Programiranje (C)
2. predavanje

Saša Singer

singer@math.hr
web.math.hr/~singer
www.math.hr/~singer

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

Pregled programskog jezika C na jednostavnim primjerima programa.

- Osnovni elementi jezika kroz primjere:
  - Nekoliko primjera programa.
  - Struktura C programa.
  - Osnovni tipovi podataka.
  - Osnovne naredbe jezika.

Sve što ovdje ilustriramo bit će detaljnije obrađeno kasnije.

Prvi potpuni program — Hello world

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

```
#include <stdio.h>

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

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

- korektno prevesti i izvršiti,

bez grešaka!

Prvi potpuni program (nastavak)

Da bismo to napravili, recimo 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 otipkamo naredbu
  - `./a.out`

Prvi potpuni program (rezultat)

- Rezultat izvršavanja programa je (prema očekivanju) ispis poruke
  - `Dobar dan.`

Struktura C programa

C program sastoji se od funkcija i varijabli. Funkcije sadrže instrukcije koje određuju koje će operacije biti izvršene. Varijable služe pamćenju podataka u memoriji računala.

- Izvršavanje programa počinje funkcijom `main`. Funkcija s tim imenom `main` ("glavna") mora biti prisutna u svakom programu.
- Svaki objekt (funkcija, varijabla) koji koristimo u programu mora biti korektno deklariran prije upotrebe.
- To vrijedi i za sve funkcije iz standardne biblioteke koje trebamo u programu.
- Nama treba funkcija `printf` za izlaz (pisanje) podataka i moramo nekako navesti pripadnu deklaraciju.

Struktura C programa (nastavak)

- Zato program započinjemo naredbom

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

- Ta naredba uključuje (engl. `include`) u program datoteku `stdio.h` koja sadrži deklaraciju funkcije `printf` (i mnogih drugih funkcija za ulaz/izlaz).
- Datoteke s nastavkom `.h` nazivaju se datoteke zaglavlja (engl. header files).
- Navođenje imena datoteke između znakova `<` i `>` kaže da se radi o standardnoj datoteci zaglavlja (koja dolazi uz C prevoditelj), a ne o "našoj".

Struktura C programa (nastavak)

- Sljedeća linija programa je deklaracija funkcije `main`

```
int main(void)
```

- Funkcija može imati jedan ili više argumenata (parametara) i obično vraća neku vrijednost.
- U našem primjeru `main` nema niti jedan argument. To se deklarira tako da se u oble zagrade `(` i `)` (gdje inače deklariramo argumente) stavi ključna riječ `void` (engl. void = prazan).
- Tip povratne vrijednosti u našem primjeru je cijeli broj, što je označeno s `int` na početku, ispred imena funkcije.
- Iza deklaracije funkcije dolazi tzv. tijelo funkcije.

Struktura C programa (nastavak)

- Tijelo funkcije sastoji se od deklaracija objekata i naredbi zatvorenih unutar vitičastih zagrada `{` i `}`.
- U našem primjeru nema deklaracija, već tijelo sadrži samo dvije naredbe
  - poziv funkcije `printf` — za ispis stringa,
  - naredbu `return`.
- Naredbe završavaju znakom točka–zarez `;`.
- Izvršavanje funkcije završava naredbom `return`.
- Vrijednost koju funkcija vraća navodi se u `return` naredbi.
- U našem primjeru, funkcija `main` vraća nulu, pa je zadnja naredba `return 0;`.

Struktura C programa (nastavak)

Napomene.

- Funkcija `main` vraća operacijskom sustavu cjelobrojnu vrijednost koja ima značenje izlaznog statusa programa.
- Nula se interpretira kao uspješni završetak, a svaka druga vrijednost kao završetak usljed greške.
- Funkcija `printf` nakon završenog ispisa ne prelazi u novi red. To se postiže specijalnim znakom `\n` ("newline" znak).
- Specijalni znakovi se pišu tako da počinju znakom `\`.

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

Prvi program još jednom

Program koji radi isto kao prvi:

```
#include <stdio.h>

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

Primjer 2 — komentari, varijable, while petlja

Primjer 2. Program treba ispisati prvih deset faktorijela.

$$1! = 1,$$
$$n! = n \cdot (n-1)!, \quad n \geq 2.$$

```
#include <stdio.h>

int main(void)
{

    /* Deklaracija varijabli */

    int n, fakt;
```

Primjer 2 (nastavak)

```
    /* Izvrsne naredbe: */

    n = 1;                      /* pridruzivanje */
    fakt = 1;                   /* pridruzivanje */
    while (n <= 10)             /* while petlja */
    {
        fakt = fakt * n;
        printf(" %d  %d\n", n, fakt);
        n = n + 1;
    }
    return 0;
}
```

Tipovi podataka

Osnovni tipovi podataka:

`char`	jedan znak,
`short`	"kratki" cijeli broj,
`int`	cijeli broj,
`long`	"dugi" cijeli broj,
`float`	realan broj jednostruke preciznosti,
`double`	realan broj dvostruke preciznosti.

Pored ovih osnovnih tipova podataka, postoje još

- pokazivači (kao posebni osnovni tip)

i složeni tipovi podataka kao što su

- polja, strukture, unije.

Relacijski operatori

Relacijski operatori (operatori uspoređivanja):

operator	primjer	značenje
`<=`	`a <= b`	manje ili jednako,
`>=`	`a >= b`	veće ili jednako,
`<`	`a < b`	strogo manje,
`>`	`a > b`	strogo veće,
`==`	`a == b`	jednako,
`!=`	`a != b`	nejednako (različito).

Važno: razlikujemo

- operator pridruživanja (`=`) i
- relacijski operator jednakosti (`==`).

Aritmetički operatori

Aritmetički operatori:

	operator	primjer
zbrajanje	`+`	`a + b`
oduzimanje	`-`	`a - b`
množenje	`*`	`a * b`
dijeljenje	`/`	`a / b`

Oprez: Operacija dijeljenja kad su oba operanda cjelobrojna daje cjelobrojan rezultat `3/2 = 1` (operacija div iz UuR).

Ako je bar jedan operand realan broj, dijeljenje je uobičajeno dijeljenje realnih brojeva, tj. `3.0/2.0 = 1.5`.

Operator `%` daje cjelobrojni ostatak (operacija mod iz UuR), tj. `5 % 3 = 2`.

While petlja

```
while (uvjet)
    naredba;
```

Dok je `uvjet` ispunjen (različit od nule) izvršava se naredba. Kad uvjet više nije ispunjen (jednak nuli), prelazi se na sljedeću naredbu.

While petlja može obuhvaćati i grupu naredbi unutar `{` i `}`.

```
while (uvjet)
{
    naredba_1;
    naredba_2;
    ...
}
```

For petlja

Ispis faktorijela (iz Primjera 2) pomoću `for` petlje:

```
#include <stdio.h>
int main(void)
{
    int n, fakt;
    fakt = 1;
    for (n = 1; n <= 10; n = n + 1)
    {
        fakt = fakt * n;
        printf(" %d  %d\n", n, fakt);
    }
    return 0;
}
```

Usporedba for i while petlje

```
for (n = 1; n <= 10; n = n + 1)
{
    naredbe;
}
```

je ekvivalentno s

```
n = 1;
while (n <= 10)
{
    naredbe;
    n = n + 1;
}
```

Operatori inkrementiranja i dekrementiranja

Naredbe oblika `n = n + 1` i `n = n - 1` često se sreću kod povećavanja ili smanjivanja brojača u petljama.

C ima posebne operatore `++` i `--` za te dvije operacije.

`n = n + 1` je ekvivalentno s `n++`,

`n = n - 1` je ekvivalentno s `n--`.

Umjesto `n++` može se koristiti `++n`.
Umjesto `n--` može se koristiti `--n`.

Napomena: Ovo vrijedi samo za brojače u `for` petlji.

Općenito, `n++` i `++n` ne znače isto! Točno značenje je:

- `n++` = prvo iskoristi vrijednost od `n`, a onda ju povećaj,
- `++n` = prvo povećaj vrijednost od `n`, a onda ju iskoristi.

For petlja s operatorom inkrementiranja

```
for (n = 1; n <= 10; ++n)
{
    fakt = fakt * n;
    printf(" %d  %d\n", n, fakt);
}
```

Zadatak. Promijenite ove programe tako da ispisuju prvih 20 faktorijela. Pogledajte rezultate i objasnite ih. Ponovite prikaz i aritmetiku cijelih brojeva iz UuR.

Primjer 3 — čitanje, for, realni brojevi

Primjer 3. Program treba izračunati učitati broj prirodni broj n i izračunati realni broj

$$\sum_{k=1}^{n} \frac{1}{k(k+1)}.$$

```
#include <stdio.h>

int main(void)
{
    int n, k;
    double suma;
```

Primjer 3 (nastavak)

```
    printf("Unesite broj n: n= ");
    scanf(" %d", &n);

    suma = 0.0;
    for (k = 1; k <= n; k++)
    {
        suma = suma + 1.0 / (k * (k + 1));
    }

    printf("Suma prvih %d clanova = %f\n",
           n, suma);
    return 0;
}
```

Čitanje podataka

Za čitanje podataka upotrebljava se funkcija `scanf`.

```
#include <stdio.h>
    ...
    int n;
    ...
    scanf(" %d", &n);
```

Ovo učitava vrijednost varijable `n`, po znaku konverzije `%d`.

Bitno: uočite da drugi argument od `scanf` nije `n`, već `&n`.

Adresni operator `&` daje adresu varijable `n` na koju `scanf` treba spremiti učitanu vrijednost.

Prijenos parametara po vrijednosti — vrijednost stvarnog parametra se kopira u "lokalnu" varijablu za funkciju.

Osnovni znakovi konverzije

Znakovi konverzije za čitanje i pisanje brojeva:

`%d`	–	učitavanje i ispisivanje cijelih brojeva,
`%f`	–	ispisivanje brojeva tipa `float` i `double`,
`%f`	–	učitavanje brojeva tipa `float`,
`%lf`	–	učitavanje brojeva `double`.

Zadatak. Izvršite ovaj program za razne vrijednosti od n. Probajte nekoliko uzastopnih vrijednosti takvih da je $n \geq \sqrt{2^{31} - 1}$. Pažljivo pogledajte rezultate. Da li monotono rastu? Objasnite ponašanje rezultata (aritmetika cijelih brojeva u nazivniku)! Kako se program jednostavno može "popraviti"?

Primjer 4 — if naredba

Primjer 4. Treba napisati program koji rješava kvadratnu jednadžbu $ax^2 + bx + c = 0$. Koeficijenti a, b i c se učitavaju.

Rješenja su općenito kompleksna i dana su formulom

$$z_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

C nema kompleksne brojeve (osim kao strukture). Zato koristimo notaciju $z_1 = x_1 + iy_1$, $z_2 = x_2 + iy_2$, i računamo realni i imaginarni dio brojeva z_1 i z_2.

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

Primjer 4 (nastavak)

```
/* Rjesavanje kvadratne jednadzbe
      a x^2 + b x + c = 0.          */

int main(void)
{
    double a, b, c, /* Koeficijenti          */
           d,       /* Diskriminanta         */
           x1, x2,  /* Realni dijelovi.      */
           y1, y2;  /* Imaginarni dijelovi. */

    printf("Upisite koeficijente a, b, c: ");
    scanf ("%lf%lf%lf", &a, &b, &c);
```

Primjer 4 (nastavak)

```
y1 = 0.0;
y2 = 0.0;

if (a != 0.0) {
    d = b*b - 4*a*c;
    if (d > 0) {
        x1 = (-b + sqrt(d)) / (2 * a);
        x2 = (-b - sqrt(d)) / (2 * a);
    }
    else if (d == 0) {
        x1 = -b / (2 * a);
        x2 = x1;
    }
```

Primjer 4 (nastavak)

```
        else {
            x1 = -b / (2 * a);          x2 = x1;
            y1 = sqrt(-d) / (2 * a);  y2 = -y1;
        }
    }
    else {
        printf("Jednadzba nije kvadratna.\n");
        exit(-1);
    }
    printf("z1=%f + i*(%f), z2=%f + i*(%f)\n",
           x1, y1, x2, y2);
    return 0;
}
```

Datoteke zaglavlja

`stdio.h`	–	koristi se radi funkcija `printf` i scanf,
`stdlib.h`	–	koristi se za funkciju `exit`,
`math.h`	–	biblioteka matematičkih funkcija potrebna zbog `sqrt` ($\texttt{sqrt}(x) = \sqrt{x}$).

Znak konverzije `%lf` služi za učitavanje brojeva tipa `double`.

Za lakše pamćenje, uzimite da je `double` = `long float`.

If naredba — uvjetno grananje

3 razna oblika `if` naredbe (naredbe uvjetnog grananja):

```
if (uvjet)        if (uvjet)        if (uvjet1)
    naredba1          naredba1;         naredba1;
                  else              else if (uvjet2)
                      naredba2;         naredba2;
                                    else
                                        naredba3;
```

Zadatak. Preuredite ovaj program tako da numerički stabilnije računa realne korijene (pogledati UuR). Apsolutno veći korijen računamo po formuli, a apsolutno manji iz Viètinih formula.