

Uvod u programiranje

algoritam, program, programiranje

Algoritam

- Pravilo (ili skup pravila) kojim se opisuje kako riješiti neki problem i koje posjeduje sljedeća svojstva:
 - algoritam je precizan
 - algoritam je jednoznačan
 - algoritam obuhvaća konačni broj koraka; svaki korak je opisan instrukcijom
 - za algoritam su definirani početni objekti (koji pripadaju nekoj klasi objekata) nad kojima se obavljaju operacije
 - ishod obavljanja algoritma je skup završnih objekata (rezultat), tj. algoritam je djelotvoran (*effective*)
- Postupak obavljanja algoritma je **algoritamski proces**

Primjer algoritma – kiseljenje krastavaca

- Početni objekti:
 - 5 kg krastavaca, 1 l alkoholnog octa (9%), 30 dag šećera, 10 dag soli, kopar, papar
- krastavce i kopar oprati i posložiti u čiste staklenke
- u 2 l vode dodati ocat, šećer, sol i papar
- zakuhati uz miješanje
- vruću otopinu uliti u staklenke
- staklenke zatvoriti celofanom i gumaticom
- složiti staklenke u široki lonac napunjen vodom do grla staklenki
- ako je toplomjer raspoloživ
 - zagrijati vodu do 80 stupnjeva
- inače
 - zagrijavati dok se s dna ne počnu dizati mjehurići zraka
- ostaviti stajati barem 24 sata
- Završni objekti:
 - kiseli krastavci á la FER

Algoritmi i programi

- *Program* - opis algoritma koji u nekom programskom jeziku jednoznačno određuje što računalo treba napraviti.
- *Programiranje* – proces opisivanja algoritma nekim od programskih jezika
- Postupci izrade algoritama nisu jednoznačni te zahtijevaju i kreativnost.
- Koristit će se programski jezik C. Za sažeti opis složenijih algoritama koristit će se pseudokod ili dijagram toka.

Primjer

- Programski zadatak: s tipkovnice pročitati dva različita cijela broja, na zaslon ispisati pročitane brojeve i veći od pročitana dva broja

Primjer - nastavak

- Pseudokod koji koristi isključivo termine govornog jezika

```
pročitaj dva cijela broja  
ispiši pročitane brojeve  
odredi veći broj  
ispiši rezultat
```

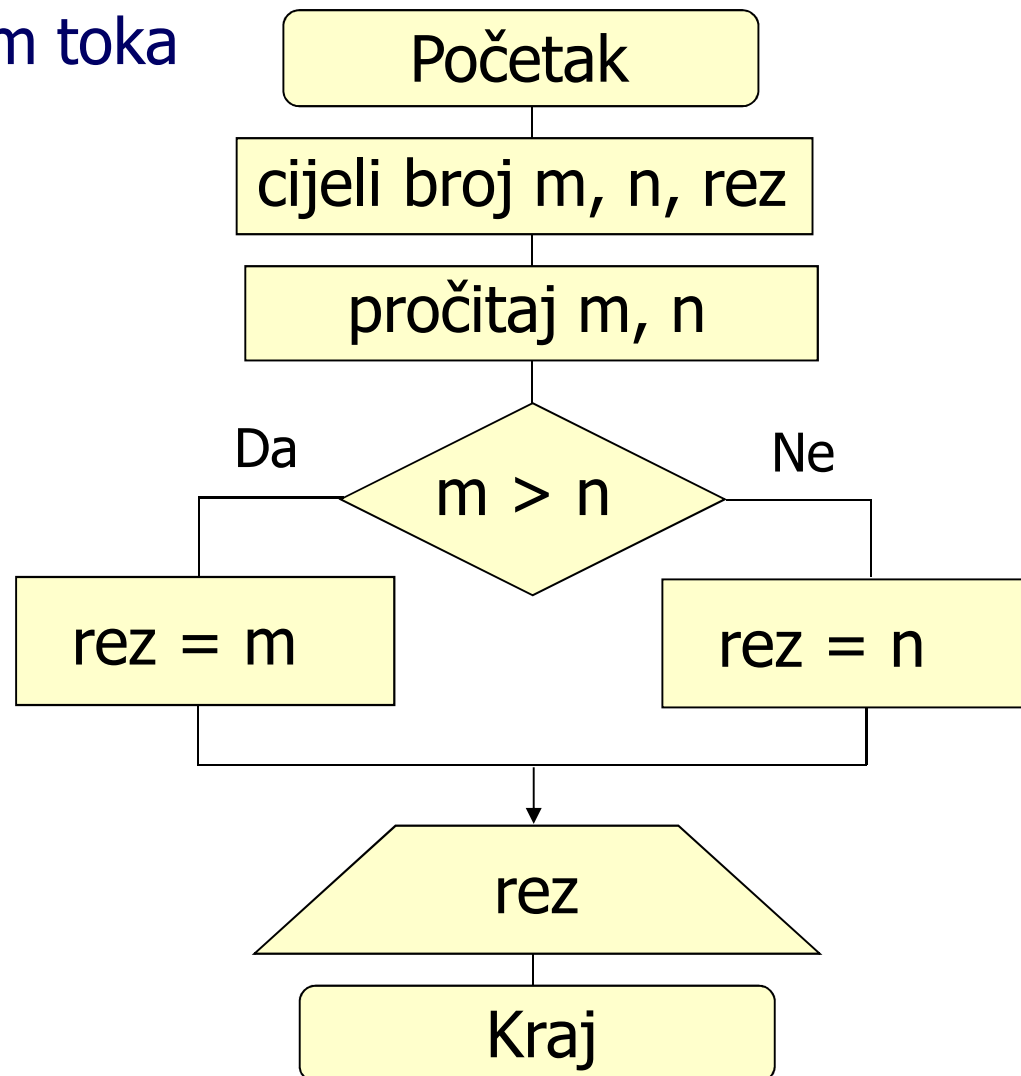
Primjer - nastavak

- Pseudokod koji koristi uobičajene simbole

```
pročitaj (m, n)
ispiši (m, n)
{odredi veći broj}
ako je m > n tada
    rez := m
inače
    rez := n
ispiši (rez)
kraj
```

Primjer - nastavak

- Dijagram toka



Primjer - nastavak

- Kôd u programskom jeziku C

```
#include <stdio.h>
int main() {
    int m, n, rez;
    /* učitaj i ispisi dva cijela broja */
    scanf("%d %d", &m, &n);
    printf("Ucitani su %d i %d\n", m, n);
    /* odredi veci broj */
    if ( m > n ) {
        rez = m;
    } else {
        rez = n;
    }
    /* ispisi rezultat */
    printf("Veci broj je %d\n", rez);
    return 0;
}
```

Objašnjenje

```
#include <stdio.h>
```

- uputa pretprocesoru: uključuje u program prije prevođenja standardno zaglavlje `<stdio.h>` koje sadrži definicije/deklaracije struktura, vrijednosti, makroinstrukcija i funkcija za standardne ulazno-izlazne jedinice
- ugraditi na početak svakog programa koji koristi funkcije `scanf` ili `printf`

Objašnjenje

```
int main() {  
    ...  
    ...  
    return 0;  
}
```

- glavna funkcija koja predstavlja mjesto gdje počinje izvršavanje programa. Svaki C program mora sadržavati točno jednu main funkciju.
- `int` ispred `main` znači da funkcija u pozivajući program vraća cijeli broj
- funkcija završava naredbom `return` koja u pozivajući program vraća rezultat. Za sada, uvijek napisati `return 0;`

Objašnjenje

```
int m, n, rez;
```

- definicija varijabli. Varijabla je prostor u memoriji računala, poznate veličine, kojem je dodijeljeno ime i čiji se sadržaj može mijenjati. Naredbom su definirane 3 cjelobrojne varijable u koje se mogu pohranjivati cijeli brojevi

```
float x;  
float y;
```

- naredbama su definirane realne varijable x i y u koje se mogu pohranjivati realni brojevi

Objašnjenje

```
/* učitaj dva cijela broja */
```

- komentar koji nema utjecaja na izvršavanje programa

```
/* učitaj  
   dva cijela  
   broja */
```

- komentar se može protezati kroz više redaka programa

Objašnjenje

```
scanf("%d %d", &m, &n);
```

funkcija za učitavanje vrijednosti s tipkovnice. Kao argumenti se navode

- **format** (ovisi o tipovima varijabli u koje se učitavaju vrijednosti)
 - korištenjem specifikacije `%d` učitavaju se cjelobrojne vrijednosti
- **adrese varijabli** u koje se učitavaju vrijednosti. Adrese varijabli se označavaju s `&imeVarijable`

npr, preko tipkovnice je uneseno ➡ 3 5↵

- nakon obavljanja naredbe `scanf` varijabla `m` ima vrijednost 3, varijabla `n` ima vrijednost 5

- korištenjem specifikacije `%f` učitavaju se realne vrijednosti

```
float x, y, z;  
scanf("%f %f %f", &x, &y, &z);
```

➡ 1.15 2 3.5↵

▪ `x ← 1.15`

`y ← 2.0`

`z ← 3.5`

Objašnjenje

```
printf("%d %d\n", m, n);
```

- funkcija za ispisivanje na zaslon. Kao argumenti se navode
 - **format (ovisi o tipovima vrijednosti koje se ispisuju)**
 - ako su vrijednosti cjelobrojne, koristi se specifikacija %d
 - ako su vrijednosti realne, koristi se specifikacija %f
 - **vrijednosti koje se ispisuju.** To (između ostalog) mogu biti varijable i konstante
- \n u formatu predstavlja uputu za skok u novi red
- uz pretpostavku da su vrijednosti varijabli m=156, n=20, na zaslon će se ispisati

```
156 20↵
```

Objašnjenje

- specifikacijom se može utjecati na širinu ispisa, npr.

```
printf("%d,%5d,%2d", 10, 20, 30);
```



10, 20, 30

```
printf("%d,%d\novo je novi red%4d", 10, 20, 30);
```



10,20
ovo je novi red 30

Objašnjenje

- kad se ispisuju realne vrijednosti, koristi se specifikacija %f

```
printf("%f %f", 15.2, -3.45);
```



15.200000 -3.450000

- ispisuje se 6 znamenki iza decimalne točke

- specifikacijom se može utjecati na širinu ispisa i broj decimala, npr.

```
printf("%6.2f,%10.4f", 15.2, -3.45);
```



15.20, -3.4500

%6.2f:

ispisuje ukupno 6 znakova, od toga dva iza decimalne točke

%10.4f:

ispisuje ukupno 10 znakova, od toga četiri iza decimalne točke

Objašnjenje

```
rez = m;
```

- naredba za pridruživanje vrijednosti u varijablu
- u varijablu rez pridruži vrijednost koja se nalazi u varijabli m

```
rez = 5;  
rez = 5 * 3;  
rez = 12 / 3;  
rez = m + 5 - 3;  
rez = 3 / 4;
```

- u rez pridruži 5
- u rez pridruži 15
- u rez pridruži 4
- u rez pridruži vrijednost varijable m uvećane za 2
- u rez pridruži 0 !!!

```
float x;  
x = 3. / 4;
```

- u x pridruži 0.75

Objašnjenje

```
if ( m > n ) {  
    rez = m;  
} else {  
    rez = n;  
}
```

- naredba za kontrolu toka programa
 - ako se uvjet u zagradama iza if izračuna kao istina, obavljaju se naredbe unutar prvih vitičastih zagrada
 - inače se obavljaju naredbe unutar vitičastih zagrada iza else
- `if (m > n)` ako je sadržaj m veći od sadržaja n
- `if (m != n)` ako je sadržaj m različit od sadržaja n
- `if (m == n)` ako je sadržaj m jednak sadržaju n
- `>=, <, <=`

Zadatak za vježbu

- S tipkovnice učitati dva cijela broja. Na zaslon ispisati, ovisno o vrijednostima koje su učitane, **jednu od** sljedećih poruka:

brojevi su jednaki

prvi broj je veci od drugog

prvi broj je manji od drugog

Rješenje (varijanta 1)

```
#include <stdio.h>
int main() {
    int i, j;
    scanf("%d %d", &i, &j);
    if ( i > j ) {
        printf("prvi broj je veci od drugog");
    }
    if ( i < j ) {
        printf("prvi broj je manji od drugog");
    }
    if ( i == j ) {
        printf("brojevi su jednaki");
    }
    return 0;
}
```

Rješenje (varijanta 2)

```
#include <stdio.h>
int main() {
    int i, j;
    scanf("%d %d", &i, &j);
    if ( i > j ) {
        printf("prvi broj je veci od drugog");
    } else {
        if ( i < j ) {
            printf("prvi broj je manji od drugog");
        } else {
            printf("brojevi su jednaki");
        }
    }
    return 0;
}
```

- Koja je varijanta bolja? Zašto?

Zadatak za vježbu

- S tipkovnice učitati polumjer kruga (realni broj). Ako je učitani broj veći od nule, izračunati opseg i površinu kruga, te na zaslon ispisati:

zadani polumjer je: xxxxxx.xx

opseg kruga je: xxxxxx.xx

povrsina kruga je: xxxxxx.xx

- inače ispisati

polumjer kruga je neispravno zadan: xxxxxx.xx

Domaća zadaća (1) – upoznavanje sa sustavima koji se koriste na predmetu

- Svaki student treba imati pristup webu FER-a (ako ga nema mora se javiti u CIP i zatražiti pristup webu FER-a)
- S weba FER-a preuzeti i proučiti datoteke iz mape *PIPI-zimski semestar 2012/2013* ▶ *Upute:*
 - Upute za provjere znanja na računalu putem sustava AHyCO
- Provjeriti možete li se prijaviti na sustav AHyCo (ahyco.fer.hr). U tu svrhu se koriste korisničko ime i lozinka za pristup webu FER-a. Sustav AHyCo će se koristiti za pisanje provjera znanja na računalu (tzv. bliceva)

Domaća zadaća (2) – instalacija i isprobavanje razvojnog okruženja


- ❑ Za pripreme laboratorijskih vježbi i vježbanje programiranja studenti trebaju preuzeti:
 - ❑ MinGW <http://mingw.org/>
- ❑ Studenti trebaju s weba FER-a preuzeti i proučiti datoteke iz mape *PIPI-zimski semestar 2012/2013* ► *Upute*:
 - ❑ Upute za korištenje MinGW paketa i GCC prevodioca

Vrste programske podrške

- Sistemska programska podrška
 - Operacijski sustavi (MS-DOS, UNIX/Linux, Windows)
 - Uslužni (*utility*) programi (prevodioci, uređivači teksta):
 - Servisi (Internet poslužitelj, poslužitelj baze podataka)

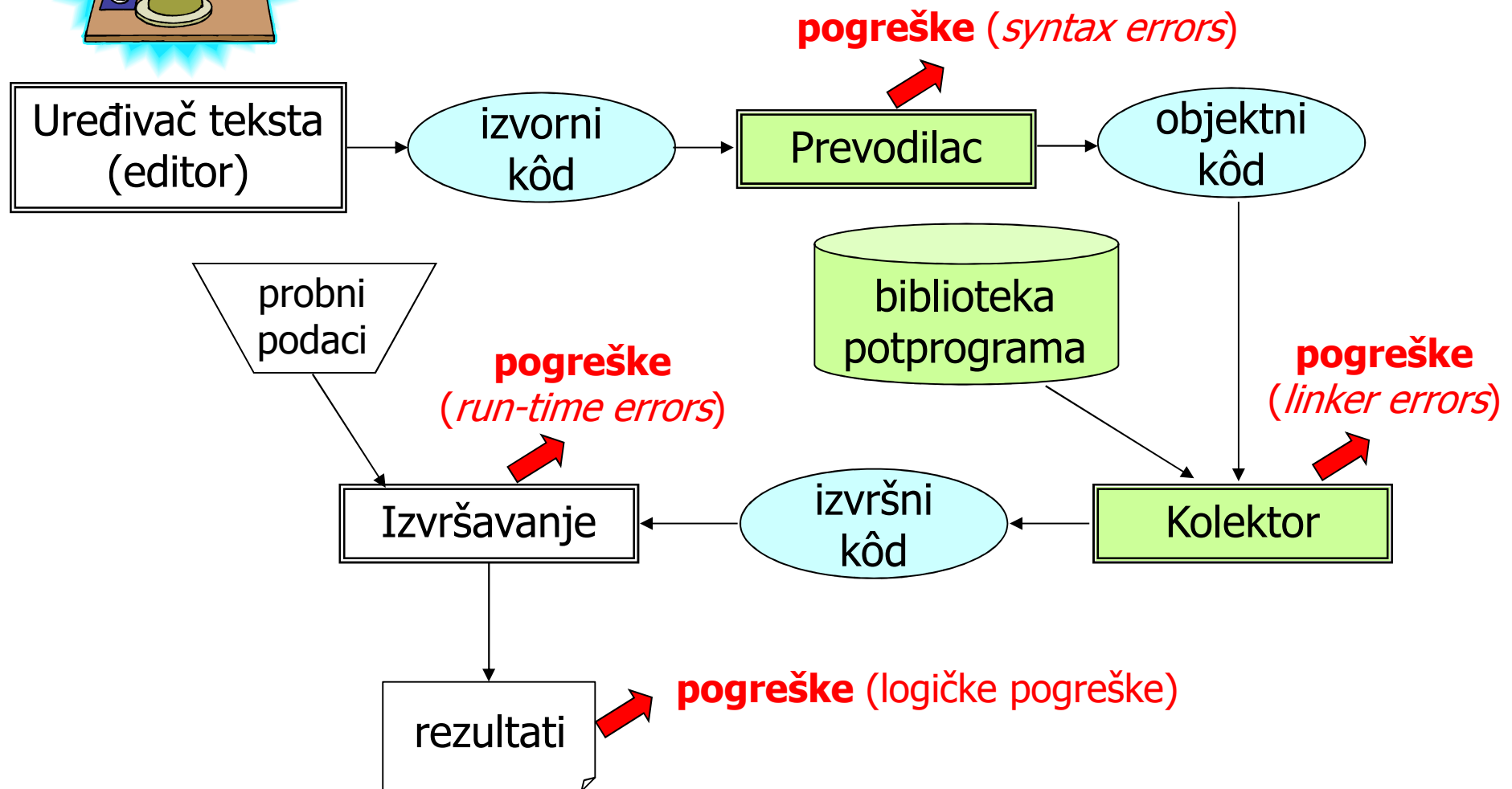
- Aplikativna (namjenska, primijenjena) programska podrška
 - Rješava probleme specifične za neku struku
 - Tablični kalkulatori (npr. Excel), obrađivači teksta (npr. Word), mrežno planiranje (npr. Project)...
 - IDE (*Integrated Development Environment*): sustav za podršku pri razvoju softvera. Uobičajeno, obuhvaća sintaksno osjetljivi editor, prevodilac, podsustav za testiranje, itd.
 - Eclipse, Visual Studio .NET 2008

Redoslijed rješavanja manjih programa

1. Uočavanje (identifikacija) problema i postavljanje programskog zadatka
 2. Oblikovanje programa (razvoj algoritma)
 3. Konverzija algoritma u logiku razumljivu računalu
 4. Kodiranje
 5. Upis programskog kôda u računalu
 6. Prevođenje (kompilacija) programa
 7. Ispravljanje formalnih pogrešaka
 8. Kolekcija programa (stvaranje izvršnog programa)
 9. Izvođenje programa s test podacima
 10. Ispravljanje uočenih logičkih pogrešaka
 11. Korištenje programa s aktuelnim podacima
- 
- ```
graph TD; 10 --> 5; 9 --> 5;
```



# Programiranje u užem smislu



# Programiranje u užem smislu

---

- Unos izvornog programa (*source code*)
  - ASCII uređivač teksta (Notepad, vi, ...)
  - Uređivač teksta ugrađen u radnu okolinu programera (Eclipse, MS Visual Studio)
- Prevođenje izvornog programa u objektni program
  - Poziv prevodioca (*compiler*)
  - Prevodilac otkriva sintaktičke (pravopisne, formalne) pogreške
    - programer ispravlja izvorni kôd i ponovo pokreće prevođenje
- Kolekcija (povezivanje) prevedenog programa u izvršni (apsolutni) program
  - Poziv kolektora (*linker*)
  - Povezuju se potrebne potprogramske biblioteke (`stdio.h`, `math.h`, ...)
  - Kolektor otkriva pogreške
    - programer ispravlja izvorni kôd i ponovno pokreće prevođenje

# Programiranje u užem smislu

---

- Izvođenje izvršnog programa
  - Definiranje skupova ulaznih probnih podataka i očekivanih rezultata
  - Izvršavanje programa na osnovi probnih podataka
    - Pogreške koje se otkrivaju prilikom izvršavanja (*run-time errors*)
      - npr. **Division by zero**
    - Logičke pogreške
      - program "radi" (ne dojavljuje pogreške), ali daje pogrešne rezultate
    - programer ispravlja izvorni kôd i ponovo pokreće prevođenje/izvođenje

# Primjer

---

- Programski zadatak  
    **pronaći najveći od tri zadana broja**
- Pseudokod koji koristi isključivo termine govornog jezika

**pročitaj tri realna broja  
ispiši pročitane brojeve  
odredi najveći broj  
ispiši nađeni broj**

# Primjer - nastavak

- Pseudokod koji koristi uobičajene simbole

```
pročitaj (x,y,z)
ispiši (x,y,z)
{odredi najveći broj}
 ako je x > y tada
 | ako je x > z tada
 | rez := x
 | inače
 | rez := z
 inače
 | ako je y > z tada
 | rez := y
 | inače
 | rez := z
ispiši (rez)
kraj
```



# Primjer - nastavak

- Unapređenje prethodnog rješenja

pročitaj (x,y,z)

**ispiši (x,y,z)**

**{odredi najveći broj}**

```
rez := z
```

ako je  $x > y$  tada

## inače

**ispiši (rez)**

kraj

ako je  $x > z$  tada

```
rez := x
```

ako je  $y > z$  tada

```
rez := y
```

# Primjer - nastavak

## ❑ Kôd u programskom jeziku C

```
#include <stdio.h>
int main() {
 float x, y, z, rez;
 scanf("%f %f %f", &x, &y, &z);
 printf("%f %f %f \n", x, y, z);
 /* odredi najveći broj */
 rez = z;
 if (x > y) {
 if (x > z) rez = x;
 } else {
 if (y > z) rez = y;
 }
 printf("%f \n", rez);
 return 0;
}
```

# Budite prevodilac (*compiler*) i komentirajte ponuđeno rješenje

---

```
#include <stdio.h>
int main() {
 float pi, triCetvrtPi;
 pi = 3.14159;
 triCetvrtPi = 3/4*pi;
 print("tri cetvrt pi = %f \n", triCetvrtPi);
 return 0;
}
```

sintaktička  
pogreška

pogreška  
povezivanja  
(linkanja)

logička pogreška

# Zadatak za vježbu

---

- Napisati pseudokod, nacrtati dijagram toka, te napisati C program za sljedeći problem:
  - Učitati koeficijente dvaju pravaca
    - $y = a_1 x + b_1$  i  $y = a_2 x + b_2$
  - Ovisno o vrijednostima koeficijenata, ispisati poruku da su pravci paralelni ili ispisati koordinate točke u kojoj se sijeku.
- Upisati, prevesti i testirati program

---

# Uvod u C-programiranje

## opća pravila pisanja C-programa

# Opća pravila pisanja C programa

---

- C razlikuje velika i mala slova. Npr:

`sum`

`Sum`

`SUM`

- C je jezik slobodnog formata (nema pravila koja propisuju stil pisanja)
- mjesto početka naredbe u retku je proizvoljno
- dopušteno je stavljanje više naredbi u istom retku. Npr:  
`int i,n; printf("Unesite n: "); scanf("%d", &n);`
- poželjno je umetanje praznina i praznih redova

# Primjer - što radi ovaj program?

---

```
#include <stdio.h>
int main() {float x, y, z, rez;scanf("%f %f %f",
&x, &y, &z); printf
("%f %f %f \n", x, y
, z); rez
= z ; if(x
> y) {if (x>z) rez
= x;} else{if (
y >
z) rez= y
;
}printf("%f \n"
, rez);
return 0;}
```

# Primjer - što radi ovaj program?

```
#include <stdio.h>
int main() {
 float x, y, z, rez;
 scanf("%f %f %f", &x, &y, &z);
 printf("%f %f %f \n", x, y, z);
 rez = z;
 if (x > y) {
 if (x > z) rez = x;
 } else {
 if (y > z) rez = y;
 }
 printf("%f \n", rez);
 return 0;
}
```

**Različiti stilovi**

```
if (x > y) {
 rez = x;
}
```

```
if (x > y)
{
 rez = x;
}
```

```
if (x > y)
{
 rez = x;
}
```

```
if (x > y)
{
 rez = x;
}
```



# Ključne riječi

---

- predefinirani identifikatori koji za prevodioca imaju posebno značenje
- ključne riječi se pišu malim slovima
- Prema ANSI standardu C ima sljedeće 32 ključne riječi:

|                       |                     |                       |                       |
|-----------------------|---------------------|-----------------------|-----------------------|
| <code>auto</code>     | <code>double</code> | <code>int</code>      | <code>struct</code>   |
| <code>break</code>    | <code>else</code>   | <code>long</code>     | <code>switch</code>   |
| <code>case</code>     | <code>enum</code>   | <code>register</code> | <code>typedef</code>  |
| <code>char</code>     | <code>extern</code> | <code>return</code>   | <code>union</code>    |
| <code>const</code>    | <code>float</code>  | <code>short</code>    | <code>unsigned</code> |
| <code>continue</code> | <code>for</code>    | <code>signed</code>   | <code>void</code>     |
| <code>default</code>  | <code>goto</code>   | <code>sizeof</code>   | <code>volatile</code> |
| <code>do</code>       | <code>if</code>     | <code>static</code>   | <code>while</code>    |

ANSI - American National Standards Institute

<http://refcards.com/refcard/ansi-c-silvermanj>

# Struktura C programa

---

- C program se sastoji od imenovanih blokova, deklaracija/definicija varijabli, direktiva pretprocesoru
  - imenovani blokovi se nazivaju **funkcije**. Za nazive funkcija se ne smiju koristiti ključne riječi.
  - deklaracija opisuje naziv i tip varijable.
  - definicija je deklaracija kojom se osim opisa varijable, rezervira prostor u memoriji
- blok započinje znakom **{** i završava znakom **}**
- blok obuhvaća deklaracije/definicije, naredbe (*statement*) i neimenovane blokove
- svaka naredba i deklaracija/definicija mora završavati znakom **;**
- blok NE završava znakom **;** tj. iza znaka **}** ne stavlja se **;**

# Struktura C programa

---

```
int suma(int i, int j) {
 int k;
 {
 int m;
 {
 m = i + j;
 k = m;
 }
 }
 return k;
}
```

*imenovani blok (funkcija)*  
*definicija variable*  
*neimenovani blok*  
*definicija variable*  
*neimenovani blok*  
*naredba*  
*naredba*

```
int produkt(int i, int j) {
 ...
}
```

*imenovani blok (funkcija)*

# Struktura C programa

- u C programu mora postojati glavna (`main`) funkcija koja predstavlja mjesto gdje počinje izvršenje programa:

```
int main() {
 programski blok
 return 0;
}
```

**ISPRAVNO**

```
main() {
 programski blok
 return 0;
}
```

**ISPRAVNO ALI SE NE PREPORUČA**

ili

```
void main() {
 programski blok
}
```

**POGREŠNO!!!**

# Struktura C programa

---

```
#include <stdio.h>
```

*direktive pretprocesoru*

```
#define PI 3.14159
```

```
int main() {
```

*funkcija **main***

```
 float r;
```

*definicija varijabli*

```
 float opseg;
```

```
 printf("Unesi polumjer: ");
```

*tijelo funkcije **main***

```
 scanf("%f", &r);
```

```
 opseg = 2 * r * PI;
```

```
 printf("%9.2f\n", opseg);
```

```
 return 0;
```

```
}
```

*kraj funkcije **main***

# Komentari

---

- mogu se protezati kroz više linija
- izbjegavati komentar oblika:  
`printf("Unesi n: ");/* Ispis na zaslonu */`  
zato što program postaje nečitkiji
- nije dopušteno koristiti komentar unutar komentara:  
`/* definicija /* funkcije */ sume */`

# Pretprocesorske naredbe

- `#include <stdio.h>` uključuje u program prije prevođenja standardno zaglavlje `<stdio.h>` koje sadrži definicije/deklaracije struktura, vrijednosti, makroinstrukcija i funkcija za standardne ulazno-izlazne jedinice (na primjer `printf`, `scanf` i druge).

```
/*
 * stdio.h
 * This file has no copyright assigned and is placed in the Public Domain.
 * This file is a part of the mingw-runtime package.
 * No warranty is given; refer to the file DISCLAIMER within the package.
 *
 * Definitions of types and prototypes of functions for standard input and output.
 *
 * NOTE: The file manipulation functions provided by Microsoft seem to
 * work with either slash (/) or backslash (\) as the directory separator.
 *
 */
#ifndef _STDIO_H_
#define _STDIO_H_
...
```

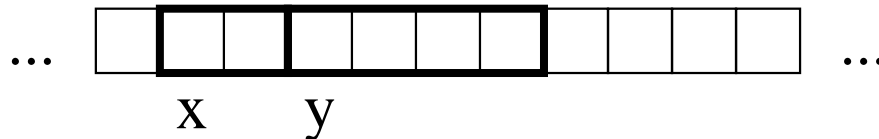
- `#define PI 3.14159` definira simboličku konstantu `PI` i pridjeljuje joj vrijednost. Simboličke konstante su naročito korisne za parametrizaciju programa.

# Variable

- Općenito: promjenljiv podatak (lat. *variabilis*-promjenljiv)
- U programiranju: varijabla je prostor u memoriji računala, poznate veličine, kojemu je dodijeljeno ime i čiji se sadržaj može mijenjati
- Simbolički se prikazuje pravokutnikom uz kojeg stoji ime

x       y

- Smještaj u memoriji računala





# Varijable

- imena varijabli i funkcija su sastavljena od slova i brojki, a prvi znak mora biti slovo ili znak potcrtavanja \_

`suma god_rod x1 pripremni_dio_studija`

~~94god~~ ~~novi datum~~ ~~x1.1~~ ~~matieni broj~~ ~~float~~

- svaka varijabla se obavezno mora definirati/deklarirati prije korištenja

`int i, n;`

`float sum;`

`char pocetno_slovo;`

- velika i mala slova se razlikuju (imena varijabli i funkcija se obično pišu malim slovom, imena simboličkih konstanti velikim)
- duljina može biti proizvoljna (značajno prvih 31 znakova)
- ključne riječi se **ne smiju** koristiti za imena varijabli

---

# Uvod u C-programiranje

## osnovni tipovi podataka

# Osnovni tipovi podataka

---

- **Osnovni tipovi podataka**

`int` - cjelobrojni tip

`float` - realni tip

`double` - realni tip u dvostrukoj preciznosti

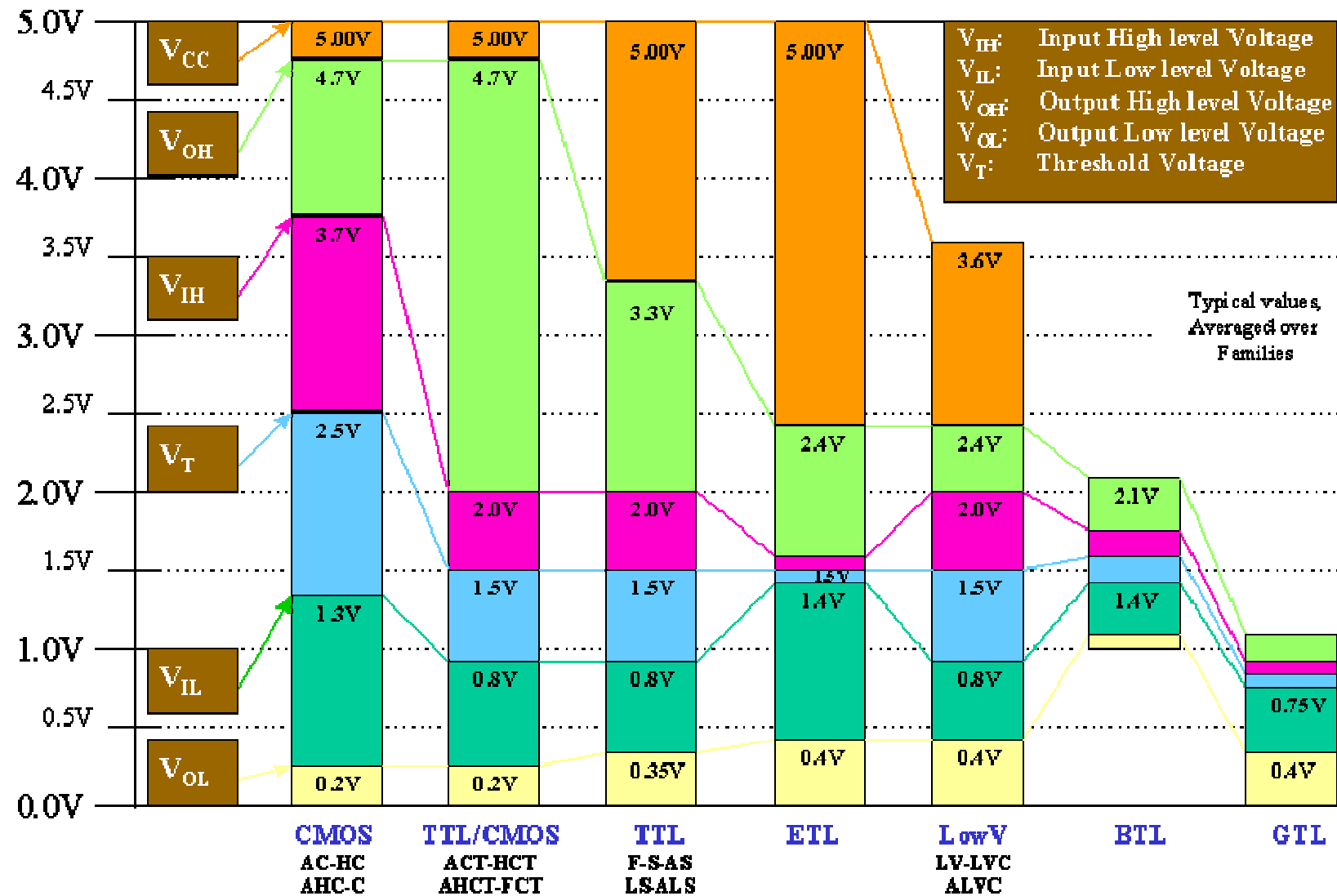
`char` - znakovni tip (ili mali cijeli broj)

# Binarni brojevni sustav

---

- Znamenke su **0** i **1**, dakle baza brojanja **B=2** što određuje **binarni brojevni sustav**
- Iz engleskog ***BI**nary **digi**T* nastalo je ime za najmanju količinu informacije, znamenku binarnog brojevnog sustava **BIT**.
- Prikaz znamenki je pouzdan i neosjetljiv na manje promjene napona.
- Broj od  $n$  znamenki u brojevnom sustavu s bazom 2:  
$$z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \dots + z_1 \cdot 2^1 + z_0 \cdot 2^0, \quad z_i \in \{0, 1\}$$

# Logic Threshold Voltage Levels



# Pretvorba dekadskog broja u binarni

$$N = z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \dots + z_1 \cdot 2^1 + z_0 \cdot 2^0$$

Izluči li se iz svih pribrojnika, osim posljednjeg, zajednički faktor 2:

$$N = 2 \cdot (z_{n-1} \cdot 2^{n-2} + z_{n-2} \cdot 2^{n-3} + \dots + z_1 \cdot 2^0) + z_0 \quad \Rightarrow$$

$$N = 2 \cdot q_1 + z_0$$

*može se zaključiti da je  $z_0$  ostatak dijeljenja  $N$  s 2*

**Pogledajmo sada količnik  $q_1$**

$$q_1 = z_{n-1} \cdot 2^{n-2} + z_{n-2} \cdot 2^{n-3} + \dots + z_1 \cdot 2^0$$

Izluči li se iz svih pribrojnika, osim posljednjeg, zajednički faktor 2:

$$q_1 = 2 \cdot (z_{n-1} \cdot 2^{n-3} + z_{n-2} \cdot 2^{n-4} + \dots) + z_1 \quad \Rightarrow$$

$$q_1 = 2 \cdot q_2 + z_1$$

*može se zaključiti da je  $z_1$  ostatak dijeljenja  $q_1$  s 2 itd .... sve dok se uzastopnim dijeljenjem s 2 ne postigne 0.*

Dakle, uzastopnim dijeljenjem cijelog broja  $N$  s 2 i zapisivanjem ostataka dijeljenja dobiju se znamenke ekvivalentnog binarnog broja.

# Pretvorba dekadskog broja u binarni

---

$$13_{10} = ?_2$$

$$N = 13 = 2 \cdot q_1 + z_0 \cdot 2^0 = 2 \cdot 6 + 1 \cdot 2^0$$

$$\Rightarrow z_0 = 1, q_1 = 6$$

$$q_1 = 6 = 2 \cdot q_2 + z_1 \cdot 2^0 = 2 \cdot 3 + 0 \cdot 2^0$$

$$\Rightarrow z_1 = 0, q_2 = 3$$

$$q_2 = 3 = 2 \cdot q_3 + z_2 \cdot 2^0 = 2 \cdot 1 + 1 \cdot 2^0$$

$$\Rightarrow z_2 = 1, q_3 = 1$$

$$q_3 = 1 = 2 \cdot q_4 + z_3 \cdot 2^0 = 2 \cdot 0 + 1 \cdot 2^0$$

$$\Rightarrow z_3 = 1, q_4 = 0$$

$$13_{10} = 1101_2$$

# Binarni brojevni sustav

- Općenito: najveći dekadski broj s  $d$  znamenaka iznosi  $10^d - 1$ 
  - Primjer:  $d=2$ , najveći broj  $99 = 10^2 - 1$
- Općenito: najveći **binarni** broj s  $b$  znamenaka iznosi  $2^b - 1$ 
  - Primjer:  $b=4$ , najveći broj  $1111 = 2^4 - 1$
- Koliko binarnih znamenaka treba za prikaz dekadskog broja?

$$10^d - 1 = 2^b - 1 \Rightarrow$$

$$10^d = 2^b \Rightarrow$$

$$d \cdot \log 10 = b \cdot \log 2 \Rightarrow$$

$$d = b \log 2 \Rightarrow$$

$$b = d : \log 2 \Rightarrow$$

$$b = d : 0,30102999566398119521373889472449 \Rightarrow$$

$$b \approx d : 0,3 \Rightarrow b \approx d \cdot (1 : 0,3) \Rightarrow b \approx d \cdot 3,33$$

$$\text{Primjer: ako treba prikazati broj } \leq 99, \quad b \approx 2 \cdot 3,33 = 6,66$$



# Binarni brojevni sustav

---

- Koliki se najveći broj može prikazati sa 6 binarnih znamenaka?
  - $111111_2 = 32+16+8+4+2+1 = 63 = 64-1 = 2^6-1$
- Koliki je najveći broj prikazan sa 7 binarnih znamenaka?
  - $1111111_2 = 64+32+16+8+4+2+1 = 127 = 128-1 = 2^7-1$
- Očito, broj znamenaka u prethodnim izrazima treba zaokružiti na viši cijeli broj tj:

$$b = \lceil d : \log 2 \rceil$$

$$b \approx \lceil d \cdot 3,33 \rceil$$

# Negativni binarni brojevi

- Budući da se u registar može pohraniti samo 0 ili 1, za pohranu negativnog predznaka je potreban dogovor (konvencija).
- Uobičajeno se negativni brojevi prikazuju tzv. tehnikom dvojnog komplementa, tj. nule pretvaramo u jedinice, jedinice u nule (komplement do baze - 1), a zatim se tom komplementu dodaje 1 (komplement do baze – dvojni komplement).
- Primjer: -37 u registru s 8 bita

|     |   |   |   |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|---|---|---|
| 37  |   | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|     |   | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
|     | + |   |   |   |   |   |   |   | 1 |
| -37 |   | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
|     | + |   |   |   |   |   |   |   |   |
| 37  |   | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|     | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Primjer svih sadržaja u registru od 3 bita (ako je prvi bit predznak)

---

U registru s  $n=3$  bita, ako je prvi bit predznak mogu se prikazati sljedeći brojevi:

| Dekadski broj | Binarni broj |
|---------------|--------------|
| 0             | 000          |
| 1             | 001          |
| 2             | 010          |
| 3             | 011          |
| -4            | 100          |
| -3            | 101          |
| -2            | 110          |
| -1            | 111          |

Za  $n=3$  dobije se interval  $[-2^2, 2^2 - 1]$ , općenito  $[-2^{n-1}, 2^{n-1} - 1]$

Za  $n=8$  dobije se interval  $[-2^7, 2^7 - 1]$ , tj.  $[-128, 127]$

# Zadatak

---

- Dekadski broj -11 prikazati u obliku binarnog broja, u tehnici dvojnog komplementa

?

# Zadatak

---

- Prethodni zadatak nije ispravno zadan jer nije navedeno pomoću koliko bitova treba prikazati zadani broj
- Ispravno zadani zadatak glasi:
  - Dekadski broj -11 prikazati u obliku binarnog broja, u tehnici dvojnog komplementa, u registru od 5 bitova

?

# Oktalni brojevni sustav

- Baza sustava je **B=8** a znamenke su **0,1,2,3,4,5,6,7**
  - Koristi se za skraćeno zapisivanje binarnih sadržaja kada je to spretno
  - Zapis se može dobiti iz dekadskog uzastopnim dijeljenjem s 8 i zapisivanjem ostataka s desna na lijevo, ali i izravno iz binarnog zapisa:

$$N = z_5 \cdot 2^5 + z_4 \cdot 2^4 + z_3 \cdot 2^3 + z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0$$

grupiramo li tri po tri pribrojnika i izlučimo zajednički faktor:

$$N = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0) \cdot 2^3 + (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0) \cdot 2^0$$

$$N = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0) \cdot 8^1 + (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0) \cdot 8^0$$

$\Rightarrow$

$$o_1 = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0), o_0 = (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0)$$

Primjer:

|                    |          |          |          |          |          |          |          |          |          |          |          |                       |
|--------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------------|
| 36-bitni broj      | 001      | 110      | 000      | 101      | 111      | 001      | 010      | 011      | 111      | 000      | 100      | 001 <sub>2</sub>      |
| oktalni ekvivalent | <b>1</b> | <b>6</b> | <b>0</b> | <b>5</b> | <b>7</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>7</b> | <b>0</b> | <b>4</b> | <b>1</b> <sub>8</sub> |

# Heksadekadski brojevni sustav

---

- Baza sustava je **B = 16**, a znamenke su **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**
  - Koristi se za skraćeno zapisivanje binarnog sadržaja.
  - Zapis se može dobiti iz dekadskog uzastopnim dijeljenjem sa 16 i zapisivanjem ostataka s desna na lijevo, ali i izravno iz binarnog zapisa

- Primjer:

|                          |          |          |          |          |               |
|--------------------------|----------|----------|----------|----------|---------------|
| 16-bitni broj            | 0111     | 1011     | 0011     | 1110     | <sub>2</sub>  |
| heksadekadski ekvivalent | <b>7</b> | <b>B</b> | <b>3</b> | <b>E</b> | <sub>16</sub> |

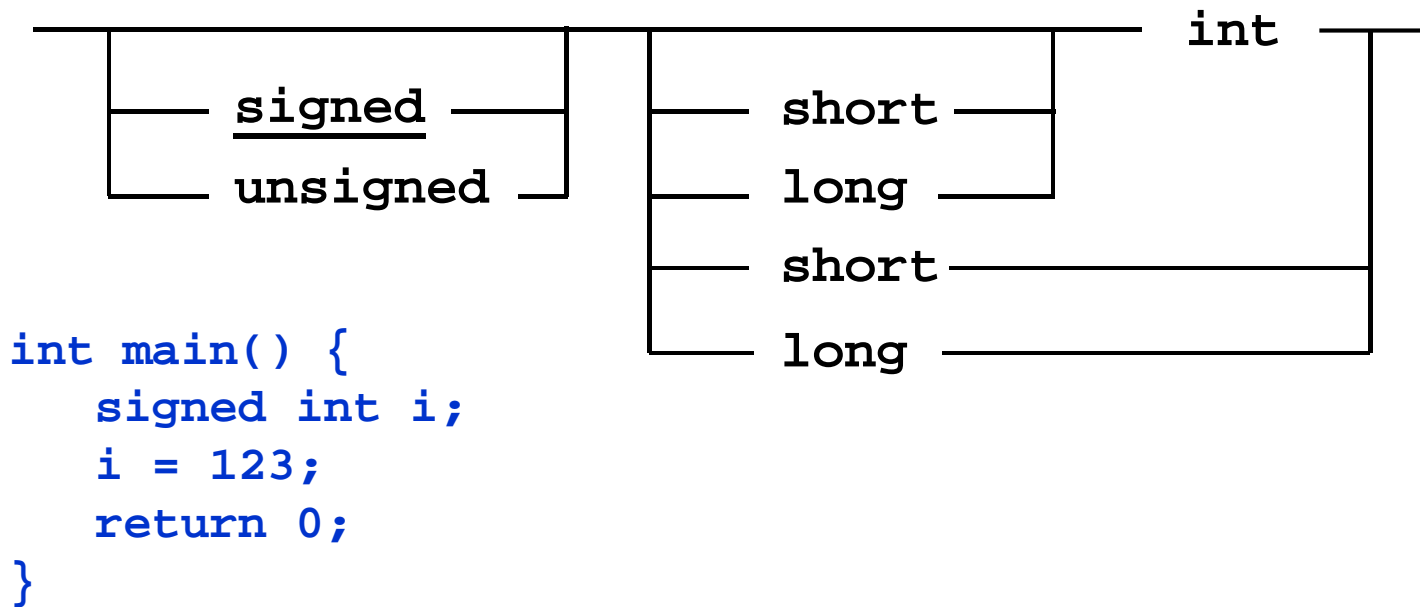
# Cjelobrojni tip podatka i prefiksi u jeziku C

---

- `int` - cjelobrojni tip
- Prefiksi (ili kvalifikatori) za cjelobrojni tip podatka
  - `short` - smanjuje raspon cjelobrojnih vrijednosti koje varijabla može sadržavati
  - `long` - povećava raspon cjelobrojnih vrijednosti koje varijabla može sadržavati
  - `unsigned` - dopušta pridruživanje samo pozitivnih vrijednosti
  - `signed` - dopušta pridruživanje pozitivnih i negativnih vrijednosti (zato dopušta manji raspon brojeva od *unsigned*)



# Formalna deklaracija cjelobrojnog tipa



`signed short int i; ⇔ signed short i; ⇔ short int i; ⇔ short i;`  
`signed int i; ⇔ int i;`  
`signed long int i; ⇔ signed long i; ⇔ long int i; ⇔ long i;`  
`unsigned short int i; ⇔ unsigned short i;`  
`unsigned int i;`  
`unsigned long int i; ⇔ unsigned long i;`

# Cjelobrojni tip podatka i preciznost

---

- Cjelobrojni tip podatka (`integer`), s obzirom na preciznost (broj binarnih znamenki), može se deklarirati kao `short` ili `long`.
- U programskom jeziku C preciznost `short` i `long` tipova nije propisana, ali vrijede sljedeća pravila:
  - `short` ne može biti precizniji od `int`
  - `int` ne može biti precizniji od `long`
  - odnosno za preciznost se može napisati:
$$\text{short} \leq \text{int} \leq \text{long}$$
- U cjelobrojni tip podatka također pripada: `char` (kada se koristi kao brojevena, a ne znakovna vrijednost).

# Cijeli broj s predznakom i bez predznaka

---

- Kvalifikator `signed` (s predznakom) označava da se u varijabli može pohraniti pozitivna i negativna vrijednost.
- Varijabla definirana kao `unsigned` (bez predznaka) može pohraniti samo pozitivne vrijednosti, što udvostručuje maksimalnu pozitivnu vrijednost koja u njoj može biti pohranjena u odnosu na `signed`.

## Primjer:

- Najveći pozitivni broj prikazan u 16-bitnom registru (`signed`):  
$$0111111111111111_2 = 32767_{10}$$
- Najveći pozitivni broj prikazan u 16-bitnom registru (`unsigned`):  
$$1111111111111111 = 65535_{10}$$

# Cijeli broj s predznakom i bez predznaka

---

## Primjer:

- Broj **5** prikazan u 16-bitnom registru:

00000000000000101

- Broj **−5** prikazan tehnikom dvojnog komplementa u 16-bitnom registru:

1111111111111011

- Isti niz binarnih znamenaka pohranjen u varijablu cjelobrojnog tipa **bez** predznaka, predstavlja broj 65531

# Cijeli broj s predznakom i bez predznaka

---

Primjer u programskom jeziku C:

- pretpostavka: `short int` koristi dva okteta

```
int main () {
 short int i;
 unsigned short int j;

 i = -5;
 printf ("%d\n", i);
 j = i;
 printf ("%d\n", j);
 return 0;
}
```

Pohrana: 11111111111111011

Ispis: -5

Pohrana: 11111111111111011

Ispis: 65531

# Zašto je to važno znati?

---

Primjer u programskom jeziku C:

- pretpostavka: `short int` koristi dva okteta

```
int main () {
 short int i;
 i = 31500;
 i = i + 1000;
 printf ("%d\n", i);
 i = i + 1000;
 printf ("%d\n", i);
 return 0;
}
```

- Što se i zašto dogodilo?

# Korištenje `char` tipa podatka za pohranu cijelog broja

---

```
#include <stdio.h>
```

```
int main () {
```

```
 char i;
```

```
 i = -5;
```

```
 printf ("%d\n", i);
```

```
 i = 12;
```

```
 printf ("%d\n", i);
```

```
 return 0;
```

```
}
```

Pohrana: **11111011**

Ispis: **-5**

Pohrana: **00001100**

Ispis: **12**

# Upamtite

---

Kod rješavanja zadatka, ako u zadatku nije drugačije navedeno, podrazumijevat će se:

- za tip podatka `char` koristi se jedan oktet
- za tip podatka `short int` koristi se dva okteta
- za tip podatka `int` koristi se četiri okteta
- za tip podatka `long` koristi se četiri okteta



# Cjelobrojne konstante u jeziku C

---

- Konstante pisane u dekadskoj notaciji:

`7 20 64 -110 8092 65535 34567821  
34567821L -176987941 (pripaziti, 1≠l)`

- Konstante pisane u oktalnoj notaciji:

`07 024 0100 0156 017634 0177777`

- Konstante pisane u heksadekadskoj notaciji

`0x7 0x14 0x40 0x6E 0x1F9C 0xFFFF  
0xFFFFFFFF`

# Primjer: Zadavanje cjelobrojnih konstanti

---

```
#include <stdio.h>
/* primjer zadavanja cjelobrojnih konstanti*/
int main () {
 int x, y, rez;
 /* oktalno zadan cijeli broj */
 x = 010;
 /* heksadekadski zadan cijeli broj */
 y = 0xF;
 rez = 10 + y / x;
 printf("Rezultat = %d\n", rez);
 return 0;
}
```

# Cjelobrojne konstante bez predznaka u C-u

---

Konstante bez predznaka pišu se s **U** ili **u** na kraju

- u dekadskoj notaciji:

**7U 20u 64u**

- u oktalnoj notaciji:

**07u 024U 0100u**

- u heksadekadskoj notaciji

**0x7u 0x14u 0xFFFFFFFFFu**

# Decimalni brojevi u binarnom sustavu

---

- Decimalni binarni brojevi sadrže "binarnu točku", analogno decimalnom zarezu, odnosno točki u anglo-američkoj notaciji.

- Primjer prikaza decimalnih brojeva u binarnom sustavu:

$$\begin{aligned} \mathbf{5.75}_{10} &= \mathbf{5} * 10^0 + \mathbf{7} * 10^{-1} + \mathbf{5} * 10^{-2} = \\ &= \mathbf{1} * 2^2 + \mathbf{0} * 2^1 + \mathbf{1} * 2^0 + \mathbf{1} * 2^{-1} + \mathbf{1} * 2^{-2} = \\ &= \mathbf{1\ 0\ 1\ .\ 1\ 1}_2 \end{aligned}$$

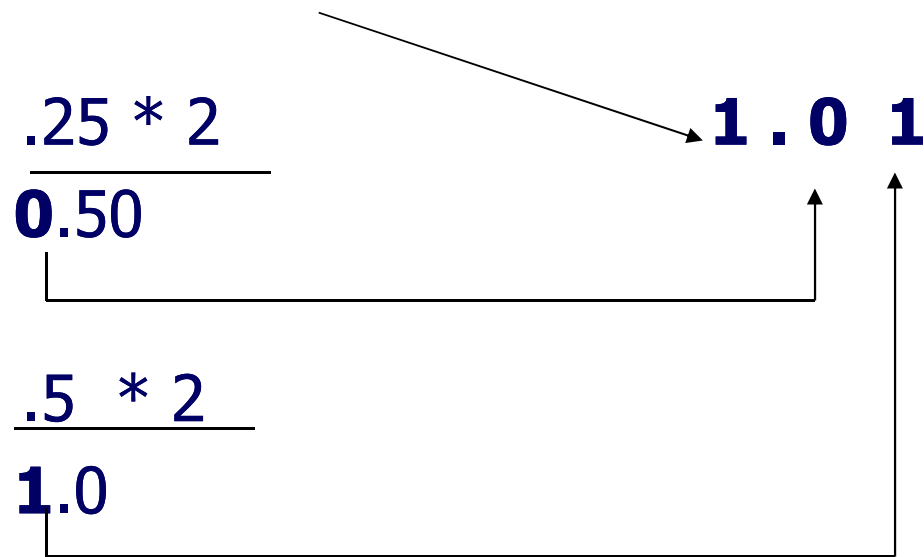
**Općenito:**

$$\begin{aligned} R = & z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \dots + z_1 \cdot 2^1 + z_0 \cdot 2^0 \\ & + z_{n+1} \cdot 2^{-1} + z_{n+2} \cdot 2^{-2} + \dots \end{aligned}$$

# Pretvaranje decimalnog broja iz dekadskog u binarni brojevni sustav

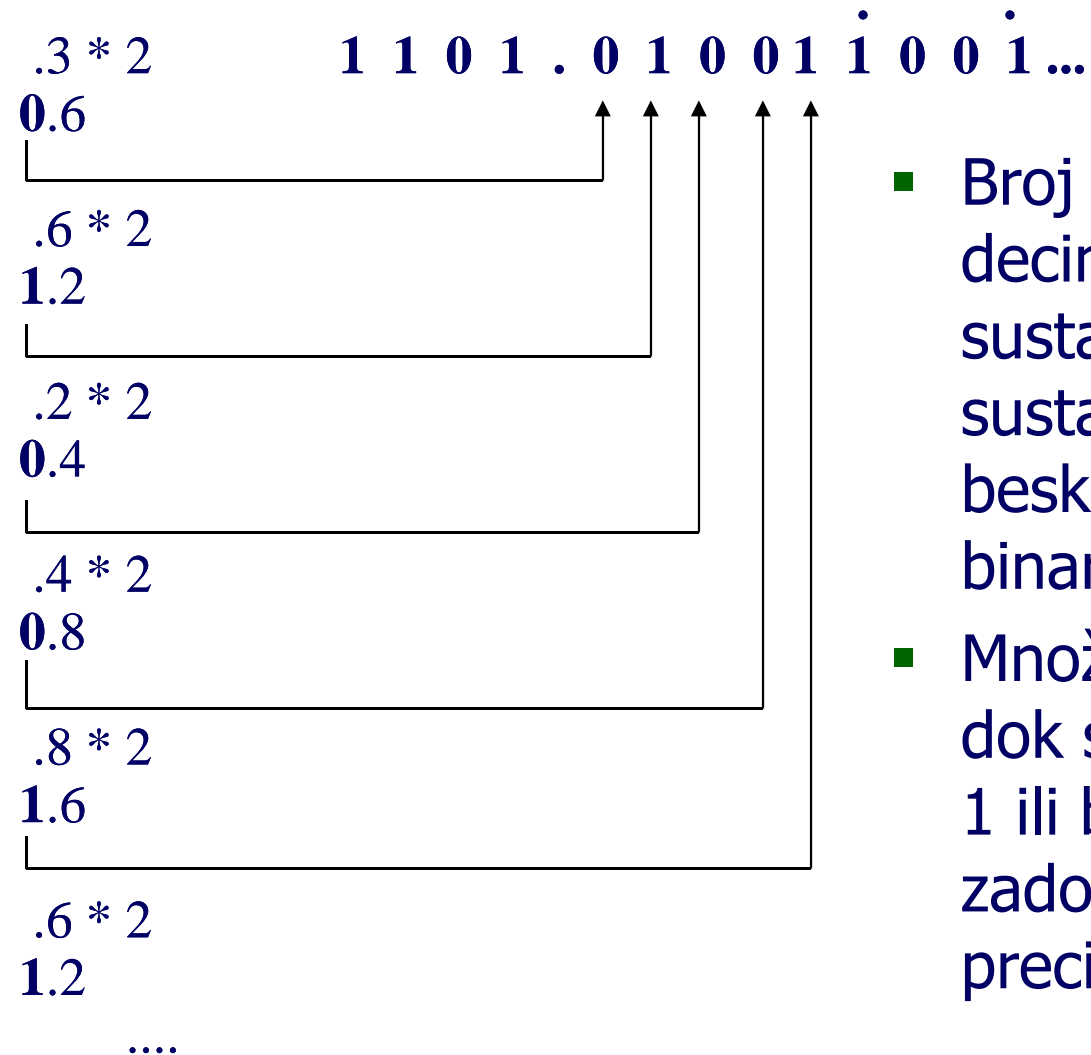
- Cjelobrojni dio dekadskog broja pretvara se u binarni uzastopnim dijeljenjem, a decimalni uzastopnim množenjem s 2, gdje cjelobrojni dio dobivenih produkata tvori decimalne znamenke binarnog broja.

$$1.25 = \mathbf{1} + \mathbf{.25}$$



# Pretvaranje decimalnog broja iz dekadskog u binarni brojevni sustav

$$13.3 = 13 + 0.3$$



- Broj s konačnim brojem decimala u dekadskom sustavu u binarnom sustavu može imati beskonačni periodički niz binarnih znamenaka.
- Množenje se nastavlja sve dok se ne dobije cijeli broj 1 ili bude dosegnuta zadovoljavajuća preciznost.

# Množenje binarnog broja s $2^n$ i $2^{-n}$

---

- 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.

- Na primjer:

$$\mathbf{1\ 1\ 1} * 2^2 = \mathbf{1\ 1\ 1\ 0\ 0}$$

$$\mathbf{1.\ 1\ 1} * 2^{-2} = \mathbf{0.\ 0\ 1\ 1\ 1}$$

- Kako, međutim, u registar pohraniti točku?

# Prikaz realnih brojeva u računalu

---

- Što je realni broj?
  - 7    2.5    -17.5    1000    0    871236173.8763723
  - $\frac{1}{3}$      $\pi$     e
  - 3.141592653589793238462643383279502884197169  $\neq \pi$
  - 0.3333333333333333333333333333333333333333333333333  $\neq \frac{1}{3}$
- ne postoji način na koji bi se u računalu mogao pohraniti svaki realni broj. Pohranjuju se približne vrijednosti realnih brojeva.



# Prikaz realnih brojeva u dekadskom obliku

- Kako inženjeri i znanstvenici prikazuju vrlo velike i vrlo male brojeve?
- Znanstvena notacija: decimalni broj s jednom znamenkom ispred decimalne točke (zareza), pomnožen odgovarajućom potencijom broja 10 (baza brojanja = 10).

- Kolika je prosječna udaljenost Neptuna i Sunca?

- 4503930000000 m

- $4.50393 \cdot 10^{12}$  m

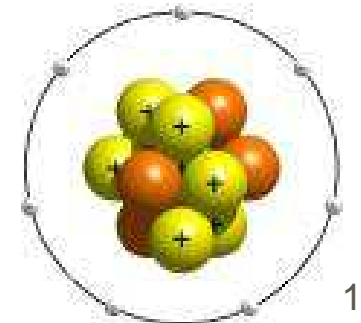
**mantisa**

**eksponent**

- Koliko iznosi masa elektrona?

- 0.0000000000000000000000000000000000910938188 kg

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



# Prikaz realnih brojeva u binarnom obliku

- Slično, realan broj u binarnom obliku, može se prikazati kao binarni decimalni broj s jednom binarnom znamenkom ispred binarne točke, pomnožen odgovarajućom potencijom broja 2 (baza brojanja = 2). Za broj u takvom obliku kaže se da je **normaliziran**.
- Na primjer:  
$$101.11 = \underbrace{1.0111}_{\text{mantisa}} \cdot 2^2$$

binarni eksponent

$$0.000000000000000010011 = 1.0011 \cdot 2^{-14}$$
- Normalizacija omogućava prikaz vrlo velikih i vrlo malih brojeva, bez korištenja velikog broja nula.

# Pohrana realnih brojeva u računalu

---

- Realni brojevi u računalu se pohranjuju u normaliziranom obliku.
- Kako normalizirani broj pohraniti u registar?
- IEEE (Institute of Electrical and Electronics Engineers) standard 754 definira način pohrane realnih brojeva

# Realni tip podatka u jeziku C

- Primjer definicije varijable:

```
float x;
```

- Primjer realnih konstanti:

```
2.f 2.34F -1.34e5f
```

- Primjer C programa:

```
int main () {
 float x;
 x = 2.f * 5.0f * 3.14159F;
 printf("Krug radijusa 5 ima opseg %f", x);
 return 0;
}
```

- Pored tipa `float`, u jeziku C postoje tipovi `double` i `long double`.

# Realni brojevi jednostruke preciznosti

---

- Najčešće prikazuje `float` tip podatka u jeziku C
- Koriste se 4 okteta (32 bita)
- Realni broj se pohranjuje u obliku:



- 1 bit za pohranu predznaka
- 8 bitova za pohranu karakteristike
- 23 bita za pohranu mantise bez skrivenog bita

# Realni brojevi jednostruke preciznosti



- P je oznaka za predznak
  - $P = 1$ : negativan broj
  - $P = 0$ : pozitivan broj
- BE je oznaka za binarni eksponent normaliziranog broja
  - raspon binarnog eksponenta  $BE \in [-126, 127]$
- karakteristika K
  - $K = BE + 127$
  - raspon karakteristike:  $K \in [0, 255]$
  - $K = 0$  i  $K = 255$  se koriste za posebne slučajeve (objašnjeno kasnije)
- mantisa M
  - u registar se ne pohranjuje cijela mantisa M, već mantisa iz koje je uklonjen skriveni bit

# Skriveni bit mantise

- U binarnom brojevnom sustavu, jedina znamenka koja se u normaliziranom broju (osim za broj 0) može pojaviti ispred binarne točke je 1

1.xxxxxxx

u registru od 8 bitova,  
moguće je prikazati broj s  
ukupno 8 znamenaka

- Ta jedinica se ne pohranjuje i zato se naziva ***skrivenim bitom (hidden bit)***. Na taj se način štedi jedan bit, a time povećava preciznost.

1.xxxxxxxxx

u registru od 8 bitova,  
moguće je prikazati broj s  
ukupno 9 znamenaka

1 ispred točke se podrazumijeva,  
stoga ga ne treba pohraniti

# Primjer: Prikazati broj 5.75 kao realni broj

1. Realni dekadski broj prikazati u obliku realnog binarnog broja

$$5.75_{10} = 101.11_2$$

2. Odrediti predznak: broj je pozitivan, stoga je  $P = 0$

3. Normalizirati binarni broj

$$101.11_2 \cdot 2^0 = 1.0111_2 \cdot 2^2$$

M
BE

4. Izračunati karakteristiku i izraziti ju u binarnom obliku

$$K = 2_{10} + 127_{10} = 129_{10} = 1000\ 0001_2$$

5. Izbaciti vodeću jedinicu iz mantise (skriveni bit)

$$M \text{ (bez skrivenog bita i binarne točke)} = 0111_2$$

6. *Prepisati* predznak, karakteristiku i mantisu bez skrivenog bita u registar

|   |          |                          |
|---|----------|--------------------------|
| 0 | 10000001 | 011100000000000000000000 |
|---|----------|--------------------------|

**P   Karakteristika   Mantisa bez skrivenog bita**

0100 0000 1011 1000 0000 0000 0000 0000<sub>2</sub>

**4   0   B   8   0   0   0   0**<sub>16</sub>



# Primjeri pohrane realnih brojeva

---

$$\mathbf{2} = 10_2 * 2^0 = 1_2 * 2^1 = 0100\ 0000\ 0000\ 0000 \dots 0000\ 0000 = 4000\ 0000_{16}$$

$P = 0, K = 1 + 127 = 128 \quad (10000000), M = (1.)\ 000\ 0000 \dots 0000\ 0000$

$$\mathbf{-2} = -10_2 * 2^0 = -1_2 * 2^1 = 1100\ 0000\ 0000\ 0000 \dots 0000\ 0000 = C000\ 0000_{16}$$

Jednako kao 2, ali  $P = 1$

$$\mathbf{4} = 100_2 * 2^0 = 1_2 * 2^2 = 0100\ 0000\ 1000\ 0000 \dots 0000\ 0000 = 4080\ 0000_{16}$$

Jednaka mantisa,  $BE = 2, K = 2 + 127 = 129 \quad (10000001)$

$$\mathbf{6} = 110_2 * 2^0 = 1.1_2 * 2^2 = 0100\ 0000\ 1100\ 0000 \dots 0000\ 0000 = 40C0\ 0000_{16}$$

$$\mathbf{1} = 1_2 * 2^0 = 0011\ 1111\ 1000\ 0000 \dots 0000\ 0000 = 3F80\ 0000_{16}$$

$K = 0 + 127 \quad (01111111).$

$$\mathbf{.75} = 0.11_2 * 2^0 = 1.1_2 * 2^{-1} = 0011\ 1111\ 0100\ 0000 \dots 0000\ 0000 = 3F40\ 0000_{16}$$

# Kalkulator za vježbanje

---

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

<http://babbage.cs.qc.edu/courses/cs341/IEEE-754.html>

# Posebni slućajevi: prikaz broja 0

---

- Kada bi vodeća znamenka normaliziranog broja uvijek bila 1, ne bi bilo moguće prikazati broj 0
- Koristi se sljedeći dogovor: kada je  $K=0$  i svi bitovi mantise postavljeni na 0, radi se o prikazu realnog broja 0
- U računalu se mogu prikazati brojevi "+0" i "-0"  
 $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \rightarrow +0$   
 $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \rightarrow -0$
- Međutim, pri usporedbi tih dviju vrijednosti, smatra se da su jednake.

# Posebni slučajevi: denormalizirani broj

---

- Kada je  $K=0$  i postoje bitovi mantise koji nisu 0, radi se o "denormaliziranom broju". Ne podrazumijeva se skriveni bit, te se smatra da je vodeći bit mantise 0. Vrijednost eksponenta je fiksirana na -126 (ne koristi se izraz  $K=\text{binarni eksponent}+127$ ).

0000 0000 0110 0000 0000 0000 0000 0000

$$\rightarrow 0.11 \cdot 2^{-126}$$

0000 0000 0000 0000 0000 0000 0000 1101

$$\rightarrow 0.000\ 0000\ 0000\ 0000\ 0000\ 1101 \cdot 2^{-126}$$

## Posebni slućajevi: prikaz $+\infty$ i $-\infty$

---

- Kada je  $K=255$  i svi bitovi mantise su postavljeni na 0, radi se o prikazu  $+\infty$  ili  $-\infty$ .
- Takvi brojevi se dobiju npr. prilikom dijeljenja s nulom:

```
float x, y, z, w;
x = 5.f;
y = -5.f;
z = x / 0.f;
w = y / 0.f;
```

0111 1111 1000 0000 0000 0000 0000 0000  $\rightarrow +\infty$

1111 1111 1000 0000 0000 0000 0000 0000  $\rightarrow -\infty$

# Posebni slučajeви: prikaz NaN

---

- Ako je K=255 i postoje bitovi mantise koji nisu 0, radi se o NaN (*not a number*), tj. ne radi se o prikazu broja. NaN je posljedica obavljanja operacije čiji je rezultat nedefiniran ili se prilikom obavljanja operacije dogodila pogreška, npr.

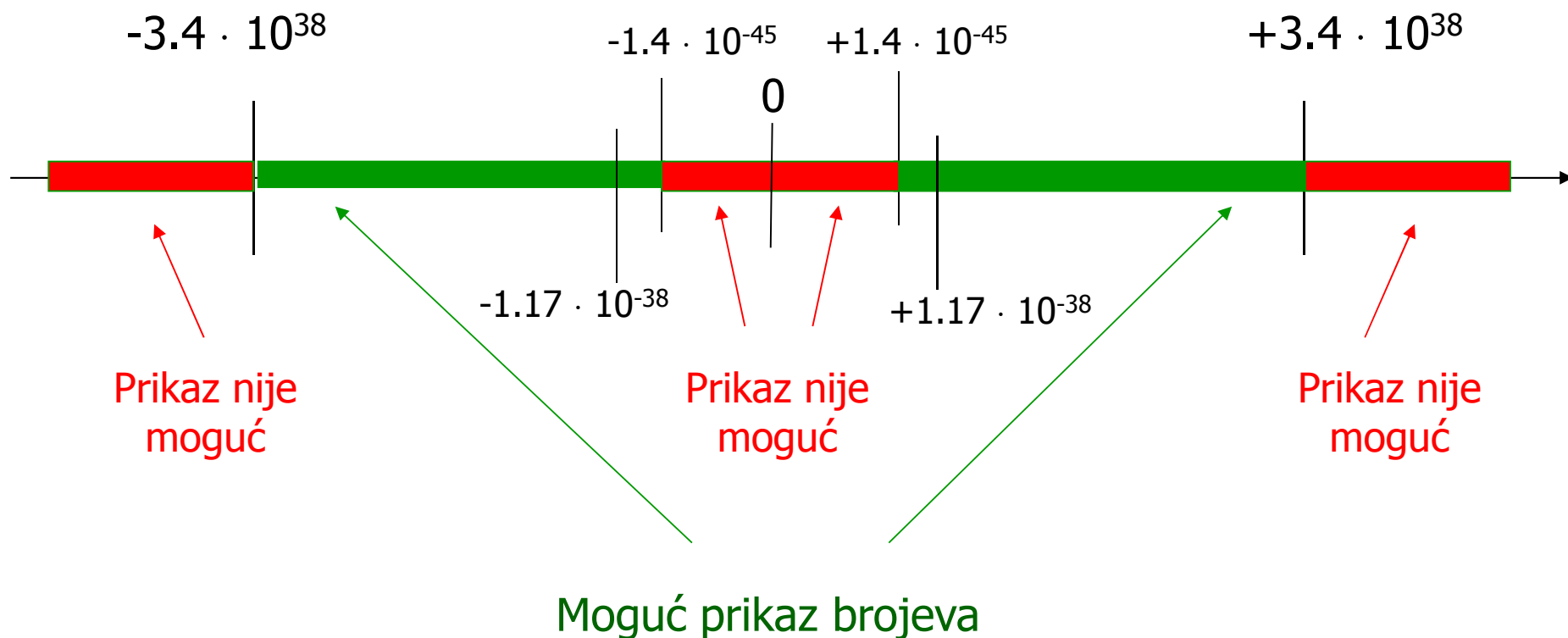
```
float x, y, z;
x = 0.f;
y = 0.f;
z = x / y;
```

0111 1111 1100 0000 0000 0000 0000 0000 → NaN



# Raspon realnih brojeva

(za format IEEE 754 - jednostruka 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, ali
  - za 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  $[-1.0, 1.0]$  ?
  - beskonačno mnogo, jer  $|[-1.0, 1.0]| \equiv |\mathbf{R}|$
- Samo konačan podskup realnih brojeva iz nekog intervala  $[-a, a]$  moguće je **točno** (bez pogreške) prikazati u registru. Ostali realni brojevi iz tog intervala mogu se pohraniti samo kao njihove približne vrijednosti

# Prikaz realnih brojeva u računalu

- [illegible]

# Koliko se različitih realnih brojeva može prikazati (za format IEEE 754 - jednostruka preciznost)

---

- za svaki  $K \in [0, 254]$ 
  - moguće su  $2^{23}$  različite mantise
  - moguća su dva predznaka
  - ukupno  $255 * 2^{23} * 2 = 4,278,190,080$  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  $[-3.4 \cdot 10^{38}, -1.4 \cdot 10^{-45}] \cup [1.4 \cdot 10^{-45}, 3.4 \cdot 10^{38}]$
- 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 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)
  - veličina pogreške pri prikazu broja je manja

# Pogreške pri prikazu realnih brojeva

---

- $x$  - realni broj kojeg treba pohraniti u registar
- $x^*$  - približna vrijednost broja  $x$  koja je zaista pohranjena u registar
- Apsolutna pogreška  $\alpha$

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

- Relativna pogreška  $\rho$

$$\rho = \alpha / x$$

- Primjer: ako je u registru umjesto potrebne vrijednosti  $x = 1.57$  pohranjena vrijednost  $x^* = 1.625$

$$\alpha = 1.625 - 1.57 = 0.055$$

$$\rho = 0.055 / 1.57 = 0.035$$

# Pogreške pri prikazu realnih brojeva

---

- Najveća moguća relativna pogreška ovisi o broju bitova mantise. Za IEEE 754 jednostruke 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 jednostruke preciznosti:

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

# Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - jednostruka preciznost)

---

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

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

```
float f1;
f1 = 332.3452f;
```

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

```
printf("%19.15f", f1); → 332.345214843750000
```

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



# Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - jednostruka preciznost)

---

Najveća apsolutna pogreška koja se uz jednostruku 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;
f2 = 0.7f;
```

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

```
printf("%17.15f", f2); → 0.699999988079071
```

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

# Numeričke pogreške

- Neki dekadski brojevi se ne mogu prikazati pomoću konačnog broja binarnih znamenaka. Primjer:

```
float f = 0.3f;
printf("%18.16f ", f); → 0.3000000119209290
```

- Za prikaz nekih realnih brojeva potrebno je "previše" binarnih znamenaka. Primjer:

[illegible]

# Numeričke pogreške

---

- Računanje s brojevima bitno različitog reda veličine može dovesti do numeričke pogreške
- **Primjer (uz jednostruku preciznost):**

$$1000000.0_{10} : (1.) 111010000100100000000000 * 2^{19}$$

$$0.015625_{10} : (1.) 000000000000000000000000 * 2^{-6}$$

Kod zbrajanja, binarne točke moraju biti poravnate:

$$1.111010000100100000000000 * 2^{19}$$

$$+ 0.000000000000000000000000 * 2^{19}$$

---

$$= 1.111010000100100000000000 * 2^{19} = 1000000.0_{10}$$

# Numeričke pogreške

---

Računanje s brojevima bitno različitog reda veličine

- primjer u programskom jeziku C:

```
float f = 6000000.0f;
float malif = 0.25f;
f = f + malif;
f = f + malif;
f = f + malif;
f = f + malif;
printf("%f ", f);
```

- očekuje se ispis

6000001.000000

- međutim, ispisat će se:

6000000.000000

# Realni brojevi dvostruke preciznosti

- Najčešće prikazuje `double` tip podatka u jeziku C
- Koristi se 8 okteta (64 bita)
- Realni broj se pohranjuje u obliku



- P je predznak ( $P = 1$ : negativan broj;  $P = 0$ : pozitivan broj)
- $K = BE + 1023$  (11 bita)  
Raspon karakteristike:  $K \in [0, 2047]$ .  
Raspon binarnog eksponenta  $BE \in [-1022, 1023]$
- Mantisa (52+1 bit).

# Realni brojevi dvostruke preciznosti

---

## ■ Posebni slučajevi

- Kada je  $K = 0$  i svi bitovi mantise su nula, radi se o broju nula
- Kada je  $K = 0$  i postoje bitovi mantise koji nisu 0, tada se radi o denormaliziranom broju
- Kada je  $K = 2047$  i svi bitovi mantise su 0, radi se o  $+\infty$  ili  $-\infty$
- Kada je  $K = 2047$  i postoje bitovi mantise koji nisu 0, tada se radi o prikazu broja (NaN)

# Raspon realnih brojeva

## (za format IEEE 754 - dvostruka preciznost)

---

- Najmanji pozitivni broj koji se može prikazati je:

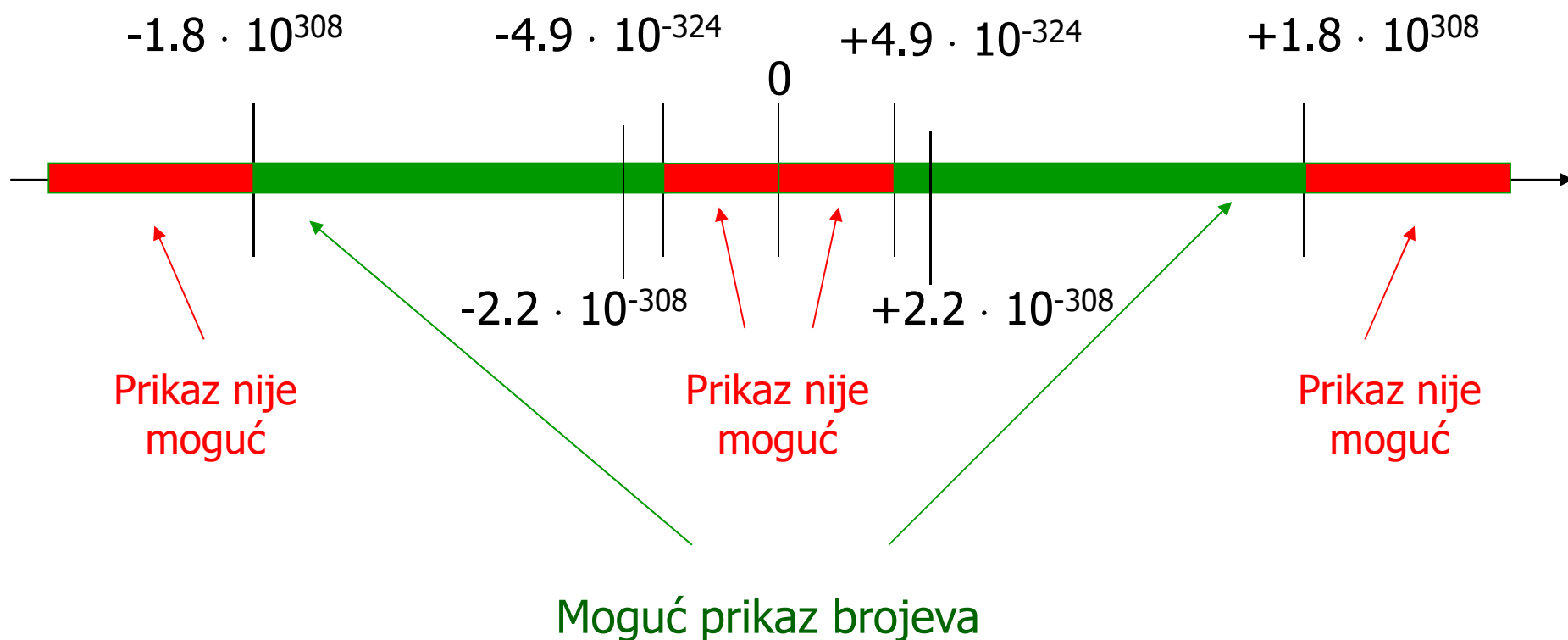
$$0.00\dots001_2 \cdot 2^{-1022} \\ \approx 4.9 * 10^{-324}$$

- Najveći pozitivni broj koji se može prikazati je:

$$1.11\dots11_2 \cdot 2^{1023} \approx 2^{1024} \\ \approx 1.8 * 10^{308}$$

# Raspon realnih brojeva

(za format IEEE 754 - dvostruka preciznost)





# Pogreške pri prikazu realnih brojeva (za format IEEE 754 - dvostruka preciznost)

---

- Najveća moguća relativna pogreška ovisi o broju bitova mantise. Za IEEE 754 dvostruke preciznosti:

$$|\rho| \leq 2^{-53} \approx 1.1 \cdot 10^{-16}$$

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

$$|\alpha| \leq 2^{-53} \cdot |x| \approx 1.1 \cdot 10^{-16} \cdot |x|$$

# Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - dvostruka preciznost)

---

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

$$|\alpha| \leq 1.1 \cdot 10^{-16} \cdot 0.7 \approx 7.7 \cdot 10^{-17}$$

```
double f4;
f4 = 0.7;
```

očekuje se da će biti pohranjen broj  $0.7 \pm 7.7 \cdot 10^{-17}$

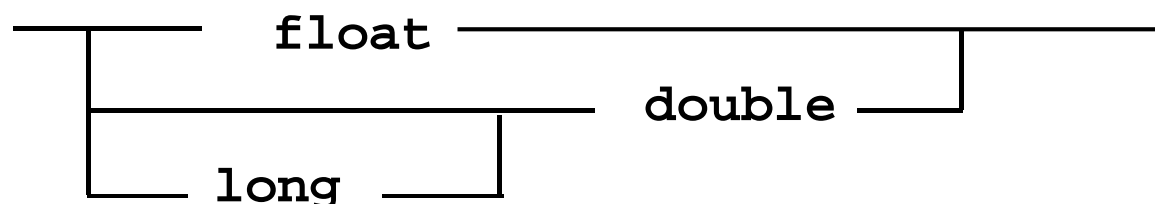
```
printf("%19.17f", f4); → 0.699999999999999996
```

Zaista, apsolutna pogreška je  $-4.0 \cdot 10^{-17}$ , što je po apsolutnoj vrijednosti manje od  $7.7 \cdot 10^{-17}$ .

# Realni tip podatka u jeziku C

- Primjer definicije varijabli u programskom jeziku C:

```
float x;
double y;
long double z;
```



- Programski jezik C ne propisuje preciznost tipova `float`, `double` i `long double`, ali vrijedi da *float* ne može biti precizniji od *double*, te *double* ne može biti precizniji od *long double*:

`float`  $\leq$  `double`  $\leq$  `long double`

# Tip long double

---

- Prostor za pohranu ovisi o platformi, pa se tako mogu naći implementacije u kojima je njegova veličina 64, 80, 96 ili 128 bita.
- ANSI standard ne propisuje preciznost, ali zahtijeva da nije manje precizan od **double** tipa.
- Primjer raspodjele ako je njegova veličina 80 bita:

Karakteristika: 15 bita

Binarni eksponent: Karakteristika – 16383

# Realne konstante

---

- Primjer realnih konstanti za različite tipove realnih brojeva:

|     |       |           |           |             |
|-----|-------|-----------|-----------|-------------|
| 2.f | 2.34F | -1.34e5f  |           | float       |
| 1.  | 2.34  | 9e-8      | 8.345e+25 | double      |
| 1.L | 2.34L | -2.5e-37L |           | long double |

# Formatske specifikacije za scanf i printf

---

```
#include <stdio.h>
int main () {
 float x;
 double y;
 scanf ("%f %lf", &x, &y);
 printf ("%f %f\n", x, y);
 return 0;
}
```

# Upamtite

---

Kod rješavanja zadatka, ako u zadatku nije drugačije navedeno, podrazumijevat će se:

- za tip podatka `float` koristi se četiri okteta
- za tip podatka `double` koristi se osam okteta

# Veličina osnovnih tipova podataka

- Veličina osnovnih tipova podataka, odnosno količina memorije koju zauzima jedna varijabla osnovnog tipa ovisi o konkretnoj implementaciji prevodioca.

| Tip                                           | Veličina              |
|-----------------------------------------------|-----------------------|
| <code>char, unsigned char, signed char</code> | 1 oktet               |
| <code>short, unsigned short</code>            | 2 okteta              |
| <code>int, unsigned int</code>                | 4 okteta              |
| <code>long, unsigned long</code>              | 4 okteta              |
| <code>float</code>                            | 4 okteta              |
| <code>double</code>                           | 8 okteta              |
| <code>long double</code>                      | 8 (10, 12, 16) okteta |



# Osobitosti C prevodilaca

---

- Veličina prostora za pohranu podatka određenog tipa nije propisana standardom. Stoga je programeru na raspolaganju operator `sizeof`, koji za zadani operand izračunava veličinu prostora za pohranu izraženu u oktetima.
- Operand može biti naziv tipa podatka ili naziv varijable
- Primjer:

```
int var;
printf("%d", sizeof(char)); /* ispisuje 1 */
printf("%d", sizeof(var)); /* ispisuje 4 */
```

# Rasponi različitih tipova cijelih brojeva

---

| Deklaracija               | Broj bita | Interval brojeva          |
|---------------------------|-----------|---------------------------|
| <b>int</b>                | 32        | [-2147483648, 2147483647] |
| <b>signed int</b>         | 32        | [-2147483648, 2147483647] |
| <b>short</b>              | 16        | [-32768, 32767]           |
| <b>short int</b>          | 16        | [-32768, 32767]           |
| <b>signed short int</b>   | 16        | [-32768, 32767]           |
| <b>unsigned int</b>       | 32        | [0U, 4294967295U]         |
| <b>unsigned short int</b> | 16        | [0U, 65535U]              |
| <b>long</b>               | 32        | [-2147483648, 2147483647] |
| <b>long int</b>           | 32        | [-2147483648, 2147483647] |
| <b>signed long int</b>    | 32        | [-2147483648, 2147483647] |
| <b>unsigned long int</b>  | 32        | [0U, 4294967295U]         |

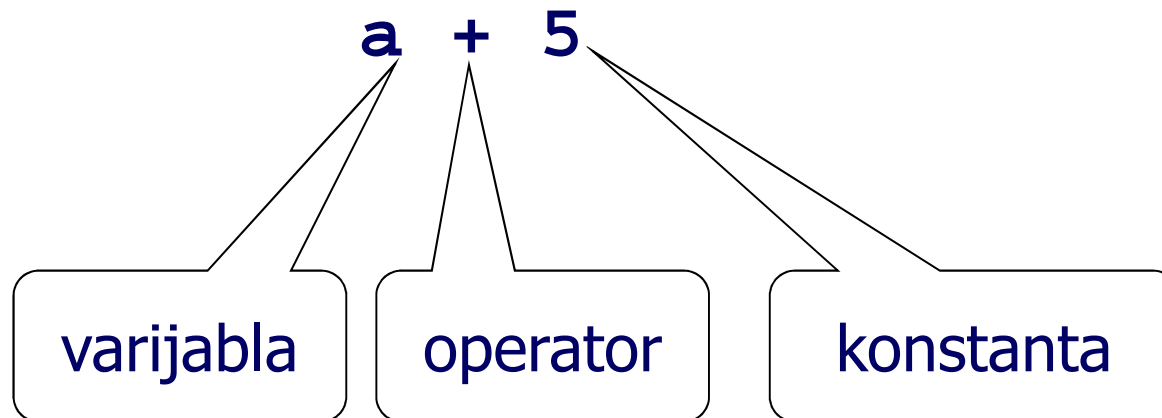
---

# Operatori i izrazi

# Izrazi

- Izraz jest kombinacija operatora, operanada (konstante, varijable, ...) i zagrada, koja po evaluaciji daje rezultat. Može biti dio većeg izraza.

- Primjer:



- Primjer: **(b + c) / ((d + e) \* 4)**

# Pridruživanje vrijednosti varijablama

- Pridruživanje vrijednosti: `varijabla = izraz;`
  - simbol u pseudokodu `:=`
  - u C-u `=`

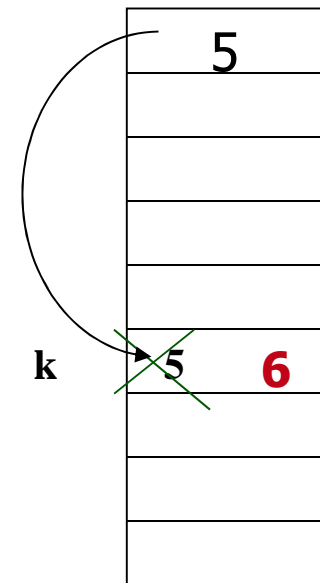
- Na primjer:

- `k := 5`  $\Rightarrow$  `k = 5;`

aritmetička naredba

- Što znači naredba ? `k = k + 1;`

aritmetički izraz



**Primjer pridruživanja:** Zadano je  $X=14.5$  i  $Y= -9.9$  . U programu treba ispisati vrijednosti  $X$  i  $Y$ , a zatim u  $X$  staviti vrijednost od  $Y$ , a u  $Y$  staviti vrijednost od  $X$ . Ispisati ponovno  $X$  i  $Y$ .

📁ZamjenaVarijabli

```
int main () {
 float x, y, p;
 x= 14.5f;
 y= -9.9f;
 printf ("x=%f, y=%f\n", x, y);
 /* zamijeniti vrijednosti x i y */
 p=x;
 x=y;
 y=p;
 printf ("Nakon zamjene: x=%f, y=%f\n", x, y);
 return 0;
}
```

| x    | y    | p    |
|------|------|------|
| ???  | ???  | ???  |
| 14.5 | ???  | ???  |
| 14.5 | -9.9 | ???  |
| 14.5 | -9.9 | 14.5 |
| -9.9 | -9.9 | 14.5 |
| -9.9 | 14.5 | 14.5 |

$x=14.500000$ ,  $y=-9.900000$

Nakon zamjene:  $x=-9.900000$ ,  $y=14.500000$

# Osnovni aritmetički operatori

---

## Operator Značenje

|   |                                                          |
|---|----------------------------------------------------------|
| + | zbrajanje                                                |
| - | oduzimanje                                               |
| * | množenje                                                 |
| / | dijeljenje                                               |
| % | ostatak kod cjelobrojnog dijeljenja<br>(modulo, modulus) |

# Djelovanje aritmetičkih operatora na cjelobrojne operande

---

```
int a, b;
a = 10; b = 3;
```

| <u>Izraz</u> | <u>Rezultat</u> |
|--------------|-----------------|
| a + b        | 13              |
| a - b        | 7               |
| a * b        | 30              |
| a / b        | 3               |
| a % b        | 1               |



# Djelovanje aritmetičkih operatora na realne operande

---

```
float a, b;
a = 12.5f; b = 2.f;
```

| <u>Izraz</u> | <u>Rezultat</u> |
|--------------|-----------------|
| a + b        | 14.5            |
| a - b        | 10.5            |
| a * b        | 25.0            |
| a / b        | 6.25            |
| a % b        | pogreška        |
| a može li:   | 12. % 5         |

# Aritmetika s različitim tipovima operanada

---

```
int i;
float f;
i = 2;
f = 2.99f;
```

Što je rezultat operacije  $i + f$

4      ili      4.99

Kada su operandi različitog tipa, prije obavljanja operacije obavlja se implicitna (automatska) pretvorba tipa rezultata u "veći (važniji)" od tipova operanada koji sudjeluju u operaciji.

U prikazanom primjeru, prije nego se obavi operacija zbrajanja, vrijednost koja se nalazi u varijabli `i` pretvara se u 2.0 (pri tome sadržaj varijable `i` ostaje nepromijenjen).

# Implicitna (automatska) pretvorba tipova podataka

---

Pretvorba tipova podataka u izrazima obavlja se prema jednom od sljedećih 5 pravila. Treba iskoristiti prvo po redu pravilo koje se može primijeniti na konkretan slučaj!

1. Ako je jedan od operandata tipa **long double**, preostali operand se pretvara u tip **long double**
2. Ako je jedan od operandata tipa **double**, preostali operand se pretvara u tip **double**
3. Ako je jedan od operandata tipa **float**, preostali operand se pretvara u tip **float**
4. Ako je jedan od operandata tipa **long**, preostali operand se pretvara u tip **long**
5. Operande tipa **short** i **char** pretvoriti u tip **int**

Kada se u izrazima pojavljuju *unsigned* tipovi, pravila pretvorbe su složenija i ovise o implementaciji. Zato se ovdje neće razmatrati.

# Primjeri: implicitna pretvorba tipova podataka

```
char c; int i; float f; long double ld;
short s; long li; double d;
```

| Operacija           | Pretvorba tipa prije obavljanja operacije                          | Prema pravilu |
|---------------------|--------------------------------------------------------------------|---------------|
| <code>ld * i</code> | sadržaj od <code>i</code> → long double                            | 1.            |
| <code>c % i</code>  | sadržaj od <code>c</code> → int                                    | 5.            |
| <code>s / f</code>  | sadržaj od <code>s</code> → float                                  | 3.            |
| <code>f - ld</code> | sadržaj od <code>f</code> → long double                            | 1.            |
| <code>li % i</code> | sadržaj od <code>i</code> → long                                   | 4.            |
| <code>i * f</code>  | sadržaj od <code>i</code> → float                                  | 3.            |
| <code>d + c</code>  | sadržaj od <code>c</code> → double                                 | 2.            |
| <code>s - c</code>  | sadržaj od <code>s</code> → int    sadržaj od <code>c</code> → int | 5.            |

## Primjeri: implicitna pretvorba tipova podataka

```
float f1, f2;
int i1, i2; /* pretpostavka int: 4 okteta */
char c1, c2;
f1 = 32000.5f; f2 = 1.0f;
i1 = 2147483647; i2 = 1;
c1 = 127; c2 = 1;
```

| <i>Izraz</i> | <i>Rezultat</i> | <i>tip rezultata</i> |
|--------------|-----------------|----------------------|
| f1 + f2      | 32001.5         | float                |
| f1 + i2      | 32001.5         | float                |
| i1 + i2      | -2147483648     | int                  |
| i1 + f2      | 2147483648.0    | float                |
| c1 + c2      | 128             | int                  |

# Pretvorba tipova kod pridruživanja

---

- Vrijednost s desne strane pretvara se u tip podatka varijable ili izraza s lijeve strane znaka za pridruživanje

- Primjer:

```
int i;
float f;
i = 2.75; /* 2.75 se pretvara u int */
f = 2147483638;
```

- Potrebno je obratiti pozornost da se pri promjeni tipa podatka može "izgubiti" manji ili veći dio informacije. U gornjem primjeru:
  - "izgubljene" su decimale 0.75 kod prvog pridruživanja, tj. ispisom vrijednosti varijable `i` dobilo bi se 2
  - `f` sadrži vrijednost za 10 veću od one koja je pridružena, tj. ispisom vrijednosti varijable `f` dobilo bi se 2147483648.

# Što kada varijabli pridružimo "prevelik" broj

---

```
char c1, c2, c3;
unsigned char c4;
float f1, f2;
c1 = 80; c2 = 150; c3 = 300; c4 = 150;
f1 = 332.345282347f; ← "previše binarnih znamenaka"
f2 = 1.0e40f; ← izvan dopuštenog raspona
printf("%d\n", c1); → 80
printf("%d\n", c2); → -106
printf("%d\n", c3); → 44
printf("%d\n", c4); → 150
printf("%20.15f\n", f1); → 332.345275878906250
printf("%f\n", f2); → 1.#INF00
```

# Eksplicitna (zadana) pretvorba tipa podataka

---

- Opći oblik zadane (eksplicitne) pretvorbe (eng. *cast operator*) glasi:

`(tip_podatka) operand`

- Operand može biti varijabla, konstanta ili izraz.
- Zadana pretvorba podataka ima viši prioritet od automatske.

- Primjer:

```
int i = 2000000000;
```

```
double d1, d2;
```

```
d1 = 2 * i;
```

→ -294967296.000000

```
d2 = 2 * (double)i;
```

→ 4000000000.000000



# Cjelobrojno dijeljenje

- Potrebno je obratiti pozornost na "neželjene" rezultate kod cjelobrojnog dijeljenja. Ako se na primjer u realnu varijablu *a* želi pridružiti vrijednost  $\frac{1}{2}$ , sljedeća naredba pridruživanja **neće** varijabli *a* pridružiti vrijednost 0.5:

`a = 1 / 2;`

- U izrazu s desne strane jednakosti **oba su operanda cjelobrojnog tipa**, pa će se obaviti cjelobrojno dijeljenje. Rezultat tog izraza je 0 (uz ostatak 1).
- Za izbjegavanje cjelobrojnog dijeljenja potrebno je koristiti zadanu pretvorbu tipa (dovoljno samo na jednom operandu) ili zadati konstante tako da je barem jedna realna:

`a = (float) 1 / 2; ili`

`a = 1. / 2; ili`

`a = 1 / 2.;`

Napomena: U prvom slučaju korištena je zadana pretvorba tipa operanda (mogla se uporabiti i nad drugim operandom).

U drugom i trećem slučaju, dodavanjem točke, konstanta je postala realna te se dijeljenje obavlja u realnoj domeni.

# Cjelobrojno dijeljenje

---

- Primjer: Koliko iznosi:

a)  $9 / 4$

Rješenje: 2 (cjelobrojno dijeljenje)

b)  $9 \% 4$

Rješenje: 1 (ostatak cjelobrojnog dijeljenja  $9 : 4 = 2$  i ostatak 1)

# Prioritet osnovnih aritmetičkih operatora

---

1.      \* / %

2.      + -

Ako u izrazu ima više operatora jednakog prioriteta, izračunavaju se *slijeva nadesno*. Na primjer:

$$2 + 3 / 2 * 4 - 5 * 6 \% 8$$

$$2 + 1 * 4 - 30 \% 8$$

$$2 + 4 - 6$$

$$6 - 6$$

$$0$$

# Korištenje okruglih zagrada

---

Kada treba koristiti okrugle zagrade ?

- a) ako se želi promijeniti ugrađeni redoslijed izvođenja operacija
- b) u slučaju vlastite nesigurnosti
- c) radi bolje čitljivosti programa

$$2 + 3 / (2 * 4) - 5 * (6 \% 8) \Rightarrow -28$$

# Primjeri

---

- Koliki je rezultat sljedećeg izraza:

$$5 + 10 / 3 * ( 8 - 6 )$$

$$5 + 10 / 3 * 2$$

$$5 + 3 * 2$$

$$5 + 6$$

$$11$$

# Primjeri

---

- Koju će vrijednost poprimiti varijable i, x, c

```
int i;
```

```
double x, c, d;
```

nakon naredbi:

```
d = 6.0;
```

```
i = (int)(d + 1.73);
```

```
x = i / 2;
```

```
c = (double)i / 2;
```

- Rješenje:

```
i = 7, x = 3.0, c = 3.5
```

# Primjeri

---

Koje su vrijednosti i tipovi izraza nakon evaluacije:

1.        `3 / 2 * 2`
2.        `3 / 2 * 2.`
3.        `3 / 2. * 2`
4.        `3 / (float)2 * 2`
5.        `(double)3 / 2`
6.        `(double)(3 / 2)`
7.        `2+0.5       * 4`
8.        `(2 + 0.5) * 4`
9.        `(int)(0.5 + 2) * 4`
10.       `(int)1.6 + (int)1.6`
11.       `(int)(1.6 + 1.6)`

Primjer: Učitati vrijednosti za cjelobrojne varijable i, j i k te ispisati njihove vrijednosti i aritmetičku sredinu.

---

```
#include <stdio.h>
int main () {
 int i, j, k;
 float sredin;
 scanf ("%d %d %d", &i, &j, &k);
 sredin = (i + j + k) / 3.f;
 printf ("Aritm. sredina je %f", sredin);
 return 0;
}
```



# Rezultat izvođenja programa

---

Ulazni podaci:

1 2 4

Ispis na zaslonu:

```
Aritm. sredina je 2.333333
```