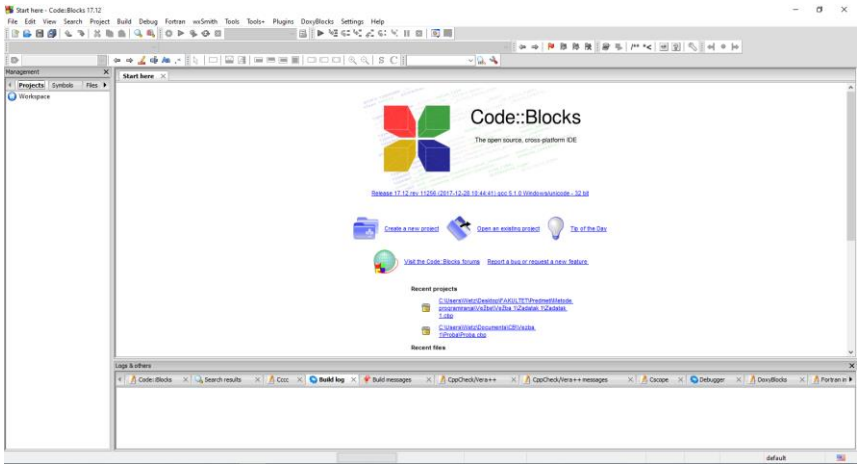
SISTEMSKO PROGRAMIRANJE

RAD U CODE::BLOCKS-U, OSNOVE PROGRAMSKOG JEZIKA C, STRUKTURA PROGRAMA, PROMENLJIVE I KONSTANTE

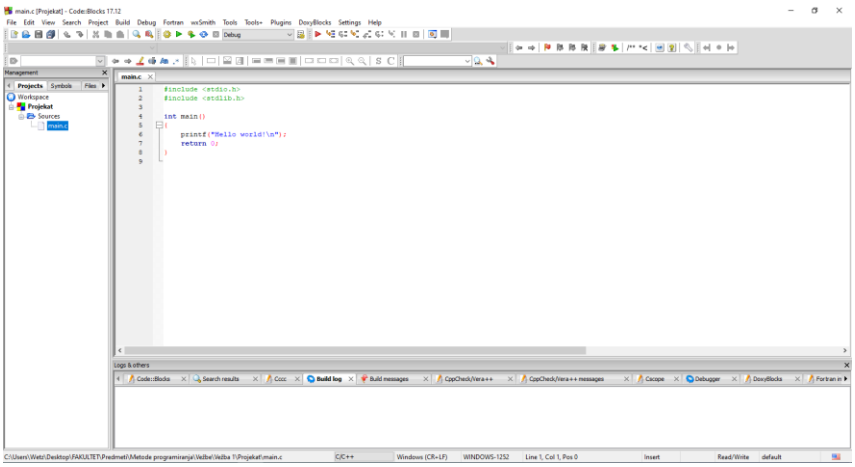
1. RAD U CODE::BLOCKS-U

Na predmetu sistemsko programiranje će se koristiti softerski alat, Code::Blocks IDE, open source koji je besplatan i može se download-ovati sa sledeće adrese: http://www.codeblocks.org/downloads/25. Da bi se program mogao kompajlirati i pokrenuti program , potrebno je skinuti instalaciju sa http://www.codeblocks.org/downloads/26 codeblocks-17.12-mingw-setup koja sadrži GCC kompajler. Nakon uspešne instalacije i pokretanja programa Code::Blocks će izgledati kao na slici 1.



Slika 1. Početni ekrana programa Code::Blocks

Sledeće što je potrebno jeste, kreirati novi projekat (File→New→Project) i nakon toga izabrati *Console application* i klikom na dugme [*Next*] otvoriće se novi prozor koji će sadržati izbor programskog jezika. Izbor je između programskog jezika C i C++. Potrebno je izabrati jezik C i klikom na dugme [*Next*] se otvara novi prozor koraku posle toga dati naziv projektu kao i lokaciju gde će se kod sačuvati. Posle ovih koraka, klikom na dugme [*Finish*] pojaviće se novi prozor kreiranog projekta (slika 2).



Slika 2. Ekran nakon kreiranog projekta

U postojećem *Workplace*-u pojavio se projekat koji je kreiran uz folder *Sources* sa jednom datotekom *main.c* u kojoj se nalazi main funkcija u kojoj se nalazi običan printf iskaz sa tekstom „Hello world!“.

2. OSNOVE PROGRAMSKOG JEZIKA C

Nakon što se kreira projekat, u njemu se automatski kreira i *main.c*. *Main.c* sadrži sledeći kod:

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

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

Listing 1. Početni kod main.c datoteke

3. STRUKTURA PROGRAMA

Programski jezik C je case-sensitive, što u prevodu znači da postoji razlika između malih i velikih slova.
Posle svake naredbe u programskom jeziku C se mora nalaziti karakter „**;**“.
Isto tako, C ne prepoznaje razmake koji se nalaze u samom kodu, tako da se prelazak u nov red pomoću „\n” nakon svake komande ili iskaza koristi samo iz razloga što je kod dosta pregledniji.
Komentari se mogu definisati na dva načina:

- Prvi način je da se koriste karakteri //. Ti karakteri se koriste kada korisnik želi da se komentar nalazi u jednom redu.
- Drugi način je da se koriste karakter /* i karakter */. Ovaj par karaktera se koristi kada je potrebno postaviti komentar u više redova.

```
// Komentar u jednom redu
   /* Komentar
      u više
      redova.    */
```

Listing 2. Definisanje komentara

U svakom postojećem kodu se nalaze funkcije koje su potrebne za izvršavanje programa.U prvom redu koda, u datoteci stdio.h se nalaze funkcije koje služe za ulaz i izlaz podataka (*printf*(), *scantf*(), *getc*(), *getchar*(), *f open*(), *f close*(),...), a datoteci stdlib.h se nalazi funkcije koje se najčešće koriste (*exit*, *system*, *div*, *abs*,...).
Sve funkcije koje se nalaze u programu se pozivaju iz glavne funkcije, odnosno iz *main*() funkcije, koja je specijalna funkcija iz razloga što izvršavanje programa počinje od nje. Ona ne mora biti tipa int, može biti i tipa npr. void i sadržati istu listu argumenata.
U samom programu se mora nalaziti i par vitičastih zagrada {}. Vitičaste zagrade predstavljaju granice delova naredbi.

```
return 0;
```

Listing 3. Vrednost nula

Izlaz glavnog programa vraća vrednost 0.

```
printf("Hello world!\n");
```

Listing 4. Funkcija printf

Funkcija *printf* je definisana u datoteci stdio.h i njen zadatak je da štampa tekst, odnosno u ovom slučaju štampa na ekranu „Hello world!“. Karakteri su ograničeni sa znacima navoda (“ “). Funkcija karaktera \n označava novi red i on se nalazi na kraju niza u *printf*-u.

4. PROMENLJIVE I KONSTANTE

Promenljive su objekti koji sadrže neke vrednosti, u koje se mogu upisivati vrednosti i iz kojih se mogu čitati vrednosti.

```
tip_promenljive ime_promenljive;
```

Listing 5. Definicija promenljive

U tip promenljive se postavljaju neki od već ugrađenih tipova u programskom jeziku C poput:

1. *int* – celobrojni tip. Iz celobrojnih tipova se mogu izvesti sledeći tipovi:
   - *short int*,
   - *long int*,
   - *unsigned int* – pozitivan celi broj
2. *float* – realni tip jednostruke tačnosti,
3. *double* – realni tip dvostruke tačnosti,
4. *char* – tip karaktera

Imena promenljivih mogu sadržati slova, donju crtu, cifre. One moraju početi ili sa donjom crtom ili sa malim slovom. Kao što je već rečeno, mala i velika slova se razlikuju u programskom jeziku C, odnosno C je case-sensitive i odnosi se na nazive promenljivih. Na osnovu imena u programu se pristupa promenljivoj i možemo saznati njenu trenutnu vrednost, promeniti joj vrednost ili je koristiti u izrazima. Ime promenljive ne može biti neka od rezervisanih reči u programskom jeziku C.

Tabela 1. Rezervisane reči u programskom jeziku C

auto	else	long	switch	_Atomic
break	enum	register	typedef	_Bool
case	extern	restrict	union	_Complex
char	float	return	unsigned	_Generic
const	for	short	void	_Imaginary
continue	goto	signed	volatile	_Noreturn
default	if	sizeof	while	_Static_assert
do	inline	static	_Alignas	_Thread_local
double	int	struct	_Alignof	

Primeri deklaracija promenljivih:

```
int a, b, c;
// int a, int b, int c; - drugi način deklarisanja promenljivih
float obim, povrsina;
double d, e;
char c;
```

Listing 6. Primer deklaracija promenljvih

Da bi se dodelila neka vrednost promenljivoj, potrebno je koristiti operator =. Ona ima sledeći oblik

```
ime_promenljive = neka vrednost;
```

Listing 7. Primer dodele vrednosti

Ukoliko je promenljiva tipa *char* vrednost mora biti jedan karakter, koji je ograničen jednostrukim znacima navoda (’). Specijalni karakteri poput (\n, \t, \b) se tretiraju kao jedan karakter, iako se, praktično sastoje iz dva karaktera. Tip *char* je ustvari celobrojna vrednost ASCII koda karaktera.

```
char x, y, z, w;
  x = 'x';
  y = '\n';
  z = '1';
  w = '.';
```

Listing 8. Definisanje tipa char

Postoji i mogućnost dodeljivanja vrednosti u istoj naredbi:

```
  char x = 5;
  float y = 1.5, z = 155e-5
  char z, w = 'x';
```

Listing 9. Drugi način dodeljivanja vrednosti

Na početku vežbe je spomenuta funkcija *printf* čiji je posao da odštampa neki niz karaktera na standardnom izlazu. Oblik funkcije je:

printf(“niz_znakova“, ime_promenljive1, ime promenljive2,...);

Niz znakova predstavlja niz karaktera pri čemu se sa znakom % pokazuje mesto gde je potrebno upisati vrednosti promenljivih koje se nalaze iza znakova za ispis. Nakon znaka % ide odgovarajuće slovo koje je zavisno od tipa promenljive:

- %d – celobrojan tip u dekadnom sistemu,
- %o – celobrojan tip u oktalnom sistemu,
- %x – celobrojan tip u heksadecimalnom sistemu,
- %f – realni tip (koriste se tipovi *float* ili *double*),
- %c – tip karaktera (*char*)

Primeri za štampanje:

```
printf (“%d”, x);  //štampa promenljivu x u dekadnom sistemu
printf (“%f %f‘, a, b); //štampa realne brojeve promenljivih a i b
printf (“%d %o %x”, x , y, z); /* štampaju se promenljive x, y, z u dekadnom, oktalnom i heksadecimalnom brojnom sistemu */
```

Listing 10. Primeri štampanja

Nakon definisanja promenljive, potrebnu memoriju za njeno čuvanje treba alocirati. Veličina te memorije zavisi od samog tipa promenljive, ali i od sistema na kojem se taj program izvršava. U programskom jeziku C je ugrađen operator *sizeof* koji određuje veličinu promenljive ili tipa promenljive u bajtovima.

```
int x = sizeof(int);
prinf (“%d”, sizeof(int));
```

Listing 11. Primer štampanja veličine u bajtovima za tip int

Zadatak 1. Napisati program koji zadaje celobrojne vrednosti, promenljivim a i b i štampa njihov zbir.

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

int main()
{
	int x = 10, y = 20;
	int r = x + y;
	printf("Zbir je %d\n", r);
return 0;
}
```

Listing 12. Rešenje zadatka 1.

Zadatak 2. Napisati program koji štampa za sve osnovne tipove podataka veličinu memorijskog prostora.

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

int main()
{
	printf("int: %d\n", sizeof(int));
	printf("short: %d\n", sizeof(short));
	printf("long: %d\n", sizeof(long));
	printf("unsigned: %d\n", sizeof(unsigned));
	printf("float: %d\n", sizeof(float));
	printf("double: %d\n", sizeof(double));
	printf("char: %d\n", sizeof(char));
	return 0;
}
```

Listing 13. Rešenje zadatka 2.

Slično promenljivima, konstante su objekti koji imaju svoje ime ali se vrenost ne može menjati tokom izvršenja programa. Za definisanje se koristi ključna reč *const*.

```
const int x = 10;
```

Listing 14. Definisanje konstante

Drugi način definisanja konstante je korišćenje direktive #define: Ovu direktivu je potrebno napisati na početku programa, ispod direktiva #include.

```
#define ime_konstante vrednost_konstante
```

Listing 15. Definisanje konstante na drugi način

Za učitavanje znakova preko konzole koristi se funkcija *scanf* koja se poziva u obliku:

```
scanf(“ format_unosa, pokazivac_na_promenljivu1, pokazivac_na_promenljivu2,…”);
```

Listing 16. Definisanje funkcije scanf

Prvi argument ove funkcije *scanf* je niz izraza koji počinju sa znakom % i koji označava vrstu podataka koji se čitaju preko konzole, a koriste iste oznake kao za ispis sa *printf*-a. Ostali argumenti predstavljaju pokazivače na promenljivim u koje će se smeštati učitane vrednosti. Pokazivače za sada treba shvatiti kao adrese promenljivih, a označavaju se sa početnim znakom & iza kojeg sledi ime promenljive. Promenljive koje se ovde nalaze moraju biti deklarisane u programu.

```
scanf (“%d”, &x);
scanf (“%f”, &y);
```

Listing 17. Primer definisanja funkcije scanf