

# **Programiranje i programsko inženjerstvo**

Predavanja  
2015. / 2016.

## **2. 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 gumicom
- 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 algoritama koristit će se pseudokod ili dijagram toka.

# Primjer

---

- Programski zadatak
  - s tipkovnice pročitati dva cijela broja, pročitane brojeve ispisati na zaslon, a zatim na zaslon ispisati veći od pročitanih brojeva
  - pretpostaviti da će s tipkovnice sigurno biti učitani različiti brojevi

# Primjer - nastavak

---

- Vrlo sažeti algoritam opisan u pseudokodu koji koristi termine govornog jezika

pročitaj dva cijela broja

ispiši pročitane brojeve

odredi veći broj

ispiši rezultat

# Primjer - nastavak

- Detaljniji opis algoritma u pseudokodu koji koristi uobičajene termine i simbole

pročitaj (m, n)

*s tipkovnice pročitaj dvije vrijednosti i  
pohrani ih u varijable m i n*

ispiši (m, n)

*na zaslon ispiši vrijednosti varijabli m i n*

{ odredi veći broj }

*komentar*

ako je m > n tada

    rez := m

*vrijednost varijable m pridruži varijabli rez*

inače

    rez := n

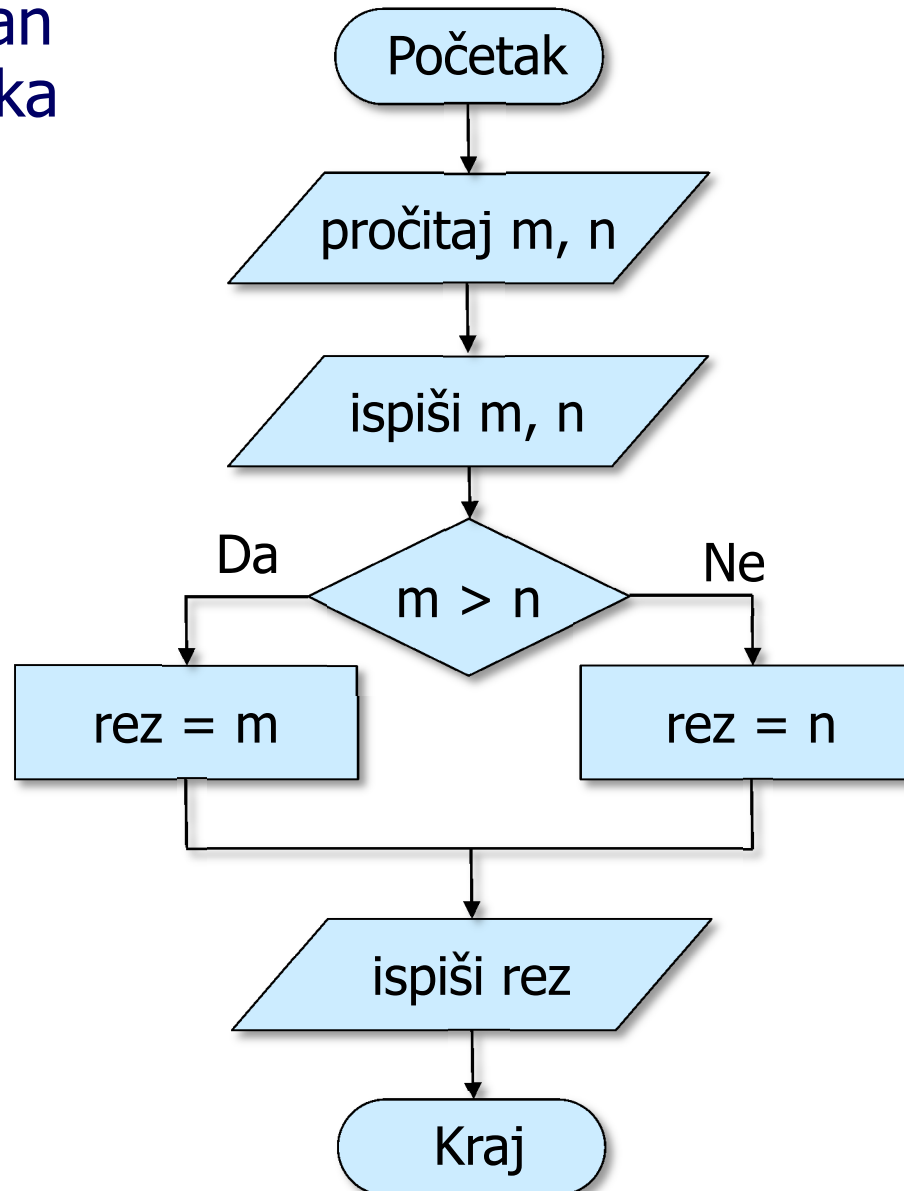
ispiši (rez)

kraj



# Primjer - nastavak

- Algoritam opisan dijagramom toka



# Primjer - nastavak

- Najčešće korišteni simboli u dijagramu toka



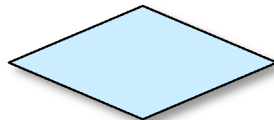
Početak, kraj



Proces, akcija



Ulaz, izlaz



Odluka, izbor



Redoslijed izvršavanja

# Primjer - nastavak

- Kôd u programskom jeziku C

```
#include <stdio.h>
int main(void) {
    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: u program prije prevođenja uključuje zaglavlja standardne ulazno-izlazne biblioteke `<stdio.h>`
  - time se, između ostalog, omogućuje ispravno korištenje funkcija poput `printf` i `scanf`
  - `<stdio.h>` ugraditi na početak svakog programa koji koristi funkcije `scanf` ili `printf`
- za sada, ulaz će uvijek biti tipkovnica, a izlaz zaslon (ekran)

# Objašnjenje

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

- glavna funkcija predstavlja mjesto na kojem počinje izvršavanje programa. Svaki program u C-u mora sadržavati točno jednu funkciju `main`.
- `int` ispred `main` znači da funkcija u pozivajući program vraća cijeli broj
- `void` unutar zagrada znači da funkcija `main` ne prima niti jedan ulazni argument
- funkcija `main` uvijek završava naredbom `return` koja u pozivajući program vraća rezultat. Za sada, pozivajućem programu uvijek vratiti cijeli broj nula, kao što je prikazano u primjeru.

# 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 isključivo cijeli brojevi

```
float x, y;  
float z;
```

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

# Objašnjenje

```
/* ucitaj dva cijela broja */
```

- komentar koji nema utjecaja na izvršavanje programa

```
/* ucitaj  
   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 `&imeVariable`

npr, preko tipkovnice je uneseno ➡ 37 5↵

- nakon obavljanja naredbe `scanf`, varijabla `m` će sadržavati vrijednost 37, a varijabla `n` vrijednost 5

- korištenjem formatske 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 učitane vrijednosti za varijable m=156, n=20, na zaslon će se ispisati

```
156 20↵
```

# Objašnjenje

- formatskom specifikacijom se može utjecati na širinu ispisa, npr.

```
printf("%d,%5d,%2d,%2d,kraj", 10, 20, 30, 150);
```



```
10,    20,30,150,kraj
```

```
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, -3.455555555);
```

- %f uvijek ispisuje 6 znamenki iza decimalne točke



15.200000 -3.455556

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

```
printf("%7.3f,%10.4f,%.4f", 15.2, -3.4555555, 127.4555555);
```



15.200, -3.4556,127.4556

%7.3f: ispisuje ukupno 7 znakova, od toga tri iza decimalne točke  
%10.4f: ispisuje ukupno 10 znakova, od toga četiri iza decimalne točke  
%.4f: ispisuje četiri znaka iza decimalne točke, a ispred koliko treba

# Objašnjenje

```
float pi, e;  
e = 2.718281;  
pi = 3.141592;  
printf("e = %f\nPI = %f", e, pi);
```

```
e = 2.718281↵  
PI = 3.141592
```

- uočiti: cijeli tekst naveden u formatu funkcije printf ispisuje se na zaslon. Izuzetak su formatske specifikacije (npr. %d, %f) koje se redom zamjenjuju stvarnim vrijednostima preostalih navedenih argumenata
- tipovi formatskih specifikacija navedenih u formatu moraju odgovarati tipovima argumenata koji slijede (isto vrijedi i za funkciju scanf)

# 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 = 17 % 3;  
  
rez = m + 5 - 3;  
  
rez = 3 / 4;
```

- u `rez` pridruži 5
- u `rez` pridruži rezultat izraza, tj. 15
- u `rez` pridruži rezultat izraza, tj. 4
- u `rez` pridruži rezultat izraza, tj. ostatak cjelobrojnog djeljenja 17 s 3
- u `rez` pridruži vrijednost izraza. tj. vrijednost varijable `m` uvećane za 5 i umanjene za 3
- u `rez` pridruži rezultat izraza, tj. 0 !!!

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

- u `x` pridruži rezultat izraza, tj. 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 sve naredbe unutar prvih vitičastih zagrada
  - inače se obavljaju sve naredbe unutar vitičastih zagrada iza else
  - u oba slučaja, nastavlja se s naredbama u nastavku (konkretno printf)
- `if ( m > n )`      ako je sadržaj m veći od sadržaja n
- `>=, <, <=`      slično: veći ili jednak, manji, manji ili jednak
- `if ( m != n )`      ako je sadržaj m različit od sadržaja n
- `if ( m == n )`      ako je sadržaj m jednak sadržaju n
  - uočiti, ne ovako: `if ( m = n )`

# Objašnjenje

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

- else dio naredbe se ne mora uvijek navesti
  - ako se uvjet u zagradama iza if izračuna kao istina, obavljaju se naredbe unutar prvih vitičastih zagrada
  - neovisno o rezultatu izračunavanja uvjeta, nastavlja se s naredbama u nastavku

# Domaća zadaća (1) – instalacija programskog paketa MinGW i rad s C prevodiocem

---

- S web stranice predmeta preuzeti, proučiti i obaviti na svom računalu:
  - mapa *PIPI-zimski semestar 2015/2016* → *Upute*: Upute za korištenje paketa MinGW i prevodioca GCC
    - rok: prije drugog (tj. sljedećeg) predavanja
  - mapa *PIPI-zimski semestar 2015/2016* → *Upute*: Upute za korištenje znakovnog sučelja u operacijskom sustavu Windows
    - rok: prije četvrtog predavanja



# Domaća zadaća (2) – rješavanje zadataka za vježbu uz predavanja

---

- **Trajna domaća zadaća:** rješavanje zadataka koji su temom vezani uz upravo održano predavanje. Zadaci će se prateći dinamiku predavanja objavljivati na web stranici predmeta u mapi *PIPI-zimski semestar 2015/2016 → Zadaci za vježbu → Vježbe uz predavanje*
  - zadaci uz današnje predavanje: **1. vježbe uz predavanje**
- Uz zadatke se objavljuju i rješenja, ali se preporuča ne gledati rješenja zadataka prije nego ih sami riješite i na računalu testirate vlastito rješenje
- Vježbe uz predavanja treba uvijek obaviti najkasnije prije sljedećih predavanja (npr. 1. vježbe uz predavanja treba obaviti najkasnije prije 2. predavanja)

# Domaća zadaća (3) – upoznavanje sa sustavom Ahyco

---

- Svaki student treba imati pristup webu FER-a (ako ga nema mora se javiti u CIP i zatražiti pristup webu FER-a)
- S web stranice predmeta preuzeti i proučiti:
  - mapa *PIPI-zimski semestar 2015/2016* → *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)
  - rok: ponedjeljak, 26.10.2015

Programiranje, prevodilac, vrste pogrešaka


# Vrste programske potpore

---

- Sistemska programska potpora
  - upravlja računalnim sklopovljem, omogućuje korištenje aplikativne programske potpore
  - operacijski sustavi (UNIX/Linux, Windows, Mac OS)
  - uslužni (*utility*) programi (konfiguracija, optimizacija, održavanje računala)
- Aplikativna (namjenska, primijenjena) programska potpora
  - izvršavanje korisnih zadataka (osim samog upravljanja sklopovljem)

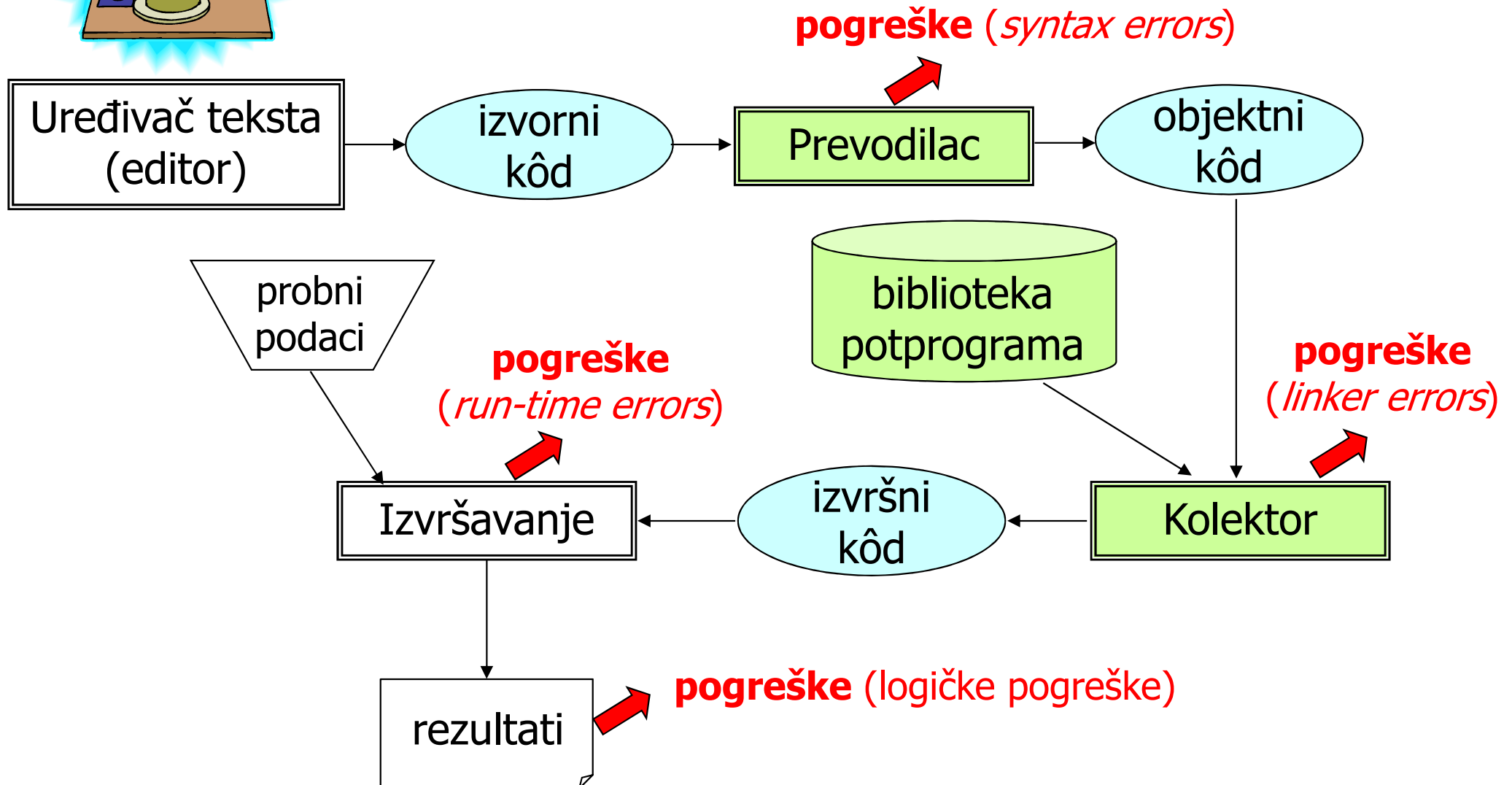
# Redoslijed rješavanja manjih programa

---

1. Uočavanje (identifikacija) problema i postavljanje programskog zadatka
  2. Oblikovanje programa (razvoj algoritma)
  3. Implementacija (kodiranje) algoritma u programskom jeziku, upis programskog kôda u računalo
  4. Prevođenje (kompilacija) programa
  5. Ispravljanje formalnih pogrešaka
  6. Kolekcija programa (stvaranje izvršnog programa)
  7. Ispravljanje pogrešaka povezivanja
  8. Izvođenje programa s test podacima
  9. Ispravljanje pogrešaka izvršavanja i logičkih pogrešaka
  10. Korištenje programa s aktuelnim podacima
- 
- ```
graph TD; 4 --> 3; 5 --> 3; 7 --> 3; 8 --> 3;
```



# Programiranje u užem smislu



# Programiranje u užem smislu

---

- Unos izvornog programa (*source code*)
  - ASCII uređivač teksta, editor (Notepad, vi, ...)
  - ili 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



# Budite C-prevodilac (C-compiler) i komentirajte ponuđeno rješenje

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

sintaktička  
pogreška

pogreška  
povezivanja  
(*linking*)

logička pogreška

# Primjer

---

- Napisati pseudokod i C program za sljedeći problem:
  - Učitati koeficijente pravaca  $y = a_1 x + b_1$  i  $y = a_2 x + b_2$
  - Ovisno o vrijednostima koeficijenata, ispisati poruku  
`Pravci su paralelni`  
  
`ili`  
  
`Pravci se sijeku u x = xxxx.xx, y = xxxx.xx`
- Testirati program

# 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(void) {int m,n,
rez;scanf("%d %d"
,&m,
&n)
;          printf
(
"Ucitani su %d i %d\n", m, n);if(
m >
n){rez=      m;}else{rez
= n;}printf("Veci broj je %d\n", rez);return 0;}
```

# Primjer - što radi ovaj program?

```
#include <stdio.h>
int main(void) {
    int m, n, rez;
    scanf("%d %d", &m, &n);
    printf("Ucitani su %d i %d\n", m, n);
    if ( m > n ) {
        rez = m;
    } else {
        rez = n;
    }
    printf("Veci broj je %d\n", rez);
    return 0;
}
```

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

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

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

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

 Različiti stilovi

# Ključne riječi

- predefinirani identifikatori koji za prevodioca imaju posebno značenje
- ključne riječi pišu se malim slovima
- Prema standardu ANSI C (C89), 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

# Struktura C programa

---

- C program se sastoji od imenovanih blokova, deklaracija/definicija varijabli i funkcija, direktiva pretprocesoru
  - imenovani blokovi se nazivaju **funkcije**
- 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;  
}
```

*početak imenovanog bloka (funkcije)*

*definicija varijable*

*početak neimenovanog bloka*

*definicija varijable*

*početak neimenovanog bloka*

*naredba*

*naredba*

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

*početak imenovanog bloka (funkcije)*

# Struktura C programa

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

## ISPRAVNO:

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

## POGREŠNO:

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

## ISPRAVNO, ALI SE NE PREPORUČA:

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

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

# Struktura C programa

```
#include <stdio.h>
#define PI 3.14159
int main(void) {
    float r;
    float opseg;
    printf("Unesi polumjer: ");
    scanf("%f", &r);
    opseg = 2 * r * PI;
    printf("%9.2f\n", opseg);
    return 0;
}
```

*direktive pretprocesoru:*

*zaglavlja za ulaz-izlaz*

*definicija simboličke konstante*

*funkcija **main***

*definicije varijabli*

*tijelo funkcije **main***

*kraj funkcije **main***

# Komentari

---

- komentari se mogu protezati kroz više linija
- izbjegavati komentare koji opisuju očito:  
`printf("Unesi n: "); /* Ispis teksta na zaslon */`  
zato što program bez potrebe 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).
- `#define PI 3.14159` definira simboličku konstantu `PI` i pridjeljuje joj vrijednost. Simboličke konstante su naročito korisne za parametrizaciju programa.

```
#include <stdio.h>
int main(void) {
    float r, opseg, površina;
    scanf("%f", &r);

    if (polumjer > 0) {
        opseg = 2 * r * 3.14159;
        površina = r * r * 3.14159;
        ...
    }
}
```

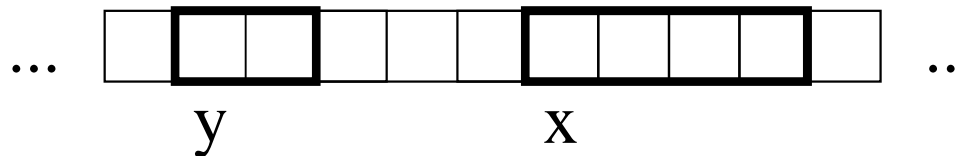
```
#include <stdio.h>
#define PI 3.14159
int main(void) {
    float r, opseg, površina;
    scanf("%f", &r);

    if (polumjer > 0) {
        opseg = 2 * r * PI;
        površina = r * r * PI;
        ...
    }
}
```

Uočiti: iza pretprocesorskih naredbi ne stavlja se ;

# Variable

- Općenito: promjenljiv podatak (lat. *variabilis*-promjenljiv)
- U programiranju: prostor u memoriji računala, poznate veličine, kojemu je dodijeljeno ime i čiji se sadržaj može mijenjati
- Smještaj u memoriji računala:



- Primjer:

x 

|       |
|-------|
| 10.50 |
|-------|

      y 

|   |
|---|
| 3 |
|---|

- Nakon obavljanja

**x = -3.1;**

**y = 10;**

x 

|      |
|------|
| -3.1 |
|------|

      y 

|    |
|----|
| 10 |
|----|

# 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 maticni broj float`~~

- svaka varijabla se obavezno mora definirati prije korištenja

`int i, n;`

`float sum;`

- 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
- smisljena imena varijabli unapređuju jasnoću programa

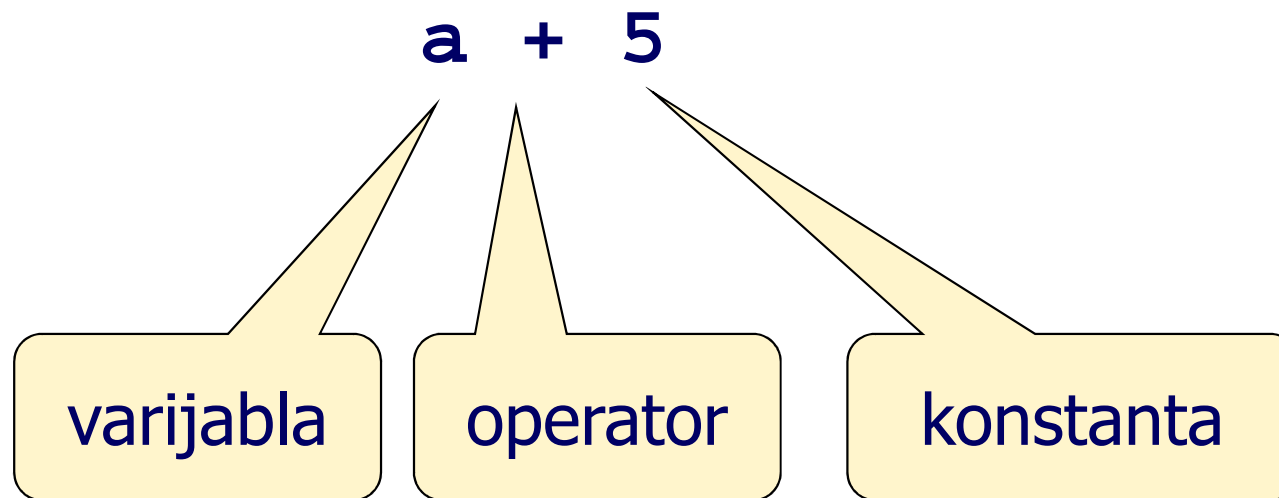
Izrazi, operator pridruživanja, aritmetički i  
relacijski operatori



# Izrazi

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

- Primjer:



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

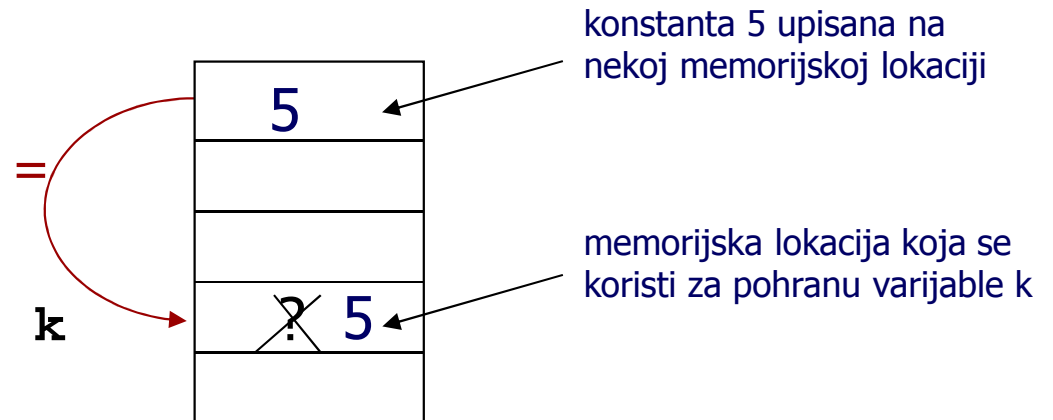
# Izraz pridruživanja

## ■ Pridruživanje vrijednosti: `varijabla = izraz;`

- simbol u pseudokodu `:=`
- u C-u `=`

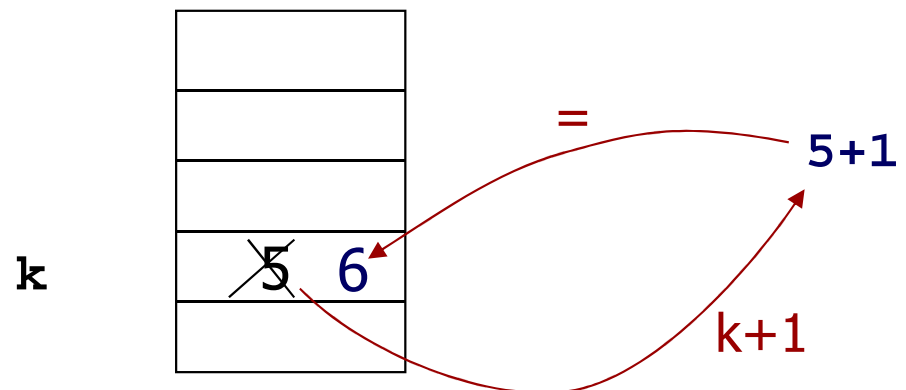
## ■ Na primjer:

- `k = 5;`



- `k = k + 1;`

aritmetički izraz



# L-value u izrazu pridruživanja

- s lijeve (**L**eft) strane operatora pridruživanja uvijek se mora nalaziti tzv. *L-value*. Pojednostavljeno, L-value je izraz kojim se referencira promjenljivi prostor u memoriji.
- za sada će se kao L-value koristiti isključivo ime varijable

```
int m, n, k;
```

```
n = 15 + 3;
```

```
m = n;
```

```
m = m + n + 1;
```

```
k + 1 = m + 1;
```

```
7 = m;
```

n jest L-value

m jest L-value

k + 1 nije L-value

7 nije L-value

- u nekim od sljedećih predavanja pokazati će se kako ime varijable nije jedini mogući L-value

# Rezultat izraza pridruživanja

---

- izraz pridruživanja se prvenstveno koristi za pridruživanje vrijednosti
- izraz pridruživanja, osim što obavlja pridruživanje, također "po evaluaciji" daje rezultat. Taj rezultat najčešće se ne koristi.

```
int m;  
m = 15 * 2;
```

- aritmetički izraz daje rezultat 30, izrazom pridruživanja 30 se pridružuje varijabli m, a konačni rezultat izraza pridruživanja jest opet vrijednost 30 (vrijednost koja je upravo pridružena). U ovom primjeru vrijednost izraza pridruživanja ostala je neiskorištena.
- u sljedećem primjeru, rezultat izraza pridruživanja će se iskoristiti:

```
int m;  
printf("%d", m = 5);
```

**Primjer:** Zadano je  $x=14$  i  $y=-9$ . Program 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  $y$ .

```
#include <stdio.h>
int main(void) {
    int x, y, p;
    x = 14;
    y = -9;
    printf ("x=%d, y=%d\n", x, y);
    /* zamijeniti vrijednosti x i y */
    p = x;
    x = y;
    y = p;

    printf ("Nakon zamjene: x = %d, y = %d\n", x, y);
    return 0;
}
```

| x  | y  | p  |
|----|----|----|
| ?  | ?  | ?  |
| 14 | ?  | ?  |
| 14 | -9 | ?  |
| 14 | -9 | 14 |
| -9 | -9 | 14 |
| -9 | 14 | 14 |

$x = 14, y = -9$

Nakon zamjene:  $x = -9, y = 14$

# Osnovni aritmetički operatori

---

| Aritmetički operator | Značenje                                              |
|----------------------|-------------------------------------------------------|
| +                    | zbrajanje                                             |
| -                    | oduzimanje                                            |
| *                    | množenje                                              |
| /                    | dijeljenje                                            |
| %                    | ostatak kod cjelobrojnog dijeljenja (modulo, modulus) |

# Djelovanje aritmetičkih operatora na cjelobrojne operande

---

```
int a, b;
```

```
a = 10;
```

```
b = 3;
```

| Izraz    | Rezultat |
|----------|----------|
| $a + b$  | 13       |
| $a - b$  | 7        |
| $a * b$  | 30       |
| $a / b$  | 3        |
| $a \% b$ | 1        |

# Djelovanje aritmetičkih operatora na realne operande

---

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

| Izraz    | Rezultat |
|----------|----------|
| $a + b$  | 14.5     |
| $a - b$  | 10.5     |
| $a * b$  | 25.0     |
| $a / b$  | 6.25     |
| $a \% b$ | pogreška |

a može li:      12. % 5



# Prioritet aritmetičkih operatora

- operatori množenja, dijeljenja i ostatka cjelobrojnog dijeljenja imaju veći prioritet od operatora zbrajanja i oduzimanja
  - $a + b * c$
  - $b * c + a$ } u oba slučaja prvo se izračunava  $b * c$
- ako aritmetički operatori imaju jednak prioritet (npr. množenje, dijeljenje i ostatak cjelobrojnog dijeljenja), tada se operacije obavljaju s lijeva na desno
  - $a / b * c \equiv (a / b) * c$
  - $x / a + b * c + d * e \equiv ((x / a) + (b * c)) + (d * e)$
- ako pretpostavljeni redoslijed obavljanja operacija treba promijeniti, koristiti okrugle zagrade, npr.
  - $(a + b) * c$
  - $b * (c + a)$
  - $x / ((a + b) * (c + d) * e)$

# Relacijski izrazi

- izrazi kojima se pomoću relacijskih operatora testira odnos među operandima. Uvrštavanjem vrijednosti operanada u izraz dobiva se sud, koji se evaluira kao istinit ili lažan.
- relacijski izraz je jedan jednostavan oblik *logičkog* izraza
  - složeniji oblici logičkih izraza gradit će se uz pomoć logičkih operatora

| Relacijski operator | Značenje          |
|---------------------|-------------------|
| >                   | veće              |
| <                   | manje             |
| >=                  | veće ili jednako  |
| <=                  | manje ili jednako |
| ==                  | jednako           |
| !=                  | različito         |

```
float a, b;  
a = 2.1;  
b = 2.2;  
if (a != b) {  
    ...  
}
```

Naredbe za promjenu programskog slijeda  
(kontrolne naredbe)

Jednostrana i dvostrana selekcija

# Naredbe za promjenu programskog slijeda (kontrolne naredbe)

---

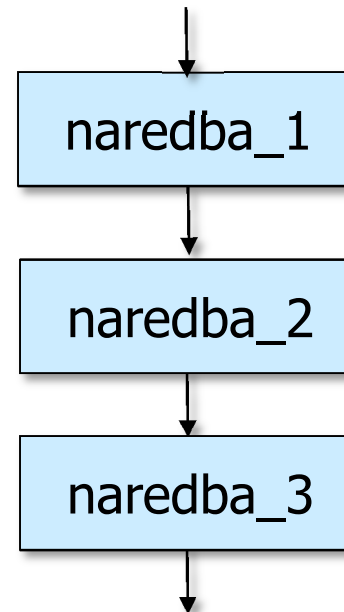
- Normalan programski slijed:

naredba\_1

naredba\_2

naredba\_3

...



# Kontrolna naredba `if` - jednostrana selekcija

- Pseudokôd

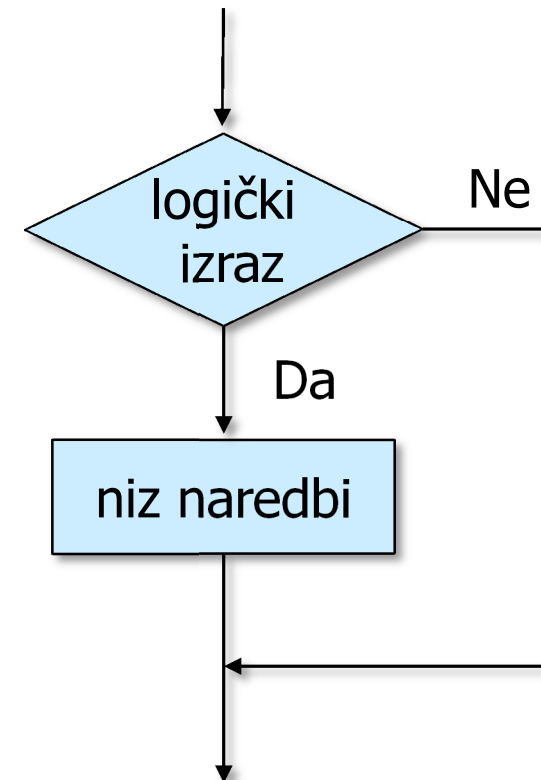
ako je `logički_izraz` tada  
|  
naredbe

- U C-u

```
if (logički_izraz) naredba;
```

ili

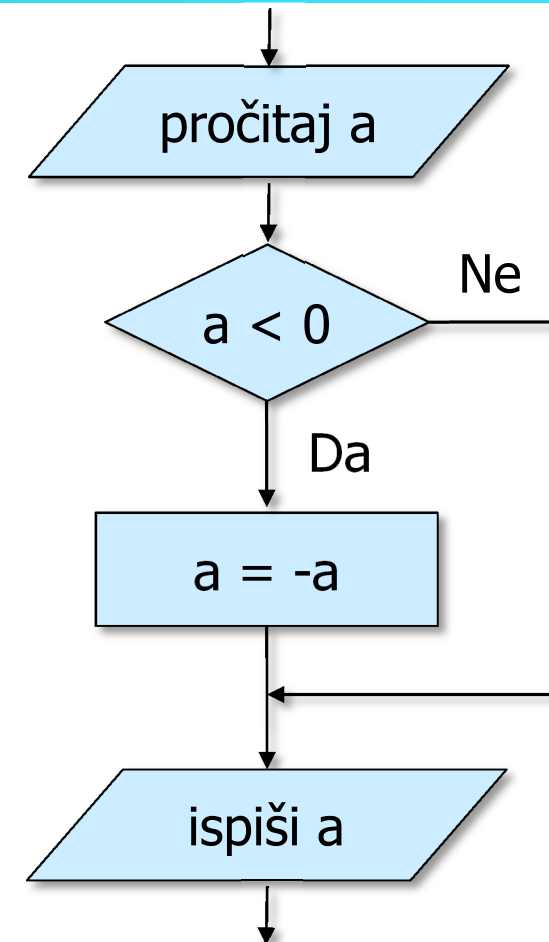
```
if (logički_izraz) {  
    niz naredbi  
}
```



# Primjeri za jednostranu selekciju

- U cjelobrojnu varijablu **a** učitati broj, vrijednost varijable promijeniti u njenu apsolutnu vrijednost te ju ispisati na zaslon.

```
int a;  
scanf("%d", &a);  
if ( a < 0 )  
    a = -a;  
printf("Apsolutna vrijednost je %d\n", a);
```

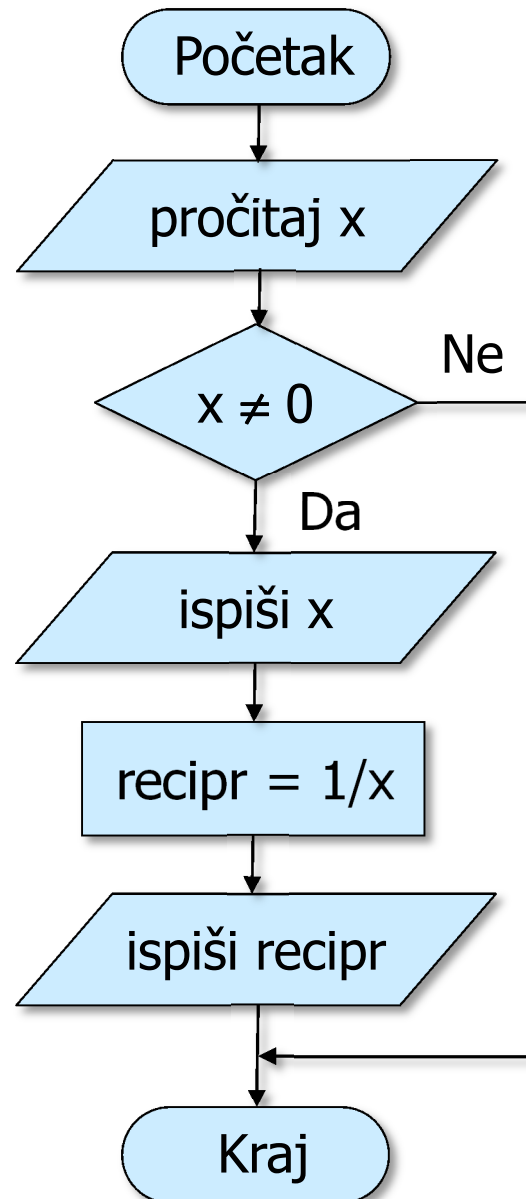


# Primjer

---

- Programski zadatak
  - u varijablu  $x$  s tipkovnice učitati realni broj. Ako je recipročna vrijednost za  $x$  definirana, na zaslon ispisati sadržaj varijable  $x$ , izračunati i zatim na zaslon ispisati recipročnu vrijednost od  $x$

# Rješenje - dijagram toka





# Rješenje - programski kod

---

```
#include <stdio.h>

int main(void) {
    float x, recipr;
    scanf("%f", &x);
    if (x != 0)
        printf("Procitan je broj %f\n", x);
        recipr = 1 / x;
        printf("Recipročna vrijednost od %f je %f\n", x, recipr);
    return 0;
}
```

Što je neispravno u prikazanom programskom kodu?

# Primjeri za jednostranu selekciju

---

- Što obavlja sljedeća naredba:

```
if ( a < 0 ) a = -a;
```

- Postoji li bitna razlika sljedećeg rješenja u odnosu na prethodno:

```
if ( a < 0 ) {  
    a = -a;  
}
```

- Postoji li bitna razlika sljedećeg rješenja u odnosu na prethodno:

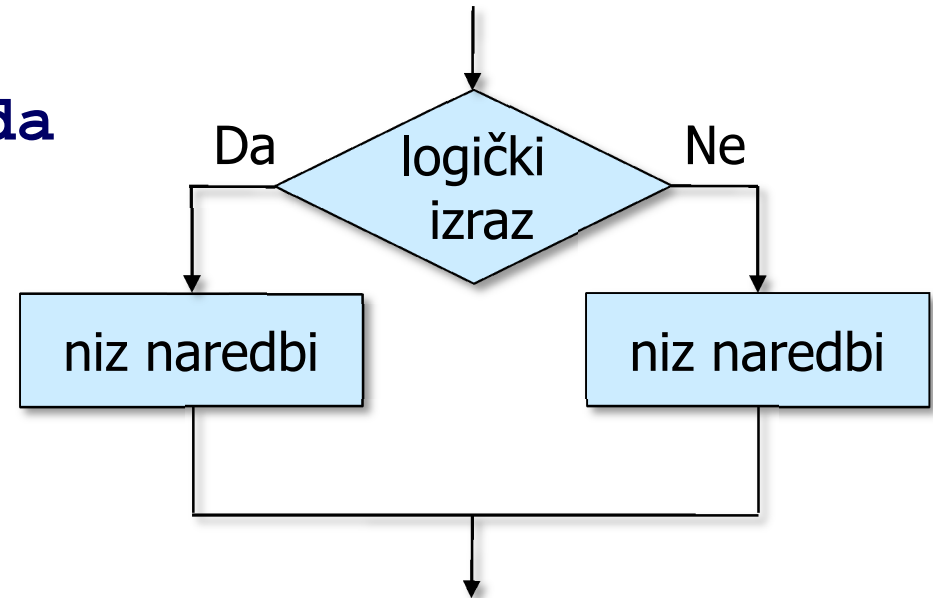
```
if ( a < 0 ); a = -a;
```

# Kontrolna naredba `if`

## - dvostrana selekcija

### ■ Pseudokôd

```
ako je logički_izraz tada
| niz_naredbi_1
inače
| niz_naredbi_2
```



### ■ U C-u

```
if (logički_izraz) {
    niz_naredbi_1
} else {
    niz_naredbi_2
}
```

- ako niz\_naredbi obuhvaća samo jednu naredbu, pripadne vitičaste zagrade se smiju ispustiti

# Primjeri za dvostranu selekciju

---

- Programski zadatak

- Na zaslon ispisati poruku `Upisite cijeli broj`
- Učitati cijeli broj s tipkovnice
- Izračunati apsolutnu vrijednost učitanoog broja te na zaslon ispisati učitani broj i njegovu apsolutnu vrijednost u obliku

`Apsolutna vrijednost od x je x`

# Rješenje (loše!)

---

```
#include <stdio.h>
int main(void) {
    int broj, aps;
    printf("Upisite cijeli broj >");
    scanf("%d", &broj);
    if (broj < 0) {
        aps = -broj;
        printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
    } else {
        aps = broj;
        printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
    }
    return 0;
}
```

# Rješenje (dobro)

---

```
#include <stdio.h>
int main(void) {
    int broj, aps;
    printf("Upisite cijeli broj >");
    scanf("%d", &broj);
    if (broj < 0) {
        aps = -broj;
    } else {
        aps = broj;
    }
    printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
    return 0;
}
```

# Rješenje (jednako dobro)

---

```
#include <stdio.h>
int main(void) {
    int broj, aps;
    printf("Upisite cijeli broj >");
    scanf("%d", &broj);
    if (broj < 0)
        aps = -broj;
    else
        aps = broj;

    printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
    return 0;
}
```

# Rješenje (također ispravno)

---

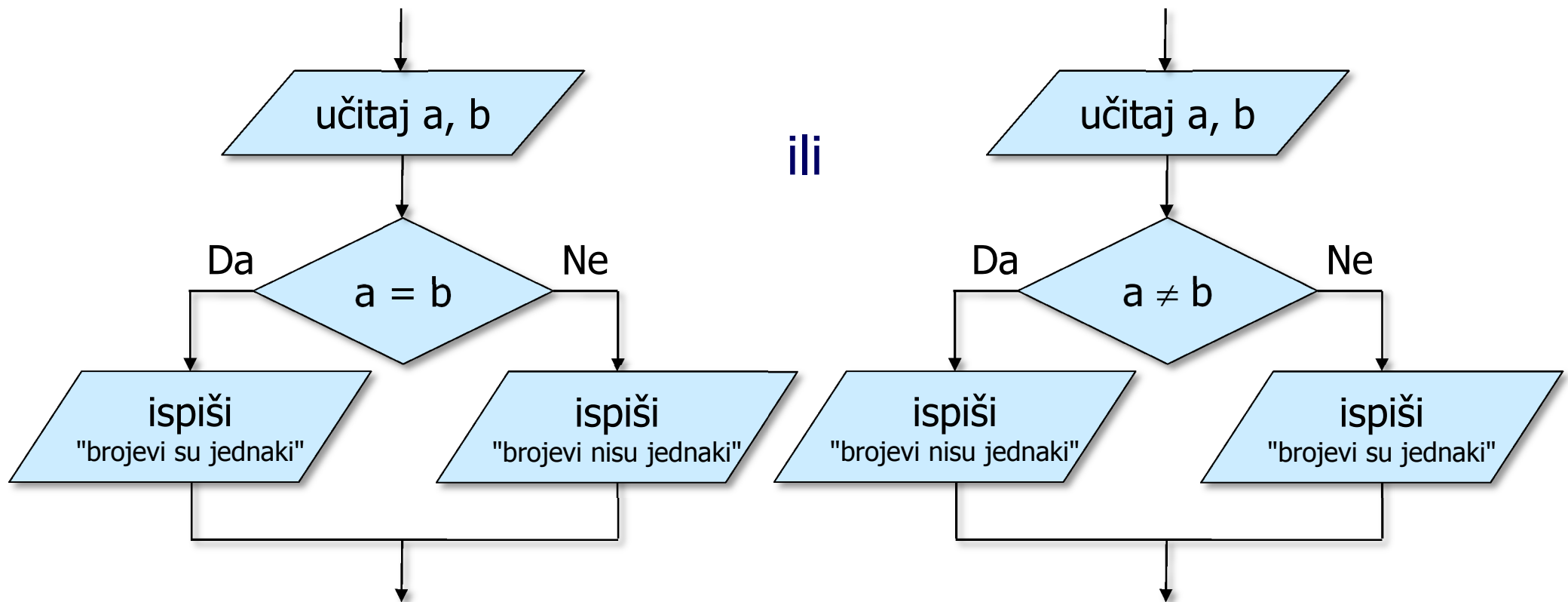
```
#include <stdio.h>
int main(void) {
    int broj, aps;
    printf("Upisite cijeli broj >");
    scanf("%d", &broj);
    printf("Apsolutna vrijednost od %d je ", broj);
    if (broj < 0)
        printf("%d\n", -1*broj);
    else
        printf("%d\n", broj);

    return 0;
}
```



# Primjeri za dvostranu selekciju

- Zadatak: učitati dva cijela broja. Ispisati "brojevi su jednaki" ili "brojevi nisu jednaki".



# Korištenje neispravnog operatora u logičkom izrazu

---

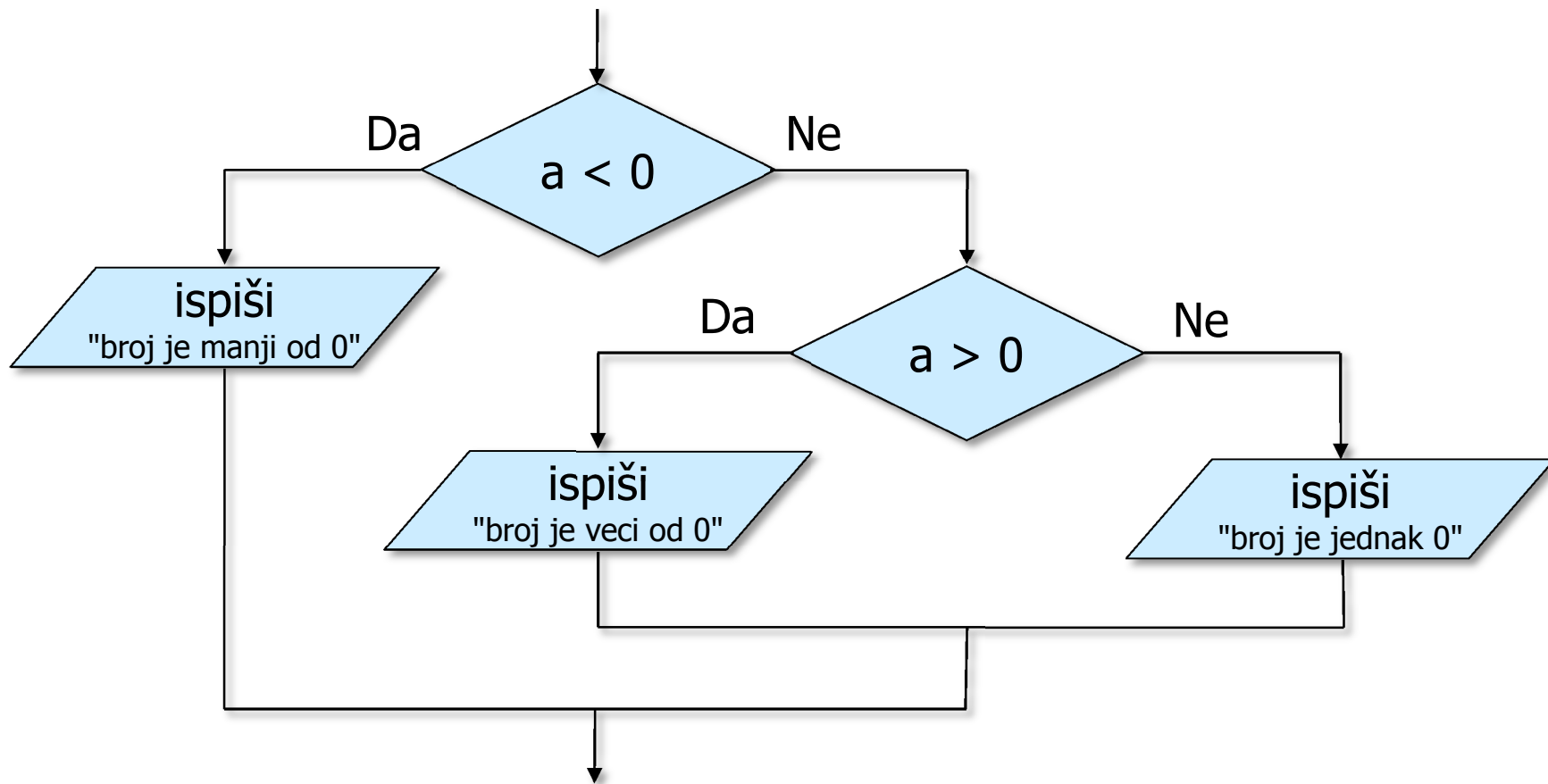
- Zašto sljedeće rješenje nije ispravno?

```
#include <stdio.h>
int main(void) {
    int a, b;
    scanf("%d %d", &a, &b);
    if (a = b)
        printf ("Brojevi su jednaki\n");
    else
        printf ("Brojevi nisu jednaki\n");
    return 0;
}
```

- Što će se programom ispisati u sljedeća dva slučaja:
  - 1.)  $a = 25, b = 4$
  - 2.)  $a = 0, b = 0$

# Primjeri za dvostranu selekciju

- Zadatak: učitati cijeli broj. Ovisno o učitanoj broju, ispisati "broj je veci od 0", "broj je jednak 0" ili "broj je manji od 0".



# Rješenje

---

```
#include <stdio.h>
int main(void) {
    int a;
    printf("Upisite cijeli broj:");
    scanf("%d", &a);
    if (a < 0) {
        printf("Broj je manji od 0\n");
    } else
        if (a > 0) {
            printf("Broj je veci od 0\n");
        } else {
            printf("Broj je jednak 0\n");
        }
    return 0;
}
```

# 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(void) {
    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(void) {
    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?

# Primjer

---

- Programski zadatak
  - s tipkovnice pročitati tri cijela broja, na zaslon ispisati pročitane brojeve, te najveću učitano vrijednost
  - pretpostaviti da će učitani brojevi sigurno biti različiti



# Primjer - nastavak

---

```
pročitaj (x,y,z)
ispiši (x,y,z)
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

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

# Primjer - nastavak

---

- Još jedno od mogućih rješenja:

```
pročitaj (x,y,z)
ispiši (x,y,z)
rez := x
ako je y > rez tada
|   rez := y
ako je z > rez tada
|   rez := z
ispiši (rez)
kraj
```

# Primjer - nastavak

---

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