# 0. Osnove programskog jezika C – ponavljanje

U uvodnom poglavlju ponovit ćemo sve pojmove i naredbe s kojima smo se susretali u drugom razredu tijekom učenja programiranja u programskom jeziku C.

## 0.1. Zapisi algoritama

Algoritam je konačan skup jednoznačno definiranih, logički povezanih koraka (postupaka). Ima jasno definirane ulazne i izlazne veličine te mora biti primjenjiv za rješavanje određenog tipa problema, a ne samo jednog konkretnog problema.

Algoritme možemo zapisivati na više načina. **Zapisi algoritama** mogu biti:

**a)** u obliku teksta
**b)** pseudokodom
**c)** s pomoću dijagrama toka (grafički)
**d)** zapis u programskom jeziku.

### Svojstva algoritama

**a)** Algoritam za problem na koji se primjenjuje mora davati točan izlaz za sve njegove dozvoljene ulaze.
**b)** Algoritam ne smije biti beskonačan – do rješenja problema mora se doći u konačnom broju koraka.
**c)** Svaki algoritam mora imati točno definirano što su mu ulazne veličine, a što izlazne veličine.
**d)** Naredbe u algoritmu moraju biti jasne, jednoznačno definirane i ne smiju biti dvosmislene.

Sva navedena svojstva koja smo naveli za algoritme primjenit ćemo i u odabranom programskom jeziku. Algoritme koje smo zapisivali u obliku pseudokoda ili s pomoću dijagrama toka pretvaramo u naredbe. Programe ćemo pisati u obliku koda prema definiranim pravilima programskog jezika. Za naše potrebe pisanja programa, odabrali smo programski jezik C. Prije pisanja programa uvijek trebamo razmisliti kako bi glasio algoritam kojim možemo riješiti određeni problem. Nakon osmišljavanja algoritma, algoritam pretvaramo u smisleno poredane naredbe koje zadajemo računalu. Računalo te naredbe pretvara u strojni kod te izvršava tražene naredbe.

## 0.2. Programski jezik C – ponavljanje

**Programski jezik C** viši je programski jezik opće namjene. Primjenjiv je na većini današnjih računala, vrlo je sličan drugim programskim jezicima, ali ima bolje mogućnosti povezivanja sa sklopovljem računala. Zbog svoje dobre strukture, široke primjene i mogućnosti koje pruža, naše prvo programersko iskustvo započeli smo u ovom programskom jeziku. **Autor programskog jezika C** je **Dennis Ritchie** (slika 0.1).

**Slika 0.1.** Dennis MacAlistair Ritchie (1941. – 2011.), američki računalni znanstvenik, autor programskog jezika C

**Osnovna struktura programa** u programskom jeziku C je:

1. pretprocesorske naredbe
2. funkcija `main` (početak)
3. tijelo funkcije `main`
   a) deklaracija podataka
   b) upis podataka
   c) obrada podataka
   d) ispis podataka
4. kraj funkcije `main`

U C-u komentare možemo pisati na 2 načina:

1. **`//komentar`**
2. **`/*komentar*/`**

**Osnovni definirani jednostavni tipovi podataka** u C-u su:

a) `char` – znakovni tip podatka, zapis cijelih brojeva
b) `short` – "kratki" cijeli broj (pokriva manji raspon cijelih brojeva)
c) `int` – cijeli broj
d) `long` – "dugi" cijeli broj (pokriva veći raspon cijelih brojeva)
e) `float` – decimalan broj jednostruke preciznosti
f) `double` – decimalan broj dvostruke preciznosti.

Riječ **varijabla** (engl. *variable* – promjenljiva vrijednost) **predstavlja niz određenog broja uzastopnih lokacija**, a njihov broj ovisi o tipu podatka. Tijekom izvršavanja programa **može promijeniti svoju vrijednost**.

Svaku **varijablu** moramo obavezno **najaviti (deklarirati) prije korištenja u programu** na način da joj damo **ime** te da joj **dodijelimo memorijski prostor** gdje će se spremiti njezina vrijednost.

**Varijable** se deklariraju na sljedeći način:

```
tip_podatka lista_varijabli;
```

**Konstante** za razliku od varijabli u programu zadržavaju istu vrijednost (ne mijenjaju se), a deklararaju se na sljedeće načine:

**a)**
```
const  tip_konstante  naziv_konstante = pridruzivanje_vrijednosti;
```

**b)**
```
#define ime_konstante vrijednost_konstante
```

**Konstante** mogu biti: **a)** cjelobrojne konstante
**b)** realne konstante
**c)** znakovne konstante.

**Funkcija za ispis podataka**:

```
printf("format", lista_varijabli);
```

| `printf("%3d", 5);` | `printf("%4.2f", 1.2);` | `printf("%4c", 'A');` |
|---|---|---|
| 3 – broj mjesta koja se zauzimaju za ispis<br>d – oznaka formata (ispis cijelog broja)<br>5 – broj koji se ispisuje | 4 – ukupan broj mjesta koja se zauzimaju za ispis<br>2 – broj decimalnih mjesta<br>f – oznaka formata (ispis decimalnog broja)<br>1.2 – broj koji se ispisuje | 4 – ukupan broj mjesta koja se zauzimaju za ispis<br>c – oznaka formata (ispis znaka)<br>A – znak koji se ispisuje |

**Funkcija za upis podataka** s tipkovnice:

```
scanf("format", &adresa_varijable);
```

**Naredba za pridruživanje izraza nekoj varijabli**:

```
varijabla = izraz;
```

## Operatori

Razlikujemo aritmetičke (tablica 0.1.), relacijske (tablica 0.2.) i logičke (tablica 0.3.) operatore.

**Tablica 0.1.** Aritmetički operatori

| Aritmetički operator | Značenje |
|---|---|
| + | zbrajanje |
| – | oduzimanje |
| * | množenje |
| / | dijeljenje (operator ovisi o tipu operanda) |
| % | cjelobrojno dijeljenje s ostatkom |

**Tablica 0.2.** Relacijski operatori

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

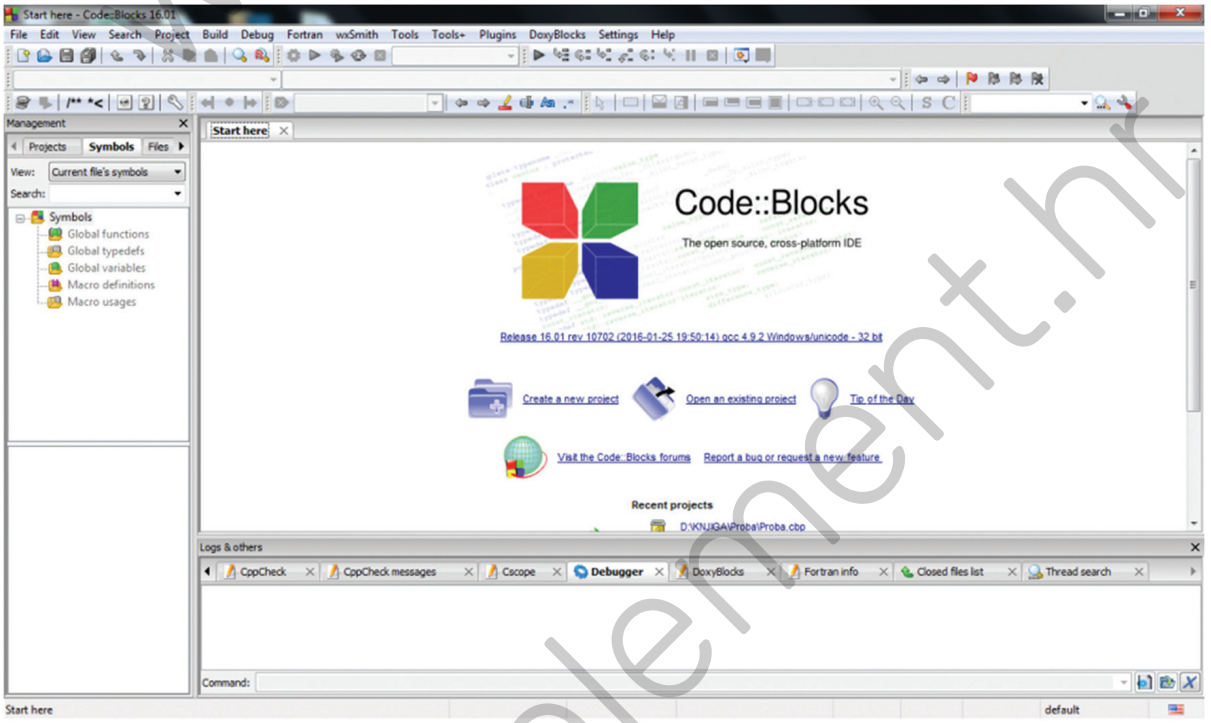
**Tablica 0.3.** Logički operatori

| Logički operator | Značenje |
|---|---|
| && | I / AND |
| \|\| | ILI / OR |
| ! | NE / NOT |

## 0.2.1. Razvojno okružje Code::Blocks i pokretanje prvog programa

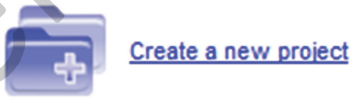
Za programiranje u C-u odabrali smo **razvojno okružje Code::Blocks.**

Dizajn razvojnog okružja Code::Blocks prikazan je slikom 0.2.



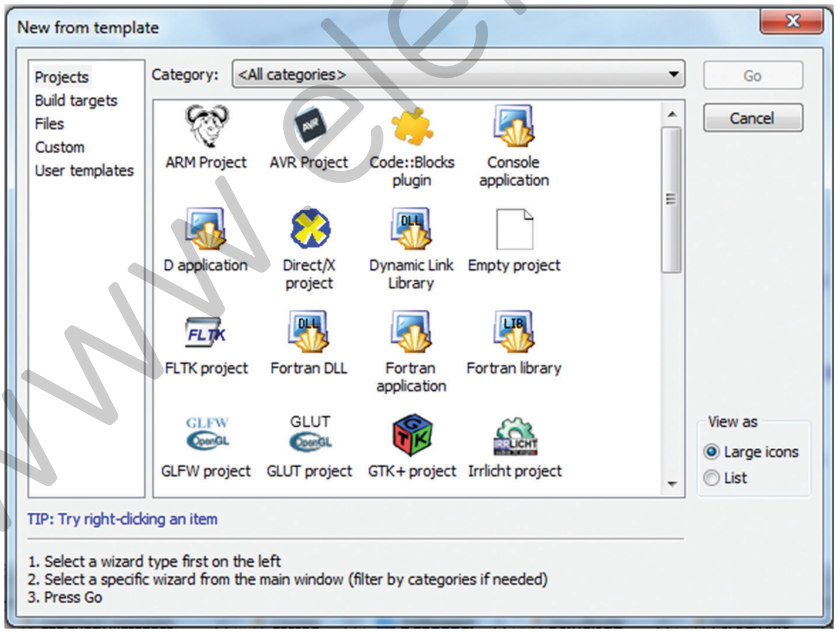

**Slika 0.2.** Razvojno okružje Code::Blocks

Ako želimo doći do editora teksta u koji ćemo upisati naš prvi "Hello world!" program, najprije moramo kreirati novi projekt. Slika 0.3. prikazuje ikonu za kreiranje novog projekta.




**Slika 0.3.** Kreiranje novog projekta

Nakon što smo kliknuli na *Create a new project* dobivamo prozor prikazan slikom 0.4.

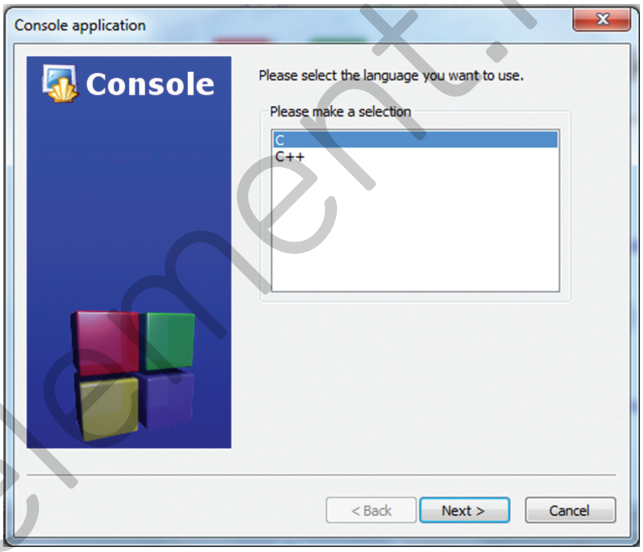



**Slika 0.4.** Odabir projekta

Potom odabiremo *Console application* ikonu prikazanu slikom 0.5.

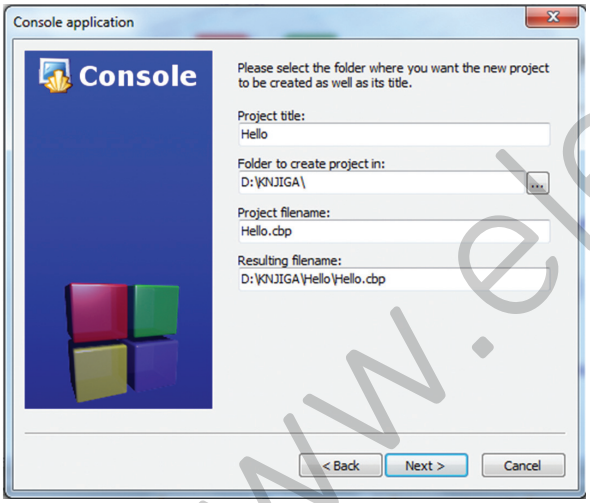
Odabiremo C kako je prikazano na slici 0.6.


**Slika 0.5.** Console application

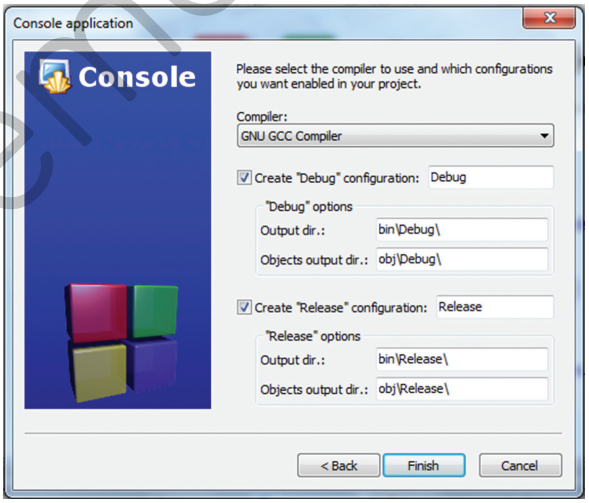



**Slika 0.6.** Odabiremo C

Ime našeg projekta će biti *Hello*. To upisujemo u *Project title* polje kako je prikazano slikom 0.7. Odabiremo *Next* i na sljedećem prozoru stisnemo *Finish* (slika 0.8).

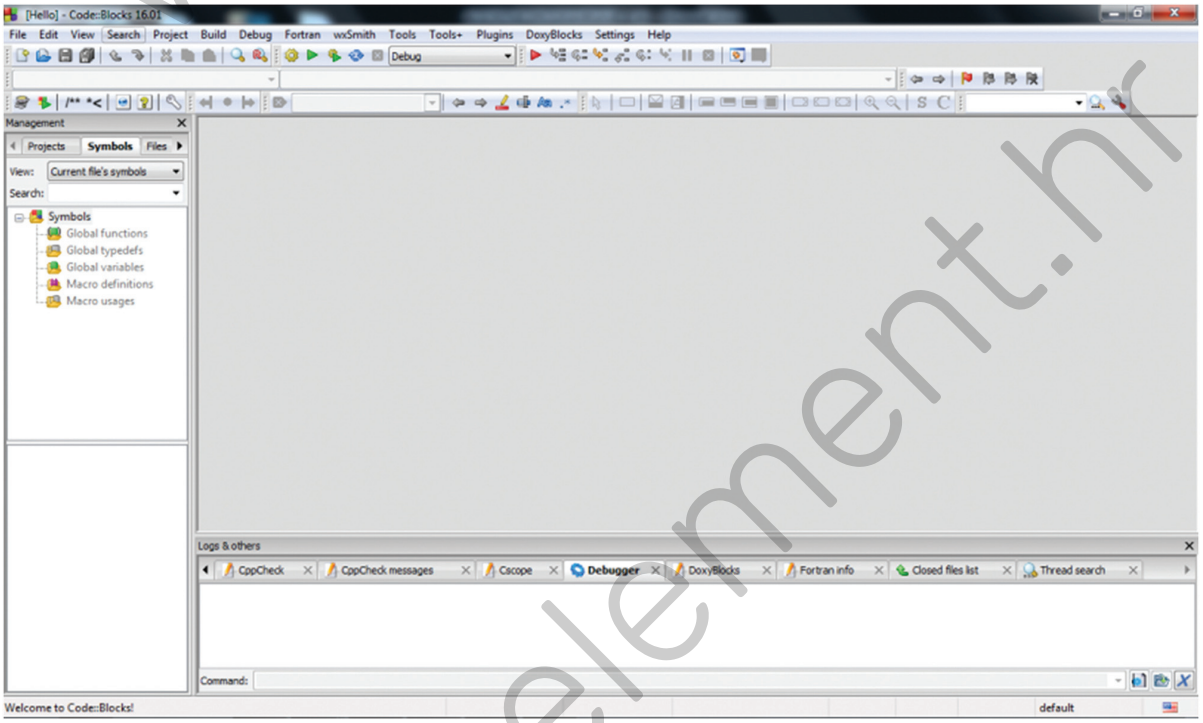



**Slika 0.7.** Dodavanje imena projektu




**Slika 0.8.** Završavanje kreiranja projekta

Nakon svih postavki koje smo do sada postavili, prozor izgleda kako je prikazano slikom 0.9.

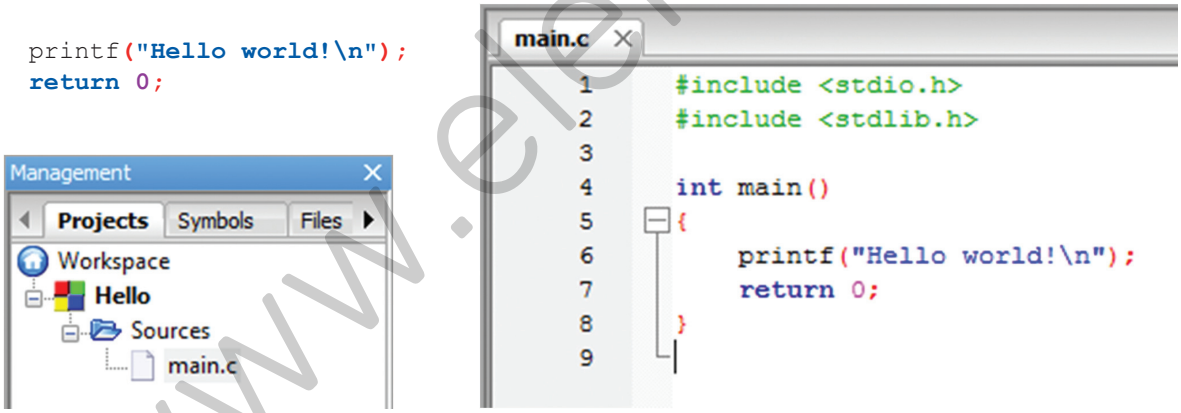



**Slika 0.9.** Program spreman za pokretanje editora teksta

Odabiremo *Projects* u lijevom kutu gore i pod *Sources* nalazimo *main.c*. Odabirom *main.c* otvorit će nam se prikazani kod:

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

int main()
{
    printf("Hello world!\n");
    return 0;
}
```



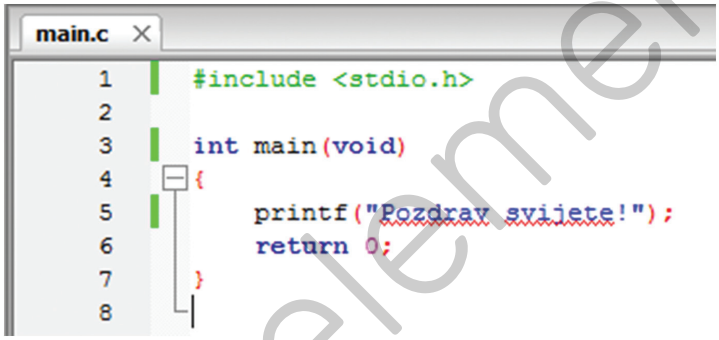

**Slika 0.10.** Editor teksta unutar razvojnog okružja Code::Blocks

Kao prvi korak u svijet programiranja dobiveni ćemo kod preurediti tako da se poruka ispiše na hrvatskom jeziku. Stoga ćemo obrisati tekst "Hello world!\n" i napisati "Pozdrav svijete!". Izgled teksta prikazan je slikom 0.12. S pomoću naredbe *Save file*, koja je u izborniku *File*, spremamo prvi program te ga kompiliramo (prevodimo u strojni jezik) s pomoću ikone ***Build*** (slika 3.11.)

**Slika 0.11.** Ikona *Build* koja služi za kompiliranje programa

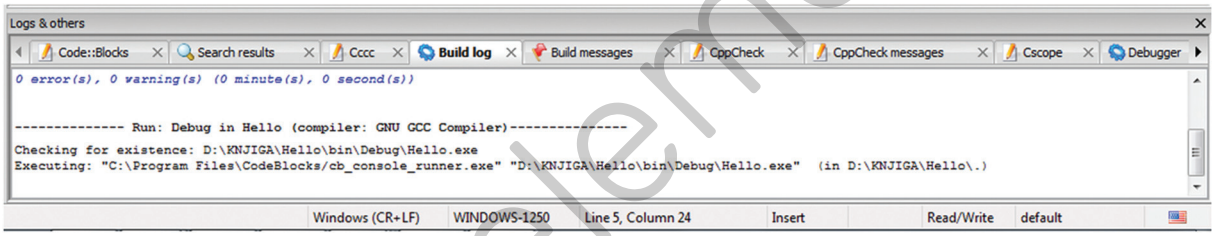
**Napomena**

Naredba `#include <stdlib.h>` nam za sada nije potrebna pa ćemo ju obrisati.




**Slika 0.12.** Upisan prvi program – "Pozdrav svijete!"

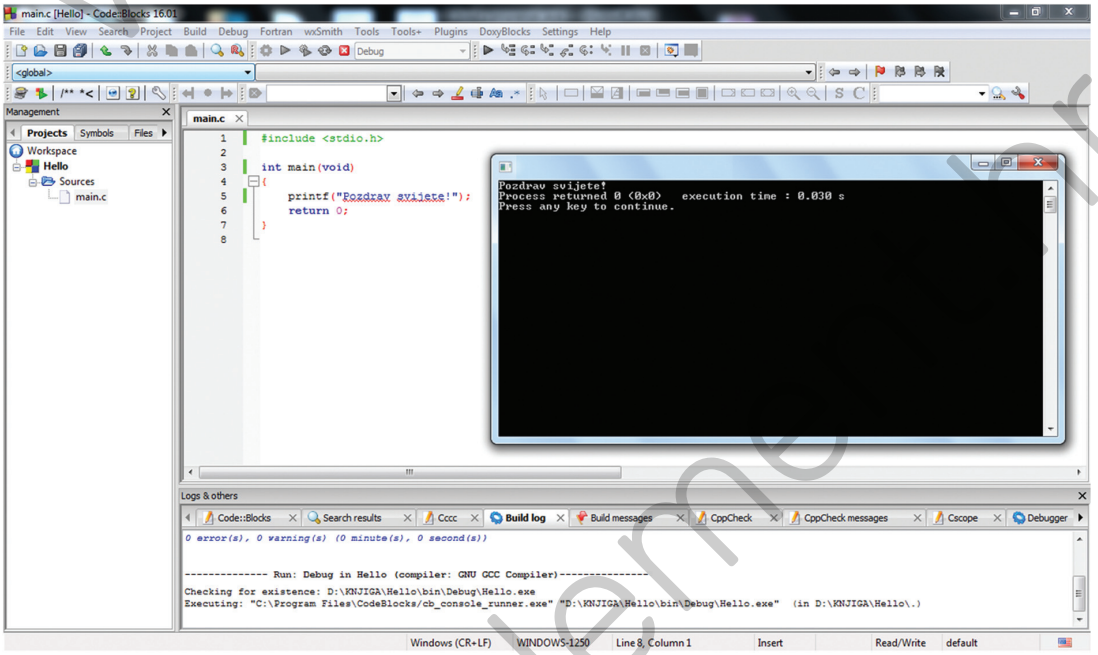
Nakon uspješnog prevođenja u strojni kod dobivamo poruku o tome u okviru prikazanom slikom 0.13.




**Slika 0.13.** Rezultat o uspješnom prevođenju programa u strojni kod

Pokretanjem zelene strelice (***Run***) (slika 0.14) dobivamo izlaz iz našeg programa na zaslonu računala. U ovom slučaju je to "Pozdrav svijete!".

**Slika 0.14.** Ikona *Run* – služi za pokretanje programa




**Slika 0.15.** Uspješno kompiliranje prvog programa i rezultat izvođenja programa

* * *

U programskom jeziku C susreli smo se sa tri osnovne algoritamske strukture:

**a)** **linijska** algoritamska struktura – slijed
**b)** **razgranata** algoritamska struktura – grananje
**c)** **ciklička** algoritamska struktura – petlja.

## 0.2.2. Linijska algoritamska struktura

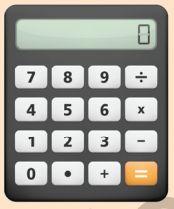
**Linijska algoritamska struktura** nastaje tako da navodimo naredbu za naredbom. Svaki dobar algoritam ima konačan broj točno povezanih blokova. Redoslijed blokova u kojima se obavljaju različite naredbe mora biti smislen kako bi i algoritam u konačnici davao smisleno rješenje problema.

**Primjer 0.1.**

Kalkulator

Napiši program koji će simulirati rad jednostavnog kalkulatora. Korisnik kalkulatora upisuje dva cijela broja *a* i *b*, a program mu izračuna i ispiše za ta dva broja rezultat njihovog:

- zbrajanja (`a + b`)
- razlike (`a – b`)
- umnoška (`a * b`)
- količnika (`a : b`)
- cjelobrojnog dijeljenja (`a / b`) i
- dijeljenja s ostatkom (`a % b`).




ULAZNI PODATCI

- cijeli broj *a* (–200 < *a* < 200), prvi operand
- cijeli broj *b* (–200 < *b* < 200), drugi operand koji ne smije biti 0

IZLAZNI PODATCI

- u prvom redu ispisati zbroj
- u drugom redu ispisati razliku
- u trećem redu ispisati umnožak
- u četvrtom redu ispisati količnik
- u petom redu ispisati rezultat cjelobrojnog dijeljenja (`DIV`)
- u šestom redu ispisati ostatak cjelobrojnog dijeljenja (`MOD`)

## PRIMJER TEST-PODATAKA

| Ulaz | Izlaz |
|---|---|
| 6<br>4 | 6 + 4 = 10<br>6 – 4 = 2<br>6 * 4 = 24<br>6 : 4 = 1.50<br>6 / 4 = 1<br>6 % 4 = 2 |

**Ponovimo**

Cjelobrojno dijeljenje:
`6 / 4 = 1`

Decimalno dijeljenje:
`6. / 4 = 1.5`
`(float) 6 / 4 = 1.5`
`6 / (float) 4 = 1.5`

Prvo ćemo deklarirati brojeve *a* i *b*, onda učitati cijele brojeve *a* i *b*, a potom unutar `printf` naredbi možemo provesti sve zadane računske operacije.

Kôd programa

```
#include <stdio.h>

int main(void){
    int a, b;

    printf("Unesite prvi broj:\n");
    scanf("%d",&a);

    printf("Unesite drugi broj:\n");
    scanf("%d",&b);

    printf("%d + %d = %d\n", a, b, a + b);
    printf("%d - %d = %d\n", a, b, a - b);
    printf("%d * %d = %d\n", a, b, a * b);
    printf("%d : %d = %.2f\n", a, b, (float)a / b);
    printf("%d / %d = %d\n", a, b, a / b);
    printf("%d %% %d = %d\n", a, b, a % b);

    return 0;
}
```

**Ponovimo**

Ako želimo podijeliti dva cijela broja, moramo napraviti pretvorbu jednog cijelog broja u realni broj. To je prikazano u `printf` naredbi označenoj u kodu:

```
printf("%d : %d = %.2f\n", a, b,(float)a / b);
```

To smo mogli napisati i ovako:

```
printf("%d : %d = %.2f\n",a, b, a /(float)b);
```

Ako ne napravimo pretvorbu podataka, rezultat će biti cjelobrojno dijeljenje, što je napisano u idućem retku koda.

## 0.2.3. Razgranata algoritamska struktura

**Razgranata algoritamska struktura omogućuje uvjetno grananje algoritma.** Uvjet određuje u kojem smjeru će ići odvijanje algoritma. Algoritam će ići u onom smjeru kako mi to definiramo.

**Grananje** u programskom jeziku C može biti:

**a) jednostruko** grananje – `if` naredba

```
if(uvjet){
    ...
    naredbe;
    ...
}
```

**b) dvostruko** grananje – `if-else` naredba

```
if(uvjet){
    ...
    naredbe;
    ...
}
else{
    ...
    naredbe;
    ...
}
```

**c) višestruko** grananje – `if-else` i `switch-case` naredba

```
if(uvjet)
    naredbe;
else  if(uvjet)
          naredbe;
      else  if(uvjet)
                naredbe;
            else
                naredbe;
```

```
switch(uvjet){
    case izraz_1:
        ...
        naredbe;
        ...
        break;
    case izraz_2:
        ...
        naredbe;
        ...
        break;
    ...
    case izraz_N:
        ...
        naredbe;
        ...
        break;
    default:
        ...
        naredbe;
        ...
}
```

Sljedeći primjer pokazuje kako koristimo `if-else` naredbu.

**Primjer 0.2.**

Tenisice

Marko želi kupiti nove tenisice za nogomet pa radi usporedbu cijena u dvije trgovine. Tenisice je isprobao u trgovini *SUPER* i trgovini *Sport*. Marko upisuje u program kolika je cijena proizvoda u jednoj trgovini, a kolika u drugoj trgovini. Napiši program koji ispisuje u kojoj trgovini je proizvod jeftiniji i za koliko kuna. Ako je cijena jednaka u obje trgovine, neka ispiše poruku: *Jednaka cijena*.

ULAZNI PODATCI

- realni broj *c1*, cijena tenisica u trgovini *SUPER*
- realno broj *c2*, cijena tenisica u trgovini *Sport*

IZLAZNI PODATCI

- u prvom redu ispisati u kojoj trgovini su jeftinije tenisice
- u drugom redu ispisati za koliko su jeftinije tenisice

PRIMJER TEST-PODATAKA

| Ulaz | Izlaz |
|---|---|
| 649.99<br>679.99 | Tenisice su jeftinije u trgovini SUPER.<br>Jeftinije su za 30.00 kuna. |

Kôd programa

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

int main(void){
    float c1, c2, R;
    scanf("%f %f",&c1,&c2);
    R = fabs(c2 - c1);

    if(c1 < c2)
        printf("Tenisice su jeftinije u trgovini SUPER.\n
                Jeftinije su za %.2f kuna.\n", R);
    else  if(c1 > c2)
              printf("Tenisice su jeftinije u trgovini Sport.\n
                      Jeftinije su za %.2f kuna.\n", R);
          else
              printf("Jednaka cijena.\n");

    return 0;
}
```

Sljedeći primjer prikazuje primjenu `switch-case` naredbe.

**Primjer 0.3.**

Prometni znakovi

Napiši program, koji će unositi jedno od slova A, B, C ili D koja predstavljaju navedene znakove opasnosti. Potrebno je ispisati odgovarajuću poruku koja opisuje navedeni znak. Ako korisnik unese pogrešno slovo treba ispisati poruku: *pogresan unos*.

A – divljač na cesti

B – životinje na cesti

C – tramvajska pruga

D – blizina obale

ULAZNI PODATCI

- slovo koje predstavlja jedan od navedenih znakova

IZLAZNI PODATCI

- ispisati tekst koji opisuje željeni znak opasnosti

PRIMJER TEST-PODATAKA

| Ulaz | Izlaz |
|---|---|
| A | divljac na cesti |
| a | pogresan unos. |

Kôd programa

```
#include <stdio.h>
int main(void){
    char c;

    printf("Unesite slovo:\n");
    scanf("%c",&c);

    switch(c){
        case 'A':  printf("divljac na cesti"); break;
        case 'B':  printf("zivotinje na cesti"); break;
        case 'C':  printf("tramvajska pruga"); break;
        case 'D':  printf("blizina obale"); break;
        default:   printf("pogresan unos.");
    }
    return 0;
}
```

## 0.2.4. Ciklička algoritamska struktura

**Cikličkom strukturom je moguće određeni blok naredbi ponoviti više puta.** To olakšava pisanje algoritma tako što ne moramo više puta pisati iste naredbe.

U C-u razlikujemo tri vrste programskih petlji: **a)** `for` petlja
**b)** `while` petlja
**c)** `do-while` petlja.

**a)** **`for`** petlja – poznat broj ponavljanja

Opći oblik `for` petlje:

```
for(inicijalizacija; uvjet; korak petlje){
    naredbe;
}
```

**b)** **`while`** petlja – petlja se izvršava u ovisnosti o postavljenom uvjetu

Opći oblik `while` petlje:

```
while(uvjet){
    naredbe;
}
```

**c)** **`do-while`** petlja – petlja se barem jednom izvršava jer je uvjet ispituje tek na kraju petlje

Opći oblik `do-while` petlje:

```
do{
    naredbe;
}while(uvjet);
```

U sljedećem primjeru je pokazana primjena `for` petlje.

**Primjer 0.4.**

Zaključna ocjena

Marka jako zanima koliki će imati prosjek (ocjenu) na kraju školske godine iz predmeta *Programiranje*. Ako je Marko tijekom školske godine dobio *N* ocjena, odredi koliku će imati zaključnu ocjenu. U ovom slučaju je pravilo da se ocjena zaključuje samo prema aritmetičkoj sredini.

ULAZNI PODATCI

- u prvom redu je prirodni broj *N* koji predstavlja broj ocjena koje je Marko dobio tijekom školske godine
- u preostalih *n* redova nalazi se *N* ocjena, *oc* (1 – 5)

IZLAZNI PODATCI

- u prvom redu ispisati prosjek na dvije decimale
- u drugom redu ispisati koju ocjenu ima učenik