Programiranje 1

Predavanje 05 - Uvod u C, Unix i Windows okruženje. Primjeri programa kroz Code::Blocks

Matej Mihelčić

Prirodoslovno-matematički fakultet
Matematički odsjek

22. listopada 2023.

Uvod u programske jezike

Gruba podjela programskih jezika:

- **Strojni jezik - izvršni program**, instrukcije u **binarnom** kodu.
- **Asembler - izvorni program** (tekst kojeg treba prevesti), **prilagođen arhitekturi računala**. Tekst direktna zamjena i lako prevodljiv u **binarne instrukcije**.
- **Viši programski jezici - izvorni program**, program je tekst sastavljen od **naredbi**, jezik prilagođen posebnim zadaćama za koje je namijenjen (`C`, `FORTRAN`, `Pascal`, `C++`, `Java`, `C#`, `Perl`, `Python`, `R`, `Scala`, `Haskel`, `Kotlin` itd.).

Primjer strojnog jezika i Assemblera (8086)

Primjer: Intelov procesor 8086, 16-bitni procesor, instrukcije se pišu byte po byte.

Strojni jezik	Asembler kod	objašnjenje
00011110	`PUSH DS`	stavi podatak DS na stog
00101011	`SUB AX, AX`	oduzmi AX od AX i spremi u AX
11000000		
01010000	`PUSH AX`	spremi AX u memoriju
10111000	`MOV AX, MYDATA`	kopiraj podatke iz MYDATA u AX
00000001		
10001110	`MOV DS, AX`	kopiraj podatke iz AX u DS
11011000		

Mane strojnog jezika i asemblera

Osnovne mane:

- **Instrukcije** strogo određene arhitekturom računala (kod **nije prenosiv** na računala drugačijeg tipa).
- Pisanje programa je **jako neefikasno**.
  - Instrukcije su **jednostavne**, prilagođene računalu a ne zadatku.
  - Programi su **dugački** i **nepregledni**.
  - Programi podložni **greškama** koje se teže pronalaze.
- Asembler je bolji jer se binarni kodovi **supstituiraju** mnemoničkim kodom (pregledniji i lakše pamtljiv zbog tekstualnih imena instrukcija, registara i podataka). Ostale mane **u potpunosti** dijeli sa strojnim jezikom.

Kada koristiti asembler?

Asembler se obično koristi kod specifičnih zadataka vezanih uz **upravljanje** hardverom. Omogućava korištenje punog potencijala hardvera, ostvarenje najviše razine optimizacije.

- Korištenje cache memorije, posebnog instrukcijskog skupa procesora:
  - Npr. omogućava stvaranje brže aplikacije za reprodukciju zvuka.
  - Omogućava redukciju veličine izvršnih datoteka.
- Omogućava visoku razinu optimizacije pri programiranju specifičnih uređaja za specifične zadatke (npr. FPGA, GPU).
- Može se koristiti i u sklopu matematičkih biblioteka, npr. `oneAPI MKL`.

Generalniji zadaci koji zahtjevaju **portabilnost**, brži i sigurniji razvoj, jednostavnije održavanje i nadogradnju se programiraju u **višim** programskim jezicima (i dalje mogu biti **jako** optimizirani).

Prednosti viših programskih jezika

- Neovisnost o arhitekturi računala (prenosivost). Kod nekih programskih jezika do te mjere da niti ne treba ponovo prevoditi program prije izvršavanja na različitim arhitekturama ili operacijskim sustavima.
- Prilagođenost većem broju zadataka, nude široki spektar tipova, omogućavaju kreiranje novih, naredbe prilagođene tipovima i operacijama nad njima.
- Naredbe su složenije, bliže ljudskom načinu razmišljanja, ovisno o programskom jeziku mogu biti visokog nivoa.
- Brži i jednostavniji razvoj programa koji rješavaju neki problem.
- Jednostavnije održavanje i nadogradnja.

Viši programski jezici

Programe kod viših programskih jezika treba prevesti iz **izvornog koda** u **izvršni kod**. Prevođenje radi poseban program koji se zove **prevoditelj** (eng. compiler).

- Prednost: **prenosivost** - program se može izvršiti na bilo kojem računalu koje ima prevoditelj za određeni jezik.
- Mana: **prevođenje** je bitno **složenije** nego kod asemblera. Programski jezici imaju **stroga pravila** (gramatike).

Programski jezik C

C je viši programski jezik opće namjene.

- Autor: **Dennis Ritchie** (Bell Telephone Laboratories).
- Razvijen: 1972 - 1973 godine.
- Svrha: pisanje jezgre operacijskog sustava UNIX.
- Ideja: ostvariti maksimalnu portabilnost UNIX-a na razne vrste računala. Zato je za implementaciju korišten posebno osmišljen viši jezi a ne strojni jezik.

Programski jezik C - osnovna svojstva

Početna svrha razvoja jezika utječe na njegova svojstva. C je razvijen za programiranje sistemskih programa.

- C je jezik **relativno niskog nivoa**, blizu arhitekture računala.
- C **operira s istim objektima kao i većina računala** (znakovi, brojevi, adrese - preko pokazivača ili pointera).
- C **podržava sve operacije** nad tim tipovima koje su **podržane arhitekturom računala**: aritmetičke, logičke, relacijske (usporedbe), posebne operacije na bitovima (npr. pomak - eng. shift).

C ima i karakteristike viših jezika:

- **Grupiranje naredbi** (tzv. blokove), **složene naredbe** za kontrolu toka (petlje, uvjetne naredbe).
- **Izvedene** ili **složene tipove podataka** poput polja, struktura, datoteka.

Programski jezik C - osnovna svojstva

- Mogućnost razdvajanja programa u **manje cjeline** koje mogu biti smještene u **različitim** datotekama.
- Manje programske cjeline (potprograme). U C-u realizirani kao funkcije.
  - Funkcije mogu **vratiti vrijednosti** svih **osnovnih** tipova i **nekih složenih** tipova (strukture).
  - Funkcije se mogu **rekurzivno** pozivati.

C ima **standardiziranu programsku biblioteku** koja sadrži **sve strojno ovisne** elemente jezika. Sastoji se od dva dijela:

- **Funkcije** za interakciju s okolinom (operacijskim sustavom).
  - Čitanje i pisanje datoteka.
  - Formatirani ulaz i izlaz.
  - Alokacija memorije.
  - Operacije sa znakovima i stringovima, itd.
- Skup standardnih zaglavlja (eng. header files) - omogućavaju korištenje standardnog skupa funkcija i tipova podataka.

Programski jezik C - opis i standardi

Programski jezik C je opisan u knjizi:

Brian W. Kernighan i Dennis M. Ritchie, *The C Programming Language*, Prentice Hall, New Jersey, 1978.

Standardizacija C-a je započela 1989. godine od strane organizacije: **American National Standard Institute** (ANSI).

Standard je uveo znatne promjene u jezik. Osnovna pravila jezika (gramatika) su **znatno postrožena** što je **olakšalo prevođenje i otkrivanje grešaka**.

Novi standard se naziva ANSI C i opisan je u knjizi:

Brian W. Kernighan i Dennis M. Ritchie, *The C Programming Language (second edition)*, Prentice Hall, Upper Saddle River, New Jersey, 1988.

Programski jezik C - opis i standardi

ANSI C je implementiran u svim modernim C-prevoditeljima. ANSI standard je 1990. godine usvojila i **Međunarodna organizacija za standarde** (ISO) čime nastaje ISO C. ANSI/ISO standard se skraćeno zove C90.

ISO prihvaća novi standard C-a 1999. godine. Taj standard uvodi manje dopune u standard C90. Time dobivamo standard C99.

ISO prihvaća novi standard C-a 2011. godine pod imenom C11.

- Uvođenje detaljnog modela memorije za podršku **istovremenog** (**paralelnog**) izvršavanja pojedinih dijelova programa (eng. *multiple threads of execution)*.
- Standardizacija novih funkcionalnosti. Poboljšanje sigurnosti jezika.
- Eliminiranje nekih teže ostvarivih dijelova C99 standarda.

Programski jezik C - opis i standardi

Posljednji standard programskog jezika C se počeo razvijati 2017. godine a dovršen je 2018. godine, stoga se naziva C17 a neki ga nazivaju i C18. Zastavice prevodioca `-std=c17` i `-std=c18` obje označavaju istu verziju C17.

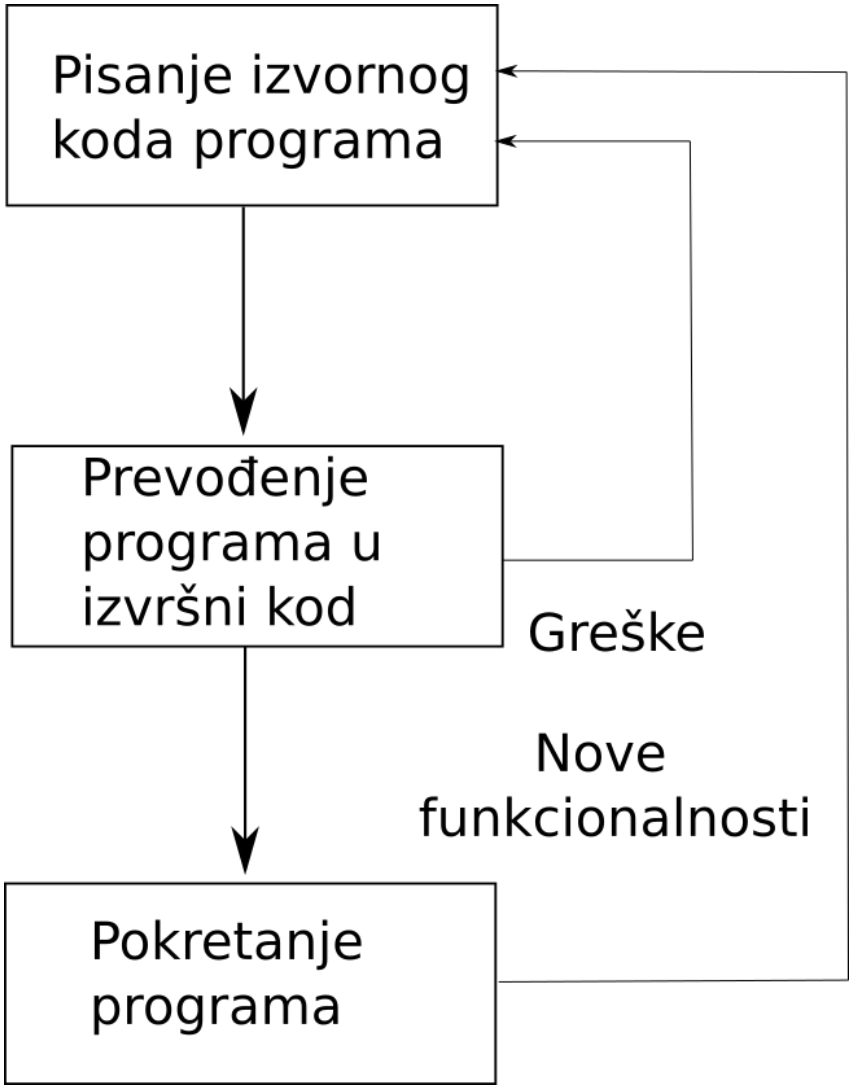
Glavni cilj standarda C17 je **popravak pogrešaka** iz standarda C11. **Nisu uvedene** nove značajke u programski jezik C.

Koristimo C90 i dijelove C99 standarda koji dozvoljavaju miješanje deklaracija varijabli i naredbi. **Nije dozvoljeno** korištenje polja varijabilne duljine.

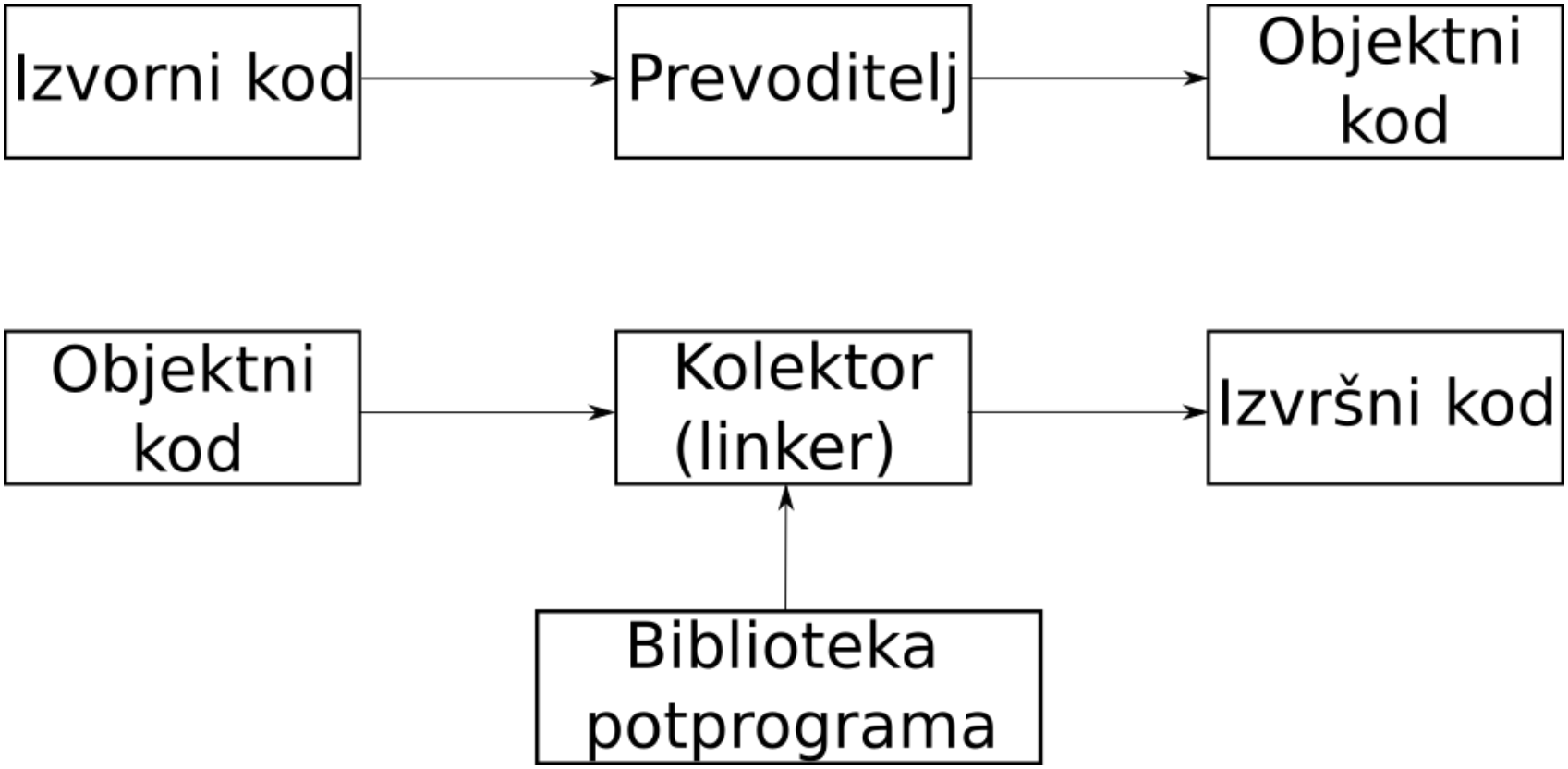
Opći postupak pisanja programa

- **Izvorni kod** programa se **upiše u tekstualnu datoteku** s ekstenzijom `.c` ili `.h` za zaglavlje.
- **Poziva se prevoditelj** koji **transformira izvorni kod u izvršni kod**. Izvršna datoteka ima ekstenziju `.exe` na `Windows` i `.out` na `UNIX` operacijskom sustavu.
- **Pozivom izvršne datoteke** se pokreće program.
- Tijekom prevođenja često dolazi do grešaka koje treba popraviti, nakon čega se ponovo mora pozvati prevoditelj.

Opći postupak pisanja programa



Shematski opis prevođenja



Poziv prevoditelja u UNIX okruženju

- **Prevođenje** izvornog koda `prvi.c` i **pokretanje** programa.

```
gcc prvi.c
./a.out
```

- **Zadavanje imena** izvršne datoteke i pokretanje programa.

```
gcc prvi.c -o prvi
./prvi.out
```

- Istovremeno **prevođenje više datoteka**.

```
gcc prvi.c drugi.c treci.c -o svi
./svi.out
```

Alati za programiranje u UNIX okruženju

- **Editor teksta** - ovisno o distribuciji UNIX-a imamo: `vi`, `pico`, `nano`, `gedit`. Služe za kreiranje i pregledavanje tekstualnih datoteka (uključuje izvorne datoteke).
- **Prevoditelj** - `cc`, `gcc`. Prevoditeljli imaju brojne mogućnosti koje zadajemo **opcijama**.
- **Povezivač** ili **kolektor** (eng. linker, loader) - `ld`. Povezuje objektne datoteke s programskim bibliotekama u izvršni kod.
- **Programske biblioteke** - uključivanje se vrši `-l` opcijom (npr. `-lm` za **matematičke funkcije** iz `math.h`).
- Postoji **priručnik** za opcije i alate (naredbe `man` ili `?`).

Primjeri: `man cc`, `man vi`, `man scanf`.

Osnove rada u Windows okruženju

U Windows okruženju možemo raditi na dva načina:

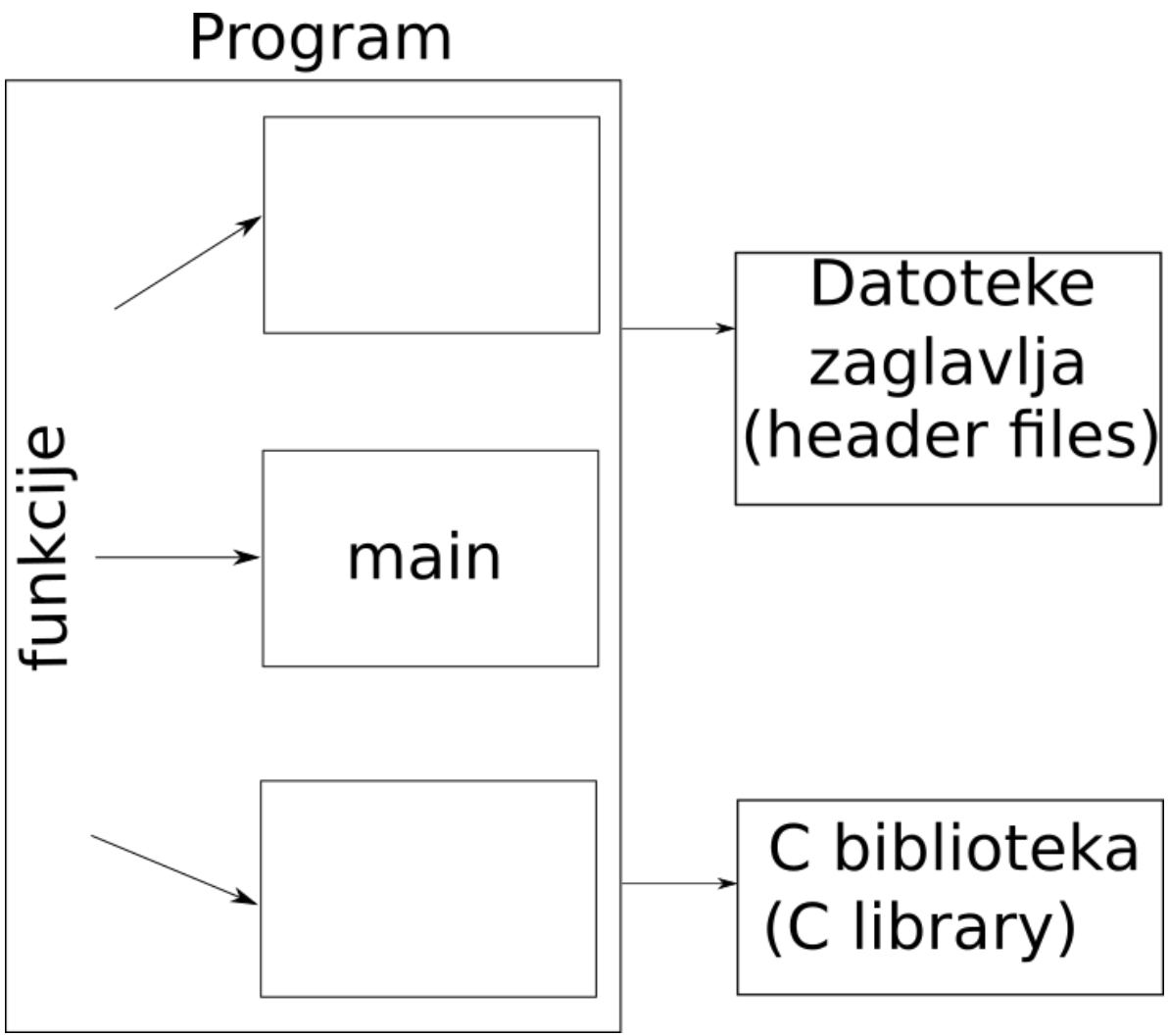
1. Možemo koristiti **komandni prozor** (eng. command prompt) - pišemo naredbe operacijskom sustavu. Potrebni su:
   - Tekst-editor (npr. Notepad++)
   - C prevoditelj
   - Razvojna podrška s linkerom i C-bibliotekom.
2. Možemo koristiti i **integriranu razvojnu okolinu** koja omogućuje obavljanje svih prije navedenih poslova kroz isti razvojni alat (program).
   Primjeri:
   - Visual Studio - baziran na Microsoftovom C prevodiocu i pripadnoj razvojnoj podršci (može se dobiti akademska licenca za korištenje osnovne verzije).
   - Code::Blocks - okolina bazirana na MinGW varijanti GNU C prevodioca za Windows. Besplatan za sve, koristi prilično nov gcc, vrlo ugodan i jednostavan za rad (**pripaziti na HR znakove, specijalne znakove**).

Opća struktura C programa

C program se sastoji od imenovanih blokova koji se nazivaju **funkcije**.
Glavni program = funkcija `main` (fiksno ime koje znači *glavni*).
Opis bloka:

- Započinje znakom {, a završava znakom }.
- Blok obuhvaća ili sadrži: deklaracije/definicije, naredbe i neimenovane blokove.
- Svaka definicija/deklaracija i naredba mora završavati znakom ; (točka zarez je kraj naredbe).
- Blok ne završava točkom zarez (;).

Opća struktura C programa



Primjer programa u C-u

```
1   #include <stdio.h>
2  //ukljucujemo standardno zaglavlje za ulaz-izlaz
3  //linijski komentari uvedeni standardom C99
4  /*blokovski dostupni i ranije*/
5    int main(void){ /*glavna funkcija programa*/
6    /*zove se main, povratna vrijednost je cijeli
7    broj s predznakom. Nema ulazne parametre -
8    oznaka void*/
9          printf("Dobar␣dan!\n"); /*Dobar dan!*/
10    /*kursor pozicioniran na pocetak novog reda*/
11     return 0;
12   }
```

Kako prevesti i pokrenuti ovaj program?

UNIX:

- Utipkati tekst programa u tekstualni editor (`gedit`, `nano`, `pico`) i spremiti datoteku kao tekstualnu datoteku s `.c` ekstenzijom. Npr. `prvi.c`.
- Pozvati *C* prevoditelj naredbom `gcc prvi.c`. a) Prevoditelj prevodi program u **objektni kod**, b) poziva linker koji uključuje standardnu biblioteku i kreira `izvršni kod` u datoteci `a.out`.
- Program **izvršavamo** pozivom `./a.out`.

Code::Blocks (Windows, UNIX, MAC):

- Odabrati `File->New empty file`
- Utipkati izvorni kod programa i spremiti s ekstenzijom `.c`. Npr. `prvi.c`.
- Može i `File ->New from template`
- Izabrati `C/C++ source`, `-> Go -> Next -> C -> Next -> Upisati ime datoteke -> Finish`

Isprobajte program:

Primjer programa u C-u

```
1 #include <stdio.h>
2
3 int main(void){
4   printf("Dobar␣");
5   printf("dan.");
6   printf("\n");
7
8   return 0;
9 }
```

Primjeri programa u C-u

Napišite program koji:

- učitava dva **cijela** broja a i b (tipa `int`),
- računa vrijednost izraza $3a^2 - b$ i sprema tu vrijednost u varijablu c,
- ispisuje vrijednost varijable c.

Primjer programa u C-u

```
1  #include <stdio.h>
2
3  int main(void){
4    int a,b,c;
5    scanf("%d%d", &a, &b);
6    c = 3*a*a-b;
7    printf("Rezultat␣=␣%d\n",c);
8
9    return 0;
10 }
```

Prethodni program možemo napisati i tako da **odmah ispišemo** vrijednost izraza, **bez** spremanja u varijablu c.

Primjer programa u C-u

```
1  #include <stdio.h>
2
3  int main(void){
4    int a,b;
5    scanf("%d%d", &a, &b);
6    printf("Rezultat␣=␣%d\n",3*a*a-b);
7
8    return 0;
9  }
```

Primjeri programa u C-u

Napišite program koji:

- učitava dva **realna** broja x i y (tipa `double`),
- računa vrijednost izraza $2x^2 - y^3$ i sprema tu vrijednost u varijablu z,
- ispisuje vrijednost varijable z.

Primjer programa u C-u

```
1  #include <stdio.h>
2
3  int main(void){
4    double x,y,z;
5    scanf("%lg%lg", &x, &y);
6    z = 2*x*x-y*y*y;
7    printf("Rezultat␣=␣%g\n",z);
8
9    return 0;
10 }
```

Primjeri programa u C-u

Primjetite razliku u formatima kod učitavanja **cijelih brojeva s predznakom** (`%d`) i **realnih brojeva dvostruke preciznosti** (`%lg` ili `%lf`). Postoji i razlika u **ispisu** (`%d`) za cjelobrojni tip a (`%g`) za realni (`float` i `double`).

Oprez: realni brojevi jednostruke preciznosti se učitavaju koristeći format `%g` ili `%f` (tip `float` u C-u).

Što se događa ukoliko pokušamo učitati `double` koristeći format `%g` ili `%f`? Broj se upisuje kao `float` u prva 4 byte-a memorijske lokacije rezervirane za spremanje vrijednosti tipa `double`. Na zadnja 4 byte-a imamo slučajne vrijednosti. Kada se taj broj interpretira kao `double` (recimo kod ispisa), rezultat je **bitno različit** od željenog!

Ovisno o postavkama zastavica prevodioca, može se dogoditi da dobijemo **upozorenje** ili da ne dobijemo nikakvu poruku ukoliko pokušamo učitati vrijednost tipa `double` koristeći format `%g`.

Zastavice prevodioca

Korištenjem zastavica: `-Wall -Wextra -pedantic` dobijemo detaljnija upozorenja.