00\ 00\ 00\ 00_{16} \rightarrow +0$
    - $80\ 00\ 00\ 00\ 00_{16} \rightarrow -0$
  - u relacijskim izrazima se te dvije vrijednosti smatraju jednakim

```
float x = 0.f, y = -0.f;
printf("x=%f, y=%f\n", x, y);
if (x == y) printf("x je jednak y");
```

```
x=0.000000, y=-0.000000↵
x je jednak y
```

Posebni slučajevi: denormalizirani broj

- kada je K = 0 i postoje bitovi mantise koji nisu 0, radi se o "denormaliziranom" broju
  - vodeći bit implicitno ima vrijednost 0
  - vrijednost binarnog eksponenta (BE) fiksirana je na -126 (ne koristi se izraz BE = K - 127)
  - omogućuje prikaz još manjih brojeva, ali uz smanjenu preciznost
    - stoga, izbjegavati – koristiti precizniji tip

```
0000 0000 0110 0000 0000 0000 0000 0000
    → 0.11 · 2^-126

0000 0000 0000 0000 0000 0000 0000 0110
    → 0.000 0000 0000 0000 0000 0110 · 2^-126
    → 0.11 · 2^-146
```

Posebni slučajevi: prikaz $+\infty$ i $-\infty$

- kada je K = 255 i svi bitovi mantise su postavljeni na 0, radi se o prikazu $+\infty$ ili $-\infty$, ovisno o bitu za predznak

```
0111 1111 1000 0000 0000 0000 0000 0000 → +∞
1111 1111 1000 0000 0000 0000 0000 0000 → -∞
```

- vrijednosti se mogu dobiti npr.
  - dijeljenjem s nulom u realnoj domeni
  - prekoračenjem najvećeg broja koji se može prikazati

```
float x = 1.f / 0.f;
float y = -1.f / 0.f;
float z = 3.e38f * 2.f;
printf("%f %f %f", x, y, z);
```

```
inf -inf inf
```

Posebni slučajevi: prikaz "ne-broja" (NaN)

- Ako je K=255 i postoje bitovi mantise koji nisu 0, radi se o nedefiniranoj vrijednosti ili vrijednosti koju nije moguće prikazati (*NaN* - *Not a Number*), tj. ne radi se o legalnom prikazu broja
  - bit za predznak nema značenje
  - *NaN* je posljedica obavljanja operacije čiji je rezultat nedefiniran ili se prilikom obavljanja operacije dogodila pogreška, npr.

```
float x = 1.f / 0.f + -1.f / 0.f;
float y = 0.f / 0.f;
float z = -1.f / 0.f * 0.f;
float w = x * 0.f;

printf("%f %f %f %f\n", x, y, z, w);
```

```
nan nan nan nan
```

x → 1111 1111 1100 0000 0000 0000 0000 0000

Raspon realnih brojeva - standardna preciznost

- Najmanji normalizirani pozitivni broj koji se može prikazati je:

  $1.000\ 0000\ 0000\ 0000\ 0000\ 0000_2 \cdot 2^{-126} \approx 1.17 \cdot 10^{-38}$

- Najmanji denormalizirani pozitivni broj koji se može prikazati je:

  $0.000\ 0000\ 0000\ 0000\ 0000\ 0001_2 \cdot 2^{-126}$

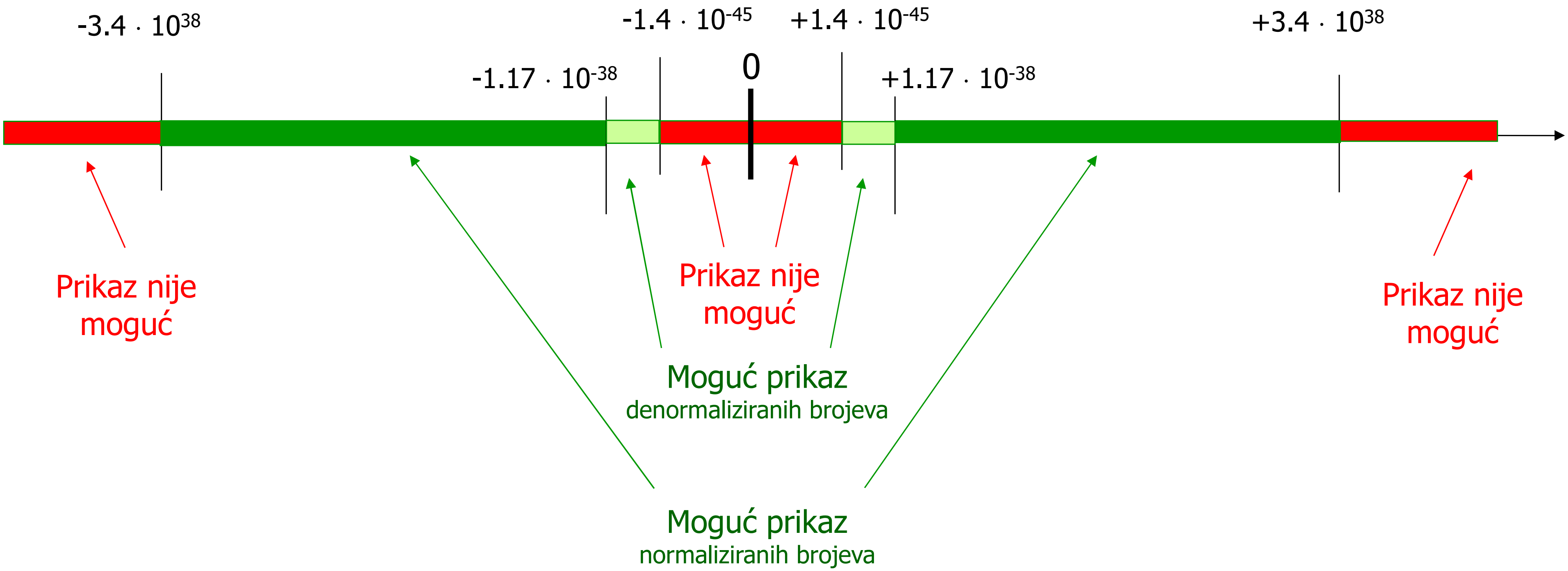
  $= 1 \cdot 2^{-149} \approx 1.4 \cdot 10^{-45}$

- Najveći broj koji se može prikazati je:

  $1.111\ 1111\ 1111\ 1111\ 1111\ 1111_2 \cdot 2^{127}$

  $\approx 1 \cdot 2^{128} \approx 3.4 \cdot 10^{38}$

Raspon realnih brojeva - standardna preciznost



Prikaz cijelih brojeva u računalu (podsjetnik)

- U registru računala s konačnim brojem bitova moguće je prikazati konačan broj različitih brojeva
- Koliko je različitih **cijelih** brojeva moguće prikazati u registru od *n* bitova?
  - 2^n različitih cijelih brojeva

- Skup cijelih brojeva Z je beskonačan - nije moguće prikazati sve brojeve iz skupa Z, ali
  - za točan prikaz **svih** cijelih brojeva iz intervala $[0, 2^n-1]$ ili iz intervala $[-(2^{n-1}), 2^{n-1} - 1]$ dovoljan je registar od *n* bitova

Prikaz realnih brojeva u računalu

- Koliko bitova bi trebao imati registar u kojem bi se mogli točno prikazati **svi** realni brojevi iz intervala [-1.0, 1.0] ?
  - u intervalu ima beskonačno mnogo brojeva, tj. $|[-1.0, 1.0]| \equiv |R|$
  - stoga, potrebno je beskonačno mnogo bitova
- Samo konačan podskup realnih brojeva iz nekog intervala [-a, a] moguće je **točno** (bez pogreške) prikazati u registru s konačnim brojem bitova. Ostali realni brojevi iz tog intervala mogu se pohraniti samo kao njihove približne vrijednosti

Prikaz realnih brojeva u računalu

- Zašto se svi decimalni razlomci, iako imaju konačni broj dekadskih znamenaka iza decimalne točke, ne mogu prikazati konačnim brojem bitova
  - jer ekvivalentni binarni razlomak može imati beskonačni broj binarnih znamenki
    - $0.3_{10} = 0.010011001100110011001 \ldots_{2}$
  - jer binarni broj može imati previše znamenaka u odnosu na broj bitova raspoloživih za pohranu. Npr. za standardnu preciznost:
    - $4194304.125=1.0000000000000000000000001_2 \cdot 2^{22}$
    - broj ima previše znamenaka mantise da bi se mogao bez pogreške prikazati u IEEE 754 formatu standardne preciznosti
    - isti broj se može bez pogreške prikazati u IEEE 754 formatu dvostruke preciznosti (objašnjeno kasnije)

Koliko se različitih realnih brojeva može prikazati

- Za IEEE 754 - standardnu preciznost
  - za svaki K $\in$ [0, 254]
    - moguće su 2^{23} različite mantise
    - moguća su dva predznaka
    - ukupno $255 \cdot 2^{23} \cdot 2$ = 4 278 190 080 različitih realnih brojeva
  - uz K = 255, moguće je prikazati $+\infty$, $-\infty$ i *NaN*

  - bez pogreške je moguće prikazati približno $4.3 \cdot 10^9$ različitih realnih brojeva iz intervala $[-3.4 \cdot 10^{38}, -1.4 \cdot 10^{-45}] \cup [1.4 \cdot 10^{-45}, 3.4 \cdot 10^{38}]$ i broj nula
    - za ostale realne brojeve (njih beskonačno mnogo) iz navedenih intervala moguće je prikazati samo približne vrijednosti (uz veću ili manju pogrešku)
    - realni brojevi izvan navedenih intervala se uopće ne mogu prikazati

Preciznost pohrane realnih brojeva

- Preciznost je svojstvo koje ovisi o količini informacije korištene za prikaz broja. Veća preciznost znači:
  - moguće je prikazati više različitih brojeva
  - brojevi su na brojevnom pravcu međusobno "bliži" (veća rezolucija)
  - smanjuje se najveća moguća pogreška pri prikazu broja

Pogreške pri prikazu realnih brojeva

- x - realni broj kojeg treba pohraniti u registar
- x* - približna vrijednost broja x koja je zaista pohranjena
  - Apsolutna pogreška α

    $$\alpha = |x^* - x|$$

  - Relativna pogreška ρ

    $$\rho = \alpha \, / \, |x|$$

- Primjer:
  - ako je u registru umjesto potrebne vrijednosti `x = 100.57` pohranjena vrijednost `x* = 100.625`

    $$\alpha = |100.625 - 100.57| = 0.055$$

    $$\rho = 0.055 \, / \, |100.57| = 0.000546$$

Pogreške pri prikazu realnih brojeva

- Najveća moguća *relativna pogreška* ovisi o broju bitova mantise
  - Za IEEE 754 standardne preciznosti:

$$\rho \leq 2^{-24} \approx 6 \cdot 10^{-8}$$

- Najveća moguća *apsolutna pogreška* ovisi o broju bitova mantise i konkretnom broju x koji se prikazuje
  - Za IEEE 754 standardne preciznosti:

$$\alpha \leq 2^{-24} \cdot |x| \approx 6 \cdot 10^{-8} \cdot |x|$$

Primjer

- Najveća apsolutna pogreška koja se uz standardnu preciznost može očekivati pri pohrani realnog broja 332.3452:

  $$\alpha \le 6 \cdot 10^{-8} \cdot |332.3452| \approx 2 \cdot 10^{-5}$$

```
float f1 = 332.3452f;
printf("%19.15f", f1);
```

  - očekuje se da će biti pohranjen broj $332.3452 \pm 2 \cdot 10^{-5}$

```
332.345214843750000
```

  - Zaista, apsolutna pogreška je $1.484375 \cdot 10^{-5}$, što je manje od $2 \cdot 10^{-5}$

Primjer

- Najveća apsolutna pogreška koja se uz standardnu preciznost može očekivati pri pohrani realnog broja 0.7:

$$\alpha \leq 6 \cdot 10^{-8} \cdot |0.7| \approx 4.2 \cdot 10^{-8}$$

```
float f2 = 0.7f;
printf("%19.15f", f2);
```

  - očekuje se da će biti pohranjen broj 0.7 $\pm 4.2 \cdot 10^{-8}$

```
0.699999988079071
```

  - Zaista, apsolutna pogreška je $1.1920929 \cdot 10^{-8}$, što je manje od $4.2 \cdot 10^{-8}$

Zadatak

Znamo da se puno float brojeva ne može ispravno pohraniti u int, npr.

```
int a = 3.14f; // a= 3
```

ali mogu li se svi cijeli brojevi bez greške pohraniti u float?

```
float f = 3;    //  f je (naravno?) 3
```

Možemo li proći neki cijeli broj koji se ne pohranjuje dobro?
Koliko ih ima?

Prije sljedećeg predavanja

- Edgar:
  - Tutorial: **09. Prije desetog predavanja**
  - **9. vježbe uz predavanja**