Programiranje 1
5. predavanje

Saša Singer

singer@math.hr
web.math.pmf.unizg.hr/~singer

PMF – Matematički odsjek, Zagreb

Sadržaj predavanja

- Uvod u programski jezik C:
  - Malo povijesti i svrha jezika C.
  - Postupak pisanja programa.
  - Osnovna struktura programa.
  - Osnove rada u Unix/Linux okruženju.
  - Osnove rada u Windows okruženju — Code::Blocks.
- Primjeri programa kroz Code::Blocks:
  - Prvi program — "Hello world".
  - Primjer 2 — učitaj, izračunaj, ispiši (`int`).
  - Primjer 3 — učitaj, izračunaj, ispiši (`double`).

Uvod u programski jezik C

Sadržaj

- Uvod u programski jezik C:
  - Malo povijesti i svrha jezika C.
  - Postupak pisanja programa.
  - Osnovna struktura programa.
  - Osnove rada u Unix/Linux okruženju.
  - Osnove rada u Windows okruženju — Code::Blocks.
  - Jednostavni primjeri programa.

Uvod u programske jezike

Gruba podjela programskih jezika:

- Strojni jezik — izvršni program, instrukcije u binarnom kôdu.
- Asembler — izvorni program (tekst kojeg treba prevesti), jezik je prilagođen arhitekturi računala.
  Tekst je direktna zamjena za binarne instrukcije i lako se prevodi u njih.
- Viši programski jezici — izvorni program, program je tekst sastavljen od naredbi, jezik je prilagođen posebnim zadaćama za koje je jezik namijenjen. Najpoznatiji jezici:
  - C, FORTRAN, Pascal, C++, Java, Perl, Python, ...

Primjer strojnih instrukcija i Assemblera (8086)

Primjer s prastarog Intelovog procesora 8086 za IBM XT.
(To je 16 bitni procesor, instrukcije se pišu byte po byte).

Strojni jezik	Ekvivalentan asemblerski kôd
00011110	PUSH DS
00101011	SUB AX, AX
11000000	
01010000	PUSH AX
10111000	MOV AX, MYDATA
00000001	
10001110	MOV DS, AX
11011000	

Mane strojnog jezika i asemblera

Osnovne mane takvih programa:

- Instrukcije su strogo određene arhitekturom računala. Posljedica: programi se ne mogu prenijeti na računalo drugačijeg tipa.
- Pisanje programa je izrazito neefikasno.
  - Instrukcije su vrlo elementarne, prilagođene stroju, a ne poslu kojeg treba napraviti.
  - Rezultat su dugački, nepregledni programi.
  - Podložno greškama, koje se teško otkrivaju.
- Asembler je samo malo humaniji od strojnog jezika (zamjena binarnog kôda tzv. mnemoničkim kôdom — tekstualnim imenima instrukcija, registara i podataka).

Svrha programskih jezika

Stvarna svrha programiranja u asembleru je samo za:

- specifične zadaće vezane uz upravljanje hardwareom.

Programer tad ima punu kontrolu nad svim komponentama računala. Na primjer, asembler se koristi za

- dobivanje maksimalne brzine na specifičnim dijelovima arhitekture (sklopovi za "paralelno" računanje, cache),

u sklopu posebnih matematičkih biblioteka (poput MKL).

Za sve ostale, ljudima "bliže" poslove koriste se tzv. viši programski jezici.

- "Viši" = bliži čovjeku, a ne stroju.

Viši programski jezici

Postoji ih gomila. Najpoznatiji jezici:

- C, FORTRAN, Pascal, C++, Java, Perl, Python, ...

Osnovne prednosti nad asemblerom:

- Neovisnost o arhitekturi računala (prenosivost).
- Prilagođenost pojedinim specifičnim zadaćama (naredbe su prilagođene tipovima podataka i operacijama nad njima — u odgovarajućoj primjeni).
- Složenije naredbe, bliže ljudskom načinu mišljenja.

Rezultat: Veća efikasnost programiranja, tj.

- brže i jednostavnije obavljanje posla.

Viši programski jezici (nastavak)

Program u takvom jeziku prvo treba prevesti

- iz izvornog kôda (engl. source code)
- u izvršni kôd (engl. executable code).

Ovaj postupak, obično, ide u više koraka (v. malo kasnije).

- Posao prevođenja radi posebni program — prevoditelj (engl. compiler) za dani jezik.

Prednosti i mane:

- Prenosivost — program se može (barem u principu) izvršiti na bilo kojem računalu koje ima prevoditelj za određeni jezik.
- Prevođenje je bitno složenije nego kod asemblera. Zato programski jezici imaju svoja stroga pravila (gramatike).

Programski jezik C

C je viši programski jezik opće namjene.

- Autor: Dennis Ritchie (Bell Telephone Laboratories).
- Razvijen sedamdestih godina prošlog stoljeća.
- Osnovna svrha:
  - Pisanje jezgre operacijskog sustava UNIX.
- Ideja:
  - maksimalna portabilnost UNIXa na razne vrste računala (nadalje pišem Unix — bolje izgleda).
- Zato je jezgra sustava napisana
  - u posebno smišljenom višem jeziku,
  - a ne u strojnom, za neko posebno računalo.

Programski jezik C — osnovna svojstva (1)

Početna svrha — razvoj sistemskih programa, uvelike određuje "izgled" jezika.

- C je jezik relativno "niskog nivoa", ne jako daleko od arhitekture računala ("high level Assembler").
- To znači da C operira s istim objektima kao i većina računala:
  - znakovima, brojevima, adresama (pointerima ili pokazivačima).
- C podržava sve operacije na tim podacima koje su podržane arhitekturom računala:
  - aritmetičke, logičke, relacijske (usporedbe), ali i
  - posebne operacije na bitovima, poput pomaka (engl. "shift").

Programski jezik C — osnovna svojstva (2)

S druge strane, kao viši jezik, C ima:

- grupiranje naredbi (tzv. blokove), složene naredbe za kontrolu toka (petlje, uvjetne naredbe),
- izvedene ili složene tipove podataka, poput polja, struktura, datoteka,
- mogućnost razbijanja programa u manje cjeline, koje mogu biti smještene u raznim datotekama,
- manje programske cjeline (potprograme) — u C-u su to funkcije,
  - koje mogu vratiti vrijednosti svih osnovnih tipova i nekih složenih tipova (strukture),
  - i mogu se rekurzivno pozivati.

Programski jezik C — osnovna svojstva (3)

Na kraju, C ima standardiziranu programsku biblioteku koja sadrži sve strojno ovisne elemente jezika. Sastoji se iz dva bitna dijela.

- Niz funkcija za interakciju s okolinom (operacijskim sustavom), poput
  - čitanja i pisanja datoteka,
  - formatirani ulaz i izlaz,
  - alokaciju memorije,
  - operacije sa znakovima i stringovima, itd.
- Skup standardnih zaglavlja (tzv. "header files" ili, skraćeno, "headers") koji uniformiraju pristup
  - deklaraciji funkcija i tipova podataka.

Programski jezik C — opis i standardi (1)

Opis jezika dan je u knjizi (KR)

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

I jezik C i operacijski sustav Unix brzo se šire sedamdesetih i osamdesetih godina prošlog stoljeća.

- American National Standard Institute (ANSI) pristupa standardizaciji C-a, koja je dovršena 1989. godine.

Novi standard uveo je značajne izmjene u jezik.

- Osnovna pravila jezika (gramatika) su znatno stroža.

Posljedica: Lakše prevođenje i otkrivanje grešaka.

Programski jezik C — opis i standardi (2)

Za razliku od prvotne verzije, novi standard često se naziva ANSI C. Opisan je u knjizi (KR2)

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

Implementiran je u svim modernim C–prevoditeljima.

- ANSI standard usvojila je i Međunarodna organizacija za standarde (ISO) 1990. godine (tzv. ISO C).

Ovaj ANSI/ISO standard skraćeno zovemo C90 standard.

- Godine 1999. ISO je prihvatio novi C standard, koji uvodi manje dopune u C90 standard.

Skraćeni naziv: C99 standard.

Programski jezik C — novi standard C11

Nakon dugih priprema, relativno nedavno,

- 2011. g., ISO je prihvatio novi C standard, neformalno poznat kao C11, punim imenom ISO/IEC 9899:2011.

Osnovne promjene:

- Standardizacija dodatnih mogućnosti koje su već neko vrijeme bile dostupne u većini prevoditelja.
- Uvođenje detaljnog modela memorije, za podršku istovremenog izvršavanja pojedinih dijelova programa (engl. "multiple threads of execution").
- Neke stvari propisane u C99 standardu postale su opcionalne (tj. neobavezne), jer se teško realiziraju — većina prevoditelja ih još ne podržava korektno.

Programski jezik C — novi standard C11

Napomena: na mom webu je link na

- završni prijedlog C11 standarda iz travnja 2011. godine, zvan N1570 (javno dostupan).

Sam standard se plaća!

Da ne bude zabune,

- mi koristimo C90 — i to onaj dio koji radi svagdje,
- a na Prog2 spomenut ćemo jedan dodatak iz C99.

Postupak pisanja programa

Opći postupak pisanja programa

Postupak "programiranja" u programskom jeziku C pod raznim operacijskim sustavima vrlo je sličan.

Bitni koraci:

- Prvo se tekst programa, tj. izvorni kôd, upiše u neku tekstualnu datoteku. Standardne ekstenzije su `.c` ili `.h`, za zaglavlja. Na primjer, u datoteku `prvi.c`.
- Zatim se poziva program prevoditelj (C compiler) koji transformira (prevodi) napisani program u izvršni kôd. Kao rezultat dobiva se izvršna datoteka (standardne ekstenzije su `.out` na Unixu, ili `.exe` na Windowsima).
- Pozivom te datoteke izvršava se program.

Opći postupak pisanja programa (nastavak)

Napomena. U pisanju programa redovito se javljaju greške, pa treba dodati:

- Nalaženje i ispravljanje grešaka,

što, obično, rezultira ponavljanjem prethodnih koraka, sve dok program ne "proradi"!

Ovo iznad (više–manje) vrijedi

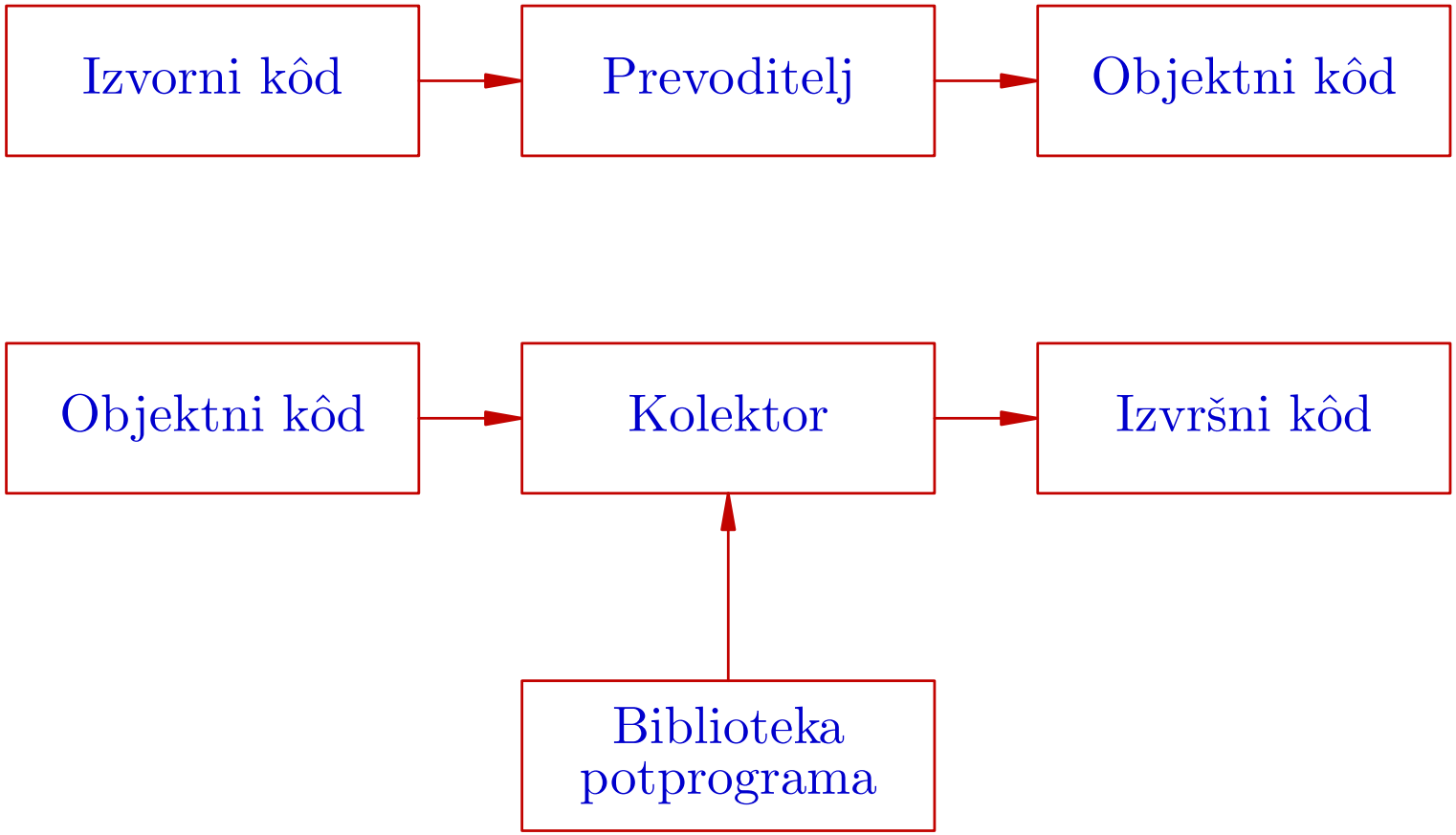
- i za Unix (identično za Linux), i za MS Windows.

Samo se imena programa koje pozivamo (koristimo) u pojedinim fazama razlikuju.

U nastavku, malo više o tome kako se to radi u pojedinom okruženju — Unix/Linux, Windows.

Opći postupak pisanja programa (nastavak)

Shematski to izgleda ovako:



Postupak pisanja programa u Unix okruženju

- Napisati izvorni kôd (engl. source code). On se sastoji od jedne ili više
  - programskih datoteka (`.c`),
  - datoteka zaglavlja (`.h`).
- Pozvati C prevoditelj (compiler) koji prevodi svaku `.c` datoteku u tzv. objektni kôd, kojeg sprema u pripadnu `.o` datoteku.
- Linker (povezivač, kolektor) povezuje sve potrebne `.o` datoteke i programske biblioteke u izvršni kôd.
- Ako mu se ne naredi suprotno,
  - prevoditelj sam poziva linker,
  - linker generira izvršnu datoteku `a.out`

Primjeri rada u Unix okruženju

Poziv prevoditelja i zadavanje imena pojedinih datoteka.

- Tekst programa je u `prvi.c`:

```
cc prvi.c
./a.out
```

- Zadavanje imena izvršne datoteke.

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

- Istovremeno prevođenje više datoteka.

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

Alati za programiranje u Unix okruženju (1)

- Editor teksta
  - standardni `vi`,
  - ili neki drugi, na pr. `pico`.

  Svrha: kreiranje tekst datoteka, poput `ime.c`, `ime.h`.
- Prevoditelj (compiler)
  - `cc`, `gcc`.

  Napomena: prevoditelj ima brojne mogućnosti koje zadajemo tzv. opcijama.
- Povezivač ili kolektor (engl. linker ili loader)
  - `ld`.

  Svrha: povezivanje objektnih datoteka s programskim bibliotekama u izvršni kôd.

Alati za programiranje u Unix okruženju (2)

- Programske biblioteke
  - uključivanje se vrši `-l` opcijom.

  Na primjer: `-lm`, za matematičke funkcije iz `math.h`.

Ovo je bila samo ilustracija osnovnih naredbi za pisanje C programa u Unix okruženju. Opis je vrlo daleko od potpunog.

Za detaljniji opis, ili kad sve ostalo zakaže, postoji i ...

- Priručnik (engl. manual) za razne stvari (i opcije)
  - `man`.

  Primjeri:
  - `man cc` za opcije prevoditelja,
  - `man vi` za standardni editor,
  - `man scanf` za standardne funkcije iz C biblioteke.

Code::Blocks okolina za Windows okruženje

Osnove rada u Windows okruženju

U Windows okruženju možemo standardno raditi na dva načina (samo prividno — dosta različita).

Možemo koristiti komandni prozor (engl. command prompt)

- u kojem pišemo komande operacijskom sustavu,

vrlo slično kao u Unix okruženju. Treba nam (kao i tamo):

- Tekst–editor, tj. uređivač običnog teksta (ne Word),
- C prevoditelj
- i razvojna podrška s linkerom i C–bibliotekom.

Primjer. Tako ja standardno koristim Intelov C compiler.

Napomena. Intelov C/C++ compiler je besplatan za studente.

Osnove rada u Windows okruženju (nastavak)

Možemo koristiti i tzv. integriranu razvojnu okolinu, koja omogućava

- obavljanje svih poslova kroz isti razvojni alat (program).

Primjeri:

- MS Visual Studio, baziran na Microsoftovom C compileru i pripadnoj razvojnoj podršci (biblioteka, linker).

Besplatan za studente (u osnovnoj varijanti). Može raditi i s Intelovim compilerom.

Mana: to je ogroman paket (čak i u osnovnoj verziji).

Alternativa:

- Code::Blocks okolina, koja se zasniva na tzv. MinGW varijanti GNU C compilera (`gcc`) za Windowse.

Code::Blocks okolina za Windows okruženje

Prednosti Code::Blocks okoline:

- besplatna za sve (link je na webu kolegija),
- trenutna verzija koristi prilično novi `gcc`.
- vrlo ugodna i jednostavna za rad.

Mane:

- relativno velika za skidanje.

No, alternative su još mnogo veće (osim prastarog DevC++).

Code::Blocks je instaliran u svim praktikumima (nadajmo se).
Oprez u praktikumima:

- Pazite na naša slova na tipkovnici, pri kucanju programa!

Posebno, na razne zagrade i specijalne znakove.

Prije prvog primjera — što dalje?

Vrijeme je da napravimo prvi "pravi" program u C-u, tj.

- napišemo tekst programa, prevedemo ga i izvršimo.

Međutim, već rekosmo da C ima

- stroga gramatička pravila (tzv. sintaksa)

po kojima se piše program.

Zato, i prije prvog primjera, treba nešto osnovno reći o izgledu ili "strukturi" programa.

Za početak, bitno je pogledati

- globalnu strukturu programa — na tzv. najvišoj razini (za nas, kao programere).

Bez toga je teško napisati bilo kakav program!

Prije prvog primjera — što dalje? (nastavak)

Zatim ćemo napraviti nekoliko primjera programa,

- s osnovnim opisom pojedinih dijelova,
- bez svih detalja i pravila,

tek toliko da negdje počnemo, tako da

- možemo pisati i izvršavati osnovne programe.

Sve što ovdje ilustriramo bit će detaljnije obrađeno kasnije.

A onda, "nema spasa". Moramo

- detaljno opisati sve stvari od "dna", tj. od najniže razine.

Što to znači?

Prije prvog primjera — što dalje? (nastavak)

Program u C-u je tekst, koji se sastoji od znakova i riječi.

Dakle, počinjemo od toga

- koji znakovi se mogu koristiti u C programu,
- kako se pravilno tvore "veće" cjeline ("riječi")
- i koja su njihova značenja (tzv. semantika).

Tek onda prelazimo na složenije dijelove jezika C:

- kako se pišu deklaracije i naredbe.

Idemo redom. Prvo o globalnoj strukturi programa.

Opća struktura C programa

Grubo govoreći, C program se sastoji od

- imenovanih blokova, koji se nazivaju funkcije.

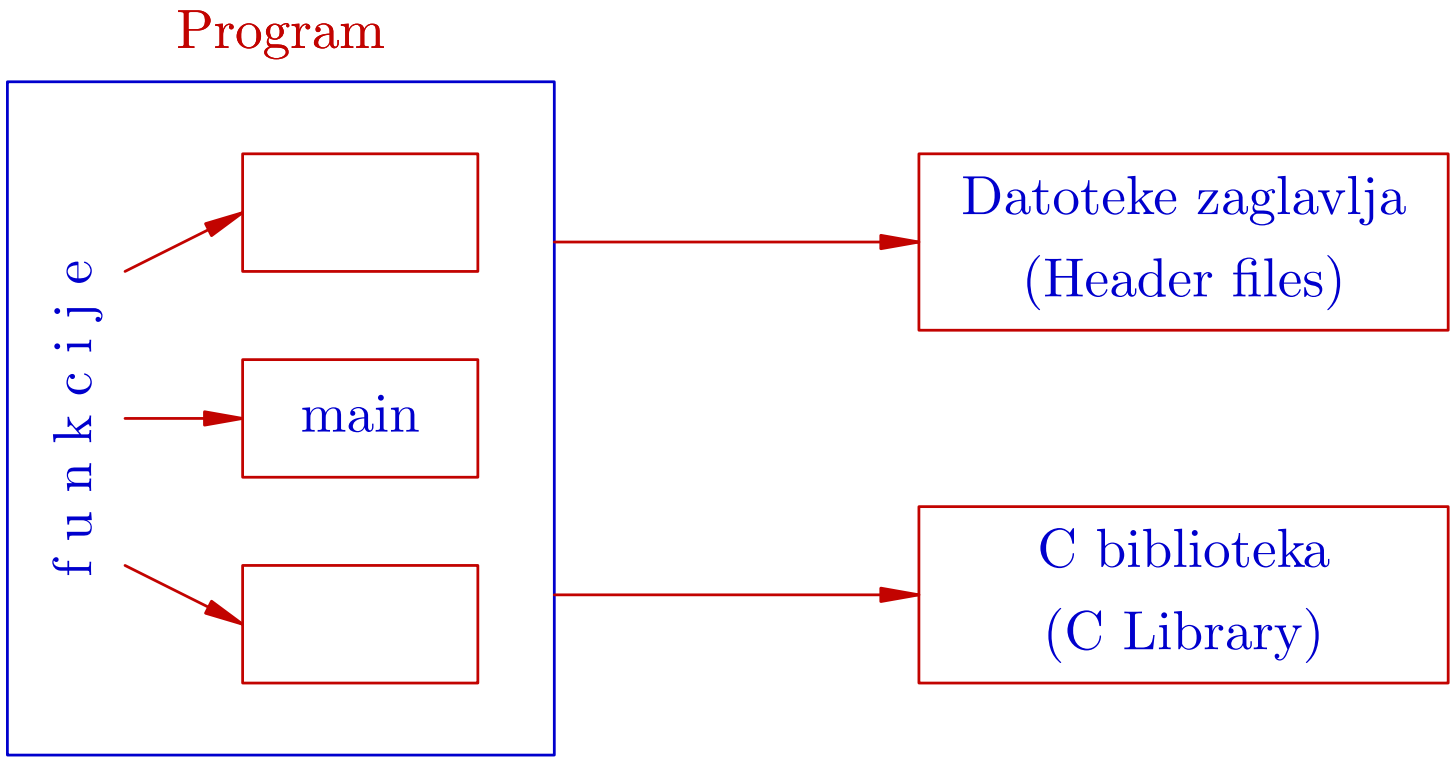
"Glavni" program = funkcija `main` (fiksno ime — "glavni").

Osnovni opis bloka:

- Blok započinje znakom `{` , a završava znakom `}`.
- Blok obuhvaća, ili sadrži:
  - deklaracije/definicije, naredbe i neimenovane blokove.
- Svaka definicija/deklaracija i naredba mora završavati znakom `;` (tj. točka-zarez je kraj, a ne separator).
- Blok ne završava znakom `;` ,
  - tj. iza znaka `}` za kraj bloka, ne piše se `;` .

Opća struktura C programa (nastavak)

Opći izgled C programa i njegova veza s okolinom:



Primjeri programa kroz Code::Blocks

Napomena — predavanja i dodatak

Napomena. Ovdje su napisani

- samo tekstovi programa, bez dodatnih objašnjenja!

Detaljna objašnjenja napisana su u dodatku ovog predavanja.

Većinu toga ću ispričati, prolazeći kroz programe, ali unutar Code::Blocks okoline.

- Ako sam u Code::Blocks-u, onda je teško istovremeno pokazivati i ostatak teksta!

Prvi program — "Hello world"

Primjer 1. Standardni prvi C program u većini knjiga izgleda (otprilike) ovako:

```
#include <stdio.h>

  /* Glavni program - funkcija main. */

int main(void)
{
    printf("Dobar dan.\n");
    return 0;
}
```

Što radi ovaj program?

Prvi program — svrha

Iskreno, ništa jako pametno:

- ispisuje tekst `Dobar dan.` na standardni izlazni uređaj.

Sjetite se, svaki program (algoritam) mora imati neki izlaz.

- Naš program ima samo to i ništa više!

Dakle, to je (skoro) "najmanji" mogući program:

- napiši zadani tekst.

Jedini "višak" u programu je komentar (v. malo kasnije).

Program je vrlo jednostavan, ali potpun, u smislu da se može

- korektno prevesti i izvršiti,

bez grešaka!

Prvi program — Unix okruženje

Pod Unixom, treba napraviti sljedeće:

- Utipkati tekst programa (u nekom editoru) i spremiti ga u neku datoteku — recimo, `prvi.c`,
- Pozvati C prevoditelj (recimo, `cc`) naredbom
  - `cc prvi.c`
- Prevoditelj prevodi program u objektni kôd, sam poziva linker koji uključuje standardnu biblioteku i kreira izvršni kôd u datoteci `a.out` (jer nismo drugačije rekli).
- Program izvršavamo tako da utipkamo naredbu
  - `./a.out`
- Rezultat izvršavanja je (prema očekivanju) ispis poruke
  - `Dobar dan.`

Prvi program — Code::Blocks

Na Windowsima, ako želimo raditi u Code::Blocks okolini,

- prvo treba startati Code::Blocks.

Zatim, treba redom:

- Odabrati File (na vrhu), pa New — Empty file, jer želimo utipkati tekst novog programa.
- Otvorit će se prozor za unos teksta programa, u kojeg treba utipkati tekst programa.
- Kad ste gotovi, vrlo je zdravo spremiti taj tekst u neku datoteku:
  - File, Save file, izaberite mapu i ime datoteke.

  Na primjer, `prog_1.c`.

Prvi program — Code::Blocks (nastavak)

Ako odmah želite "obojani" tekst (tzv. syntax highlighting), onda postupak ide ovako:

- Odabrati File (na vrhu), pa New — File ....
- U prozoru New from template treba redom:
  - Izabrati (kliknuti) C/C++ source, pa Go,
  - onda Next (ili isključite tu stranicu),
  - izabrati C, pa Next,
  - upisati puni put i ime datoteke, ili preko ... izabrati mapu i upisati ime datoteke, pa Finish.
- Sad (na)pišete program, a spremite ga ovako:
  - File, Save file.

Prvi program — Code::Blocks ('odi tamo)

- Tu sam u Code::Blocks-u i pričam po dodatku.

Prvi program — još malo

Zadatak. Probajte što radi prvi program kad izbrišemo `\n` na kraju stringa u pozivu funkcije `printf`.

Zadatak. Sljedeći program radi isto kao i prvi. Probajte!

```
#include <stdio.h>

int main(void)
{
    printf("Dobar ");
    printf("dan.");
    printf("\n");
    return 0;
}
```

Primjer 2 — učitaj, izračunaj, ispiši (int)

Primjer 2. Napišite program koji

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

Ovo je ponešto kompliciraniji program od prvog, jer sadrži ulaz podataka, računanje izraza i ispis rezultata.

Tekst programa spremljen je u datoteci `prog_2.c`.

Drugi program — tekst

```
#include <stdio.h>

int main(void)
{
    int a, b, c;

    scanf("%d%d", &a, &b);

    c = 3 * a * a - b;

    printf(" Rezultat = %d\n", c);

    return 0;
}
```

Drugi program — izvršavanje i rezultat

Kad pokrenemo program u Code::Blocks, otvori se komandni prozor u kojem se ništa ne događa!

- U stvari, program uredno radi, ali čeka nas da upišemo vrijednosti za `a` i `b`.

Zato je vrlo korisno, prije svakog čitanja, ispisati neki tekst koji kaže što se od nas očekuje (v. treći program).

Kad (na ulazu) napišemo niz znakova:

- `3␣2` i stisnemo `ENTER`,

dobivamo izlaz (opet niz znakova):

- `Rezultat = 25`

Nevjerojatno, ali radi! Provjerite!

Ulaz je u datoteci `prog_2.in`, a izlaz u `prog_2.out`.

Drugi program — još malo

Zadatak. Program možemo napisati i tako da odmah ispišemo vrijednost izraza, bez spremanja u varijablu c.

```
#include <stdlib.h>

int main(void) {
    int a, b;

    scanf("%d%d", &a, &b);
    printf(" Rezultat = %d\n", 3 * a * a - b);

    return 0;
}
```

Primjer 3 — učitaj, izračunaj, ispiši (double)

Primjer 3. Napišite program koji

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

Osnovna razlika između ovog i prethodnog programa je u tipu podataka s kojim radimo. Tamo su bili cijeli brojevi, a ovdje su realni.

Sve ostalo je vrlo slično!

Tekst programa spremljen je u datoteci `prog_3.c`.

Treći program — tekst

```
#include <stdio.h>

int main(void)
{
    double x, y, z;

    printf(" Upisi x i y:\n");
    scanf("%lg %lg", &x, &y);

    z = 2 * x * x - y * y * y;

    printf(" Rezultat = %g\n", z);
```

Kad nema žute crte na kraju, nastavak je na sljedećoj stranici!

Treći program — tekst (nastavak)

```
    return 0;
}
```

Jedina stvarna razlika obzirom na prethodni program je

- u oznakama konverzije za formatirano čitanje i pisanje.

Zato obratite pažnju na ta mjesta u programu.

Treći program — izvršavanje i rezultat

Kad pokrenemo program, prvo se ispiše poruka

- `Upisi x i y:` s prijelazom u novi red.

Zatim program čeka da upišemo vrijednosti za `x` i `y`.
Ako napišemo niz znakova:

- `3.0␣2.0` i stisnemo `ENTER`,

dobivamo izlaz:

- `Rezultat = 10`

Oznaka konverzije `%g` ne piše nepotrebne nule i decimalnu točku, pa rezultat izgleda kao cijeli broj.

Probajte neke druge vrijednosti na ulazu!

Ulaz je u datoteci `prog_3.in`, a izlaz u `prog_3.out`.

Treći program — još malo (čitanje)

Oprez s oznakom konverzije za čitanje realnih brojeva:

- `%g` — služi za čitanje vrijednosti tipa float,
- `%lg` — služi za čitanje vrijednosti tipa double.

Nemojte zaboraviti slovo `l` kod čitanja za `double`!

Što se dogodi ako zaboravimo slovo `l`?

- Pristojan prevoditelj se pobuni s porukom — ako ste ga "zamolili" da javlja sve što može. Inače, može i "šutiti"!

Na pr., `gcc` u Code::Blocks, uz `-Wall -Wextra -pedantic`,

- javi: 0 errors, 2 warnings,
- s vrlo urednim opisom što ga "smeta".

Nemojte ignorirati te poruke, čak ni upozorenja!

Treći program — još malo (pisanje)

Ako ipak izvršimo takav program,

- čita se `float` i sprema u prva 4 bajta na zadanoj adresi, a ne na svih 8 bajtova.

Dobijemo "svašta" na zadnja 4 bajta!

- Rezultati su slučajni.

Pogledati: `prog_4.c`, ulaz `prog_4.in`, izlaz `prog_4.out`.

Kod pisanja nema te opasnosti, ali pazite na oznaku!

Oznaka konverzije za formatirano pisanje realnih brojeva:

- `%g` (a ne `%lg`) — služi i za `double` i za `float`.

(Tip `float` se pretvara u `double`.)