# Programski jezik I

Uvod

# Osoblje

Nastavnici: **Doc. dr Miloš Brajović**
mejl: **milosb@ucg.ac.me**
kabinet: **333**

**Doc. dr Stefan Vujović**
mejl: **stefanv@ucg.ac.me**
kabinet: **Laboratorija za obradu signala**

Saradnici: **M.Sc. Andrej Cvijetić**
mejl: **andrejc@ucg.ac.me**
kabinet: **Laboratorija za multimedije**

# Bodovanje

- Laboratorijske vježbe **10** × (**max**)**1**
- Kolokvijum **40**
- Ispit **50**
- **Ne postoji** mogućnost organizovanja predrokova i vanrednih provjera znanja (postoje vrlo rijetki izuzeci, predviđeni propisima)
- Mogućnost dobijanja ekstra bodova tokom nastave i vježbi.
- Ocjene se formiraju prema pravilu:

  $50 \leq \mathbf{E} < 60$

  $60 \leq \mathbf{D} < 70$

  ...

  $90 \leq \mathbf{A} \leq 100.$

# Literatura


- Osnovna literatura (pokriva čitavo gradivo) je:
  - Slobodan Đukanović, Igor Đurović, Vesna Popović-Bugarin: "Programski jezik C sa zbirkom riješenih zadataka",
    https://slobodan.ucg.ac.me/books/C/

- Dodatna literatura:
  - A. Hansen: "Programiranje na jeziku C – potpuni vodič za programski jezik C", Mikroknjiga, Beograd, 2000.
  - L. Kraus: "Zbirka riješenih zadataka iz programskog jezika C", Mikroknjiga, Beograd, 1997.
  - Internet, forumi.

# Struktura kursa

- Uvod sa pregledom istorijata programskih jezika i korišćene terminologije (oko 5% nastave).
- Programski jezik C kao paradigma strukturnog programiranja (oko 75% nastave).
- Osnove linkovanih tipova podataka (oko 20% nastave).

- Kolokvijumi i ispiti se održavaju u računarskoj sali.
- Laboratorijske vježbe su namijenjene isključivo za samostalan rad studenata uz konsultacije sa predmetnim asistentom.

# Jezik računara

- Savremeni elektronski računari rade na principu binarne logike sa alfabetom {**0**,**1**}.
- Očigledno je veoma teško ljudsku logiku, zasnovanu na procedurama, pretvoriti u binarni zapis direktnim putem.
- Poseban je problem što relativno prosta procedura zapisana binarno može da ima desetine hiljada, pa i milione bita, što je čini nemogućom za održavanje i prepravljanje.
- Stoga su se razvili programski jezici kao posrednici između jezika procedure i jezika koji razumiju računari.

# Od čovjeka do jedinica i nula

Išao, bio, radio, €, Facebook

**Programski jezici**

Viši: C, C++, C#, Java, MATLAB, Python...

Niži: Asembler

0101001000
1110100010
0100100101

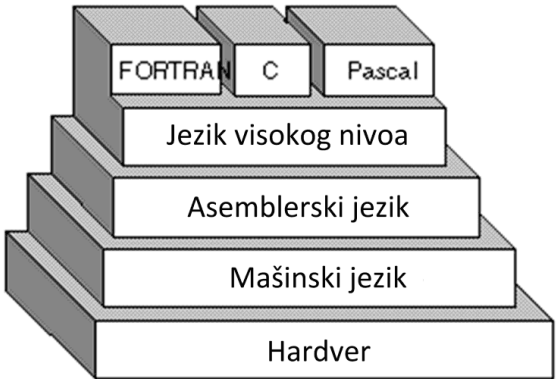
# Programski jezici

- **Mašinski jezik** je jezik koji razumije procesor računara. Predstavlja binarnu reprezentaciju (0 i 1) procesorskih instrukcija.
- **Asemblerski jezik** je jezik instrukcija procesora, na primjer:

```
ADD R1, R2, R3      // saberi registre R2 i R3 i smjesti zbir u R1
MULT R1, R2, R3     // pomnoži registre R2 i R3 i smjesti proizvod u R1
LOAD R1, A          // učitaj podatak sa mem. adrese A i smjesti ga u R1
```

- Program napisan u **višem programskom jeziku** podsjeća na jednostavne direktive engleskog jezika, kombinovane sa preciznim matematičkim formulacijama.

```
while(I < 10) {
   if(I > 2)
      X = Y - 3;
}
```



# Faze u razvoju programskih jezika

- **I faza** (peta i šesta decenija XX vijeka): Razvoj prvih jezika i prva programerska iskustva
  - **Plankalkul** – prvi pravi programski jezik (1943-45.)
  - **Short-Code** – prvi koji radi na elektronskim računarima
  - **FORTRAN** – primjena u naučno-istraživačkom radu
  - **COBOL** – rješavanje problema u ekonomiji, uključujući rad sa bazama podataka
- **II faza** (sedma decenija XX vijeka): Kreiranje prvih kvalitetnih programskih jezika i početak programerske edukacije
  - **Space Wars** – prva kompjuterska igra na MIT-u (1962.)
  - **BASIC** – programiranje za širu populaciju
  - **Pascal** – uspostavljen standard za naredbe kontrole toka programa

# Faze u razvoju programskih jezika

- **III faza** (osma decenija XX vijeka): Podrška programiranju hardvera
  - **Prolog** – podrška ekspertnim sistemima i vještačkoj inteligenciji
  - **Smalltalk** – objektno-orijentisano programiranje
  - **C** – početak modernog doba u programskim jezicima. Ne samo programski, već i programerski jezik – zamišljen, kreiran i razvijen od strane programera, tj. ljudi koji su ga koristili. Omogućen jednostavan pristup hardveru.
- **IV faza** (deveta decenija XX vijeka): Objektno-orijentisano programiranje
  - **Softverska kriza** – programeri ne mogu da isporuče kvalitetan i testiran softver usled prevelike potražnje
  - **C++** – jezik C nadograđen pojmom klase

# Faze u razvoju programskih jezika

- **V faza** (posljednja decenija XX vijeka): Prenosiv programski kôd. Web programiranje
  - **Visual Basic** – zvanični makro jezik u Microsoft programskim paketima
  - **Python** – vještačka inteligencija i big data
  - **R** – data mining i big data
  - **Java** – prenosiv programski kôd. Nastao pod uticajem C i C++
  - **HTML** – jezik za opis Web strana (nije programski jezik)
  - **JavaScript** – danas možda najpopularniji jezik
  - **PHP** – izrada dinamičkih veb stranica

# Faze u razvoju programskih jezika

- **VI faza** (2000-danas): Veliki broj pravaca
  - **C#** – Nastao pod uticajem C i C++. Desktop aplikacije
  - **CSS** – stilizovanje Web strana
  - **Web frameworks**
    - PHP frameworks – Laravel, CodeIgniter, Symphony
    - JavaScript frameworks – Angular, ReactJS, Vue.js
    - CSS frameworks – Bootstrap
  - **Programiranje pametnih uređaja** (mobilni telefoni, tableti, satovi...)
    - Android platforma – Java, Kotlin
    - iOS platforma – Objective C, Swift
  - **Cross-platform programiranje** – razvoj za više platformi istovremeno. Dominantnu ulogu imaju **JavaScript** (Apache Cordova, Appcelerator Titanium, RhoMobile) i **C#** (Xamarin, Visual Studio).
  - **Programiranje hardverskih platformi** – Arduino, Raspberry Pi.

# Terminologija

- Kod nas nije standardizovana računarska terminologija, pa se za jedan pojam koristi više naziva, i za različite pojmove se često koriste isti nazivi. Stoga ćemo uvesti definicije nekoliko najvažnijih pojmova.
- Podatak je objekat koji se obrađuje.
- Podaci mogu biti elementarni i složeni.
- Nad podacima se izvode elementarne operacije.
- Redosljed izvršavanja operacija određuje kontrola toka programa.
- Izvorni kôd programa (eng. *source code*) se piše u (višem) programskom jeziku.
- Izvorni kôd je potrebno prevesti na neki način u kôd razumljiv računaru, sastavljen od 0 i 1, koji često nazivamo mašinski kôd.
- Prevođenjem se bave programi koje jednim imenom nazivamo translatori.
- Dvije osnovne grupe translatora su interpreteri i kompajleri.

# Interpreteri

- Interpreteri prolaze kroz izvorni kôd naredbu po naredbu, vrše prepoznavanje naredbe i pozivaju dio kôda interpretera koji tumači tu naredbu.
- Prednosti interpretera:
  - relativno su jednostavni,
  - ne stvaraju izvršne verzije programa na računaru - štede memoriju.
- Mane interpretera:
  - sporost (puno se radilo na ovom planu, sve manja mana),
  - program se ne može tumačiti bez interpretera,
  - ne postoji mogućnost sakrivanja sopstvene programerske vještine.
- Poznati interpreteri su Java, MATLAB, PHP, Basic.

# Kompajleri

- Kompajleri prave izvršnu verziju programa (.exe fajl), koji predstavlja samostalnu aplikaciju (može se izvršavati bez dodatnog softvera).
- Kompajliranje je znatno složenije od interpretiranja.
- Prednosti kompajlera:
  - brzina izvršavanja,
  - sakrivanje programerske vještine,
  - finalni produkt je samostalan bez potrebe za isporučivanjem programa za tumačenje (kompajliranje).
- Mana kompajlera je složenost. Kompajleri su skupi programi, kreiranje jednog kompajlera je vrlo složen proces. Detaljno izučavanje procesa kompajliranja zahtjeva poseban kurs.
- Jezici čiji se programi kompajliraju su C, C++, Fortran, Pascal, Visual Basic.

# Podaci

- Elementarni podaci se realizuju hardverski. To su cijeli broj, realni broj i karakter.
- Složeni podaci su kolekcije elementarnih podataka.
- Tri osnovne karakteristike svakog tipa podataka su: domen, memorija koju zauzima i operacije koje se nad podatkom mogu vršiti.
- Na primjer, karakter se po ASCII kôdu kodira sa 8 bita. Potrebna memorija je 1 bajt. Domen je od 0 do 255 ako se podaci tumače kao cijeli broj bez predznaka, ili –128 do 127, ako se tumače kao označeni cijeli brojevi, dok su moguće operacije svojstvene karakteru (poređenje po abecedi itd.).

# Podaci

- Domen i memorija mogu biti mašinski zavisni (na jednom računaru ili jednom operativnom sistemu jedna veličina, dok na drugom druga veličina). Dobro napisani programi kontrolišu mašinski zavisne elemente.
- Osnovni primjer složenog podatka je **niz**.
- Na primjer, u programskom jeziku C int niz[40] predstavlja deklaraciju niza - direktivu računaru da zauzme memoriju za 40 cijelih brojeva, koji se nazivaju niz[0], niz[1], ..., niz[39].

# Operacije na računaru i u matematici

- Operacije koje se izvršavaju na računaru nijesu iste kao i matematičke operacije.
- Primjeri:
  - Pretpostavimo da se cijeli brojevi smještaju u registar od 8 bita. Neka prvi bit predstavlja predznak. Vrijednosti koje se mogu upisati u registru su tada od –128 do 127. U slučaju da sabirate 127 sa 1 nećete dobiti rezultat koji je jednak 128, već koliko? Uzeti u obzir hardversku predstavu cijelih brojeva.
  - U matematici se na sasvim prirodan način obavlja operacija dijeljenja brojeva. Tako je rezultat operacije 5/2 jednak 2.5. Kod većine viših programskih jezika, zbog načina na koji računar obavlja operacije, situacija je nešto drugačija.

# Operacije na računaru i u matematici

- Kod svih operacija računar pogleda kojeg su tipa operandi. Ako su istog tipa, pretpostavi da je i rezultat istog tipa i memoriju rezerviše za rezultat toga tipa i tumači rezultat kao podatak toga tipa. Tako, za 5/2, odnosno operaciju dijeljenja, rezerviše se memorija za cjelobrojni rezultat i dolazi do odsijecanja necjelobrojnog dijela i rezultat je, suprotno očekivanom, 2.

3+2 -> **operacija** je sabiranje, **operandi** su 3 i 2, a + je **operator** sabiranja

- U slučaju operacije nad podacima različitog tipa (npr. realni broj i cijeli broj) rezerviše se memorija za "veći" (složeniji) operand kao rezultat, pa 4.2/2 daje "pravilan" rezultat 2.1.
- U svim dobrim programskim jezicima korisnik može da utiče na rad operatora.
- Na ovu priču ćemo se vratiti kada budemo radili konverziju podataka.

# PROGRAMSKI JEZIK

# C

# Primjer C programa

- Dajmo primjer jednostavnog C programa:

```
#include <stdio.h>
int main() {
        int a, b;
        a = 3;
        b = 4;
        printf("Zbir brojeva %d i %d je %d", a, b, a+b);
}
```

- Program je krajnje jednostavan. Počinje sa funkcijom main(), u okviru koje je zauzeta memorija za dvije cjelobrojne promjenljive, kojima se nakon toga pridružuju vrijednosti, i na kraju se štampaju ta dva broja i njihov zbir. Programu prethodi dio koji se naziva pretprocesor i koji počinje sa znakom #. Program je uveden da bismo ilustrovali neke od sintaksnih elemenata sa sljedećeg slajda.

# Sintaksni elementi

1. Imena (identifikatori)
2. Konstante
3. Specijalni simboli
4. Rezervisane (ključne) riječi
5. Stringovi
6. Komentari
7. Bjeline

Imena (identifikatori) se koriste za imenovanje promjenljivih, funkcija i makroa. U programu na prethodnom slajdu imena promjenljivih su a i b, dok je printf ime funkcije. Osnovna pravila:

1. Imena se sastoje od slova, cifara i znaka podvlaka (underscore) _ , koja se tretira kao slovo.
2. Ime ne može početi cifrom.
3. Programski jezik C je case sensitive, odnosno razlikuje mala i velika slova. MM, mM, mm i Mm su različite promjenljive.
4. Ime ne može biti jednako rezervisanoj (ključnoj) riječi jezika.

U imenima ne može biti specijalnih simbola:
~!@#$%^&()[]{}.,;:'"\|/<>?+-*=

Poželjno je da ime odražava ulogu promjenljive!

# Konstante (literali)

- **Cjelobrojne konstante** (1, 12, -45). Konstante koje počinju 0 (nulom) predstavljaju brojeve zapisane oktalno, dok konstante koje počinju sa 0x predstavljaju heksadecimalne brojeve. Pošto se 0 ili 0x navode ispred brojeva nazivaju se prefiksi. Cjelobrojnim konstantama se mogu uvesti sufiksi (navode se nakon broja) L i U. L znači da se za cjelobrojnu konstantu ostavi više mjesta u memoriji (L potiče od long), dok U potiče od unsigned i znači da se konstanta u memoriji zapisuje kao neoznačeni cijeli broj (broj koji ne može imati predznak, odnosno ne može biti negativan).
- Na primjer, dozvoljene cjelobrojne konstante su 0x1AF ili 017U.
- U našem primjeru, konstante su 3 i 4 i pridružene su promjenljivim a i b.

# Konstante

- **Realne konstante** (konstante u pokretnom zarezu ili konstante tipa float –1.1, 0.234, 1.23e6, -.128). Kod ovih konstante se koriste sufiksi F za kratak memorijski zapis, L za dugačak memorijski zapis. Podrazumijevani format je double (kada nema sufiksa)

obratiti pažnju

- **Znakovne konstante** (konstante tipa char). Pod navodnicima **'c'** ili **'\ooo'**, gdje je **ooo** oktalni zapis broja koji predstavlja karakter po ASCII kodu, ili **'\xhhh'**, gdje je **hhh** heksadecimalni zapis karaktera.

Kosa crta \ najavljuje da karakteri koji slijede imaju specijalno značenje

# Konstante

- Neka od specijalnih karaktera nakon kose crte unazad '\' (eng. backslash) su:
  - **'\n'** - prelazak u novi red (koristićemo ga veoma često a korišćen je i u našem primjeru),
  - **'\t'** - tabulacija (kao taster tab koji se nalazi na tastaturi),
  - **'\v'** - je vertikalna tabulacija,
  - **'\b'** - povratak za jedan karakter unazad,
  - **'\r'** - prelazak na početak tekućeg reda,
  - **'\f'** - prelazak na novu stranicu,
  - **'\a'** - daje zvuk sa zvučnika računara (bip),
  - '\'', '\"' i '\\' - ove kombinacije redom znače apostrof, navodnike, ili samu kosu crtu (ovi simboli sami bez kose crte imaju specijalnu namjenu u programskom jeziku C).

# Specijalni simboli

- Već smo ih naveli kao nešto što ne može da se nalazi u okviru imena (identifikatora). Specijalni simboli:

**~ ! @ # $ % ^ & ( ) [ ] { } ; : ' " \ | < > ? . , + - / * =**

Zagrade koje se koriste redom za: davanje prioriteta operacija (ili poziv funkcije), indeksiranje nizova i matrica i za označavanje bloka naredbi.

Zagradni izrazi moraju biti korektni, što znači da svaka otvorena zagrada mora biti zatvorena i to po pravilnom redosljedu (prvo se zatvaraju unutrašnje pa spoljašnje)

pored značenja manje od i veće od, koriste se kod pretprocesora

oznake osnovnih matematičkih operacija

Sa ostalim specijalnim simbolima upoznaćemo se kasnije detaljno!!!

Identifikujte i diskutujte primjenu pojedinih simbola u našem primjeru!

# Rezervisane riječi

- Određeni skup riječi programski jezik C koristi za naredbe i najavu tipa podataka promjenljivih. Nazivaju se rezervisane ili ključne riječi.
- Imena (identifikatori) promjenljivih ne mogu biti ista kao ključne riječi.

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

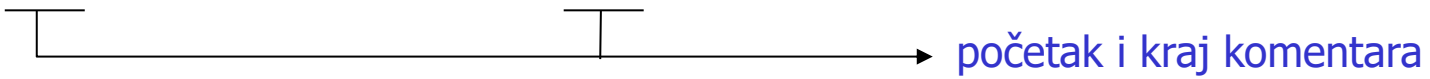
- Na primjer, programski jezik C koristi riječ int kao najavu cjelobrojne promjenljive. Nijedno ime u programu ne može biti int (može, recimo, int2, ali i to treba izbjegavati).
- Programski jezik C posjeduje **programske biblioteke** koje se po potrebi mogu uključivati u kôd programa. Ako je programska biblioteka uključena, sva navedena imena u njoj se ponašaju kao ključne riječi i ne mogu se koristiti za imena u našem kôdu. U našem primjeru je uključena programska biblioteka stdio.h, u kojoj je definisana funkcija printf.

# Stringovi

- Stringovi su nizovi karaktera navedeni unutar znaka navoda, npr. **"tekst"**.
- Dakle, pojedinačni karakteri se navode unutar apostrofa (npr. '#', '@'), a stringovi unutar znaka navoda (npr. "#@").
- Ako se unutar stringa nalazi kosa crta unazad, onda ono što slijedi za njom ima specijalno značenje (objašnjeno kod znakovnih konstanti).
- Analizirajte string koji je korišćen kao argument funkcije printf u našem primjeru.

# Komentar

- U programskom jeziku C, komentar se navodi unutar
  /* tekst komentara */




- Tekst unutar komentara ne utiče na izvršavanje programa!
- // - linijski komentari (važe u liniji u kojoj se nalaze)

**Komentarišite programe!**

- Nakon 2 mjeseca ni vi nećete znati što ste htjeli da uradite sa nekim dijelom kôda.
- Na početku programa navedite malo zaglavlje sa komentarom o namjeni programa, reviziji, datumu, ciljevima, opaženim problemima.
- Ne komentarišite očigledne stvari jer ćete se zamoriti i od nekog momenta prestati da komentarišete i neočigledne.

# Bjeline

- Bjeline se koriste radi poboljšavanja preglednosti programa. U pitanju su praznina (space), tabulacija (tab) i novi red.
- Bjeline se mogu kombinovati i svaka kombinacija bjelina se tretira kao jedna bjelina.
- Kod programskog jezika C (slično kao u MATLAB-u) ne poštuje se pravilo programske linije (npr. u FORTRAN-u), već se naredba ili iskaz može prostirati u više redova, ali se može i u jednom redu nalaziti više naredbi.
- Pravila kod bjelina:
  - Unutar osnovnog sintaksnog elementa ne smijemo upotrebljavati bjeline osim unutar stringa;
  - Između osnovnih sintaksnih elemenata može da stoji proizvoljno mnogo bjelina;