UNIVERZITET METROPOLITAN BEOGRAD

CS130 - C/C++ PROGRAMSKI JEZIK

Uvod u C jezik

Lekcija 01

PRIRUČNIK ZA STUDENTE

CS130 - C/C++ PROGRAMSKI JEZIK

Lekcija 01

UVOD U C JEZIK

UVOD

Jezik C je nastao sasvim spontano u Bell-ovim laboratorijama 1972. godine. C++ je nastao 1983. godine kao proširenje jezika C u cilju podrške objektno orijentisanom programiranju

Programski jezik C je nastao spontano. Naziv ***C*** se pojavio po tome što je C bio naslednik prethodne verzije za interno korišćenje koji se zvao ***B*** . Jezik ***C*** razvili su 1972. godine Denis Riči (Dennis Ritchie) i Brajan Kernigan (Brian Kernighan) evolucijom jezika ***BCPL*** i ***B*** .

Početkom osamdesetih godina takođe u Belovim laboratorijama Bjorn Stroustrup radeći na proširivanju jezika C razvio je suštinski nov jezik koga je nazvao ***“C sa klasama”*** . Autor je želeo da poboljša jezik C razvojem podrške objektno orijentisanom programiranju. 1983. godine taj naziv je promenjen u ***C++*** .

Jezik C++ se razlikuje od običnog C-a pre svega u podršci objektno-orijentisanom programiraju. Ali, u njemu ima i niz novih mogućnosti, koje nisu objektnog karaktera, zbog kojih je pisanje programa koji nisu objektne prirode pogodnije realizovati u C++-u nego u C-u.

Poglavlje 1

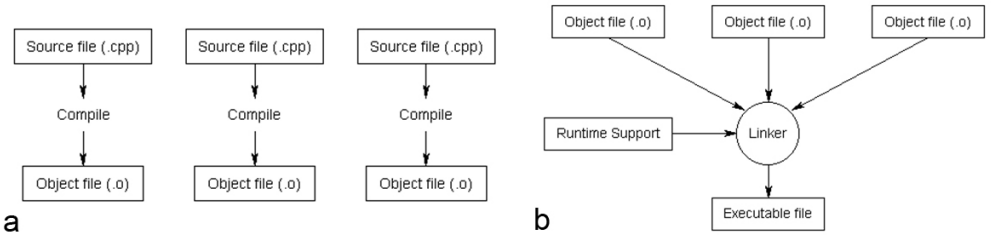
Faze generisanja programa

RAZVOJ C I C++ PROGRAMA

Razvoj programa se sastoji iz sledećih koraka: pisanje programa, prevođenje, povezivanje programa (linkovanje), punjenje memorije i izvršavanje programa

Razvoj programa se sastoji iz sledećih koraka:

- ***Pisanje programa*** - Ekstenzija fajlova koji sadrže izvorni kod (source code) za programski jezik C je **.c*, a u programskom jeziku C++ je **.cpp*.
- ***Prevođenje*** - prevođenje programa na mašinski jezik koji je razumljiv procesoru. Tako se dobija datoteka (fajl) sa ekstenzijom **.obj*.
- ***Povezivanje programa*** - (linkovanje, eng. linking) uključivanje standardnih funkcija iz C/C++ biblioteka, kao i novo napravljenih funkcija od strane samog programera.



Slika 1.1 a) Postupak prevođenja programa na jezik razumljiv računaru; b) povezivanje programa u izvršni fajl [7]

- ***Izvršenje programa*** - startovanje aplikacije koja je napravljenja prevođenjem i povezivanjem. Ekstenzija tako dobijenog programa u **Windows** okruženju je **.exe*.

Da bi se neki C/C++ kod programa preveo u izvršni fajl koji računar može da izvršava, obično se koriste tri programa: Pretprocesor, Kompajler i Linker -

Pretprocesor se izvršava pre nego što kompajler počne sa prevođenjem (kompajliranjem) programa.

Kompajler čita izvorni kod i proizvodi nezavisan izvršni fajl.

Konačno, linker se koristi da poveže izvršnu (run-time) biblioteku sa objektnim fajlom, da bi se dobio program koji se može izvršiti na računaru.

UPOTREBA INTEGRISANIH RAZVOJNIH OKRUŽENJA

Integrisano razvojno okruženje objedinjava sve korake odnosno faze razvoja softvera u jedinstvenu celinu što olakšava pisanje i testiranje programa

Softverski paket poznat kao integrisano razvojno okruženje (integrated development environment – IDE) integriše sve prethodno opisane funkcije (razvoj koda, kompajliranje, linkovanje, kao i debug-iranje) u jednu jedinstvenu celinu što olakšava pisanje i testiranje programa.

Korisnik uz IDE dobija koristan editor koda koji u sebi sadrži opcije za numeraciju linija koda i isticanje (highlighting) grešaka u sintaksi.

IDE sadrži i debug-er, tj. integrisani program koji omogućava praćenje izvršavanja vašeg koda liniju po liniju što olakšava pronalaženje grešaka.

Osim toga, IDE obično uključuje i veliki broj drugih pomoćnih alata, kao što je integrisani program za pomoć (help), dovršavanje reči (autocomplete), itd.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Neki od najpoznatijih integrisanih razvojnih okruženja, čije se korišćenje istovremeno preporučuje u okviru ovog predmeta, su:

- Microsoft’s Visual C++ (Windows OS)
- Code::Blocks (Windows/Linux OS)
- Netbeans C/C++ (Wndows/Linux OS)

Poglavlje 2

Osnovna anatomija C programa

PRVI PROGRAM U C-U

C program se sastoji iz funkcija i promenljivih, i svaki program mora imati main() funkciju

Program u C-u se sastoji od funkcija i promenljivih. Glavna funkcija je main od koje počinje izvršavanje programa. Svaki program mora imati *main* funkciju koja zatim poziva sve druge funkcije. U programskom jeziku C kod se može nalaziti u jednom ili više fajlova. Međutim, samo jedan od tih fajlova može da sadrži funkciju *main*.

Napišimo jedan kratki program koji će odštampati na ekranu poruku ***"Nole Sampion!*** "

```
#include <stdio.h>
void main()
{
 printf("Nole Sampion!\n");
}
```

Da bi mogao da koristi funkcije iz sistemske biblioteke u programu se mora navesti direktiva pretprocesoru kojoj prethodi znak *#*. U navedenom primeru pozivamo direktivu *#include* ***<stdio.h>*** kojom se uključuju sistemske funkcije (standardne ulazne/izlazne funkcije itd).

Opis funkcije u C-u se sastoji iz zaglavlja i tela funkcije. Zaglavlje funkcije u ovom prostom obliku čine ime funkcije i zagrade *()*. Iza zaglavlja se navodi telo funkcije čiji se početak označava znakom *{*, a kraj znakom *}*. Velike zagrade (vitičaste zagrade *{}*) se istovremeno koriste za početak i kraj nekog programskog bloka u jeziku C.

Da bismo ispisali nešto na ekranu koristimo funkciju *printf*. Ono što se ispisuje na ekranu se navodi unutar navodnika ***" … "*** . Deklaracija funkcije printf se upravo nalazi u ***stdio.h*** fajlu. Na kraju samog teksta se nalaze dva znaka: *\n*. To je ustvari znak koji omogućava prelazak u novi red. Na kraju linije se stavlja tačka zarez (*;*) da bi kompajler znao gde je kraj naredbe.

Ukoliko bi zapisali naš program u fajl ***Nole.c*** a zatim ga preveli i povezali, on bi nakon startovanja ispisao poruku na ekranu (konzolu): ***"Nole Sampion!"*** .

OSNOVNA PRAVILA PISANJA C PROGRAMA

Jedan C program se u osnovi sastoji iz: pretprocesorkih naredbi, funkcija, promenljivih, iskaza, izraza i komentara

U jeziku C postoje 32 rezervisane reči i to:

auto default enum if short typedef

break do extern int sizeof union

case double float long static unsigned

char else for register struct void

continue goto return switch volatile const

Ove rezervisane reči ne mogu biti korišćenje u druge svrhe osim u one za koje su namenjene. Uvek se pišu malim slovima. Može se javiti još rezervisanih (službenih) reči u određenim kompajlerima (prevodiocima) za jezik C/C++.

Karakteri koji se koriste prilikom pisanja programa u jeziku C su:

- A – Z (velika i mala slova engleske abecede)
- 0 – 9 (cifre)
- #%&!_{}[]<>| (simboli)
- Tab dugme . , : ; ’ **$** ” (specijalni simboli)
- + - / * = (osnovne računske operacije).

Komentari predstavljaju pomoćni tekst u C/C++ programu koji se ignoriše od strane kompajlera. Komentari se u C-u pišu između /* i */ znakova:

```
/* moj prvi program u C-u */
```

Drugi oblik komentara je jednolinijski komentar. On počinje simbolom // i proteže se do kraja linije. Na primer:

```
printf("Zdravo svete!"); // Sve odavde na desno ce biti ignorisano
```

FAJLOVI SA ZAGLAVLJIMA (HEADER FILES)

Fajlovi sa zaglavljima sadrže deklaracije i makro definicije koje će biti korišćenje u drugim fajlovima sa izvornim kodom

Header fajl ili fajl sa zaglavljima je datoteka sa ekstenzijom ****.h*** koja obično sadrži deklaracije funkcija i makro definicije koje će biti korišćenje u drugim fajlovima sa izvornim kodom. Postoje dve vrste fajlova sa zaglavljima i to: korisnički fajlovi i header fajlovi koji idu uz kompajler.

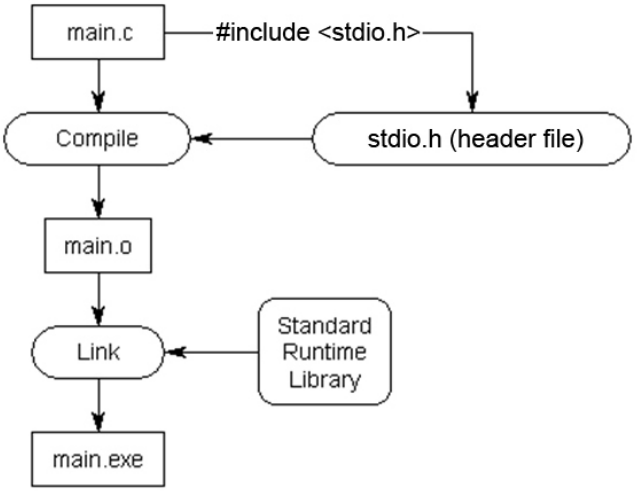
Da bi se neki header fajl koristio u programu, neophodno je pre toga uključiti ga pomoću pretprocesorske direktive ***#include***. Fajlovi sa zaglavljima koji su deo standardne C (C++) biblioteke se uključuju na sledeći način:

```
#include <file>
```

Kompajler pre samog kompajliranja koda u direktorijumu standardne biblioteke traži da li postoji datoteka pod imenom *file*.

U header fajlovima se obično nalaze deklaracije, dok se implementacija nalazi u okviru ***izvršne biblioteke*** (runtime library), koja je automatski uključena u program prilikom faze povezivanja (linkovanja).

Standardna izvršna biblioteka (Standard Runtime Library) je ustvari jedan paket koda koji će po potrebi biti više puta korišćen u programima. Na Slici 2.1 je primer uključivanja standardne izvršne biblioteke u fazi linkovanja programa.



Slika 2.1 Postupak uljučivanja fajla zaglavlja u vaš program, za kojim sledi prevođenje i povezivanje programa (uz uključivanje standardne biblioteke) u izvršnu verziju [7]

Obično se biblioteka sastoji iz: fajlova zaglavlja u kojima se nalaze deklaracije za sve što je proglašeno javnim i dostupnim korisnicima, odnosno piscima programskog koda, i gotovih i unapred iskompajliranih (precompiled) objekata čiji je kod već preveden na mašinski jezik. Ove biblioteke obično imaju ekstenziju *.lib* ili *.dll* na Windows platformi, odnosno *.a* ili *.so* ekstenziju na Unix-u.

Poglavlje 3

Standardni ulaz i izlaz u C-u

FUNKCIJE SCANF I PRINTF

Funkcija scanf() učitava podatke sa standardnog ulaza (tastatura) dok funkcija printf() štampa podatke na ekranu. Neophodno je uključiti stdio.h header fajl u cilju njihovog korišćenja

Učitavanje podataka u program se vrši primenom funkcije *scanf* dok se ispis podataka vrši primenom funkcije *printf*.

Funkcije *printf* i *scanf* nisu deo C jezika već su deo standardne sistemske biblioteke ***stdio.h*** . Stoga, datoteka ***stdio.h*** se mora uključiti u program odgovarajućom *include* direktivom.

Ukoliko želimo da sa tastature unesemo neki ceo broj, neophodno je da deklarišemo promenljivu koja će biti korišćena u programu:

```
int a;
```

i da zatim pozovemo funkciju *scanf* u sledećem obliku:

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

Oznaka *%d* predstavlja format za unos celobrojnih promenljivih a *&a* predstavlja adresu promenljive gde želimo da upišemo unetu vrednost.

Za sada nećemo ulaziti u detaljnije objašnjenje zašto se baš koristi adresa promenljive *a*, jer se to obrađuje u odeljku za pokazivače. Ukoliko želimo da ispišemo neki rezultat na ekranu koristimo funkciju *printf*. Opšti oblik funkcije *printf* je:

```
printf(format, promenljiva1, … promenljivaN);
```

Prvi argument funkcije *printf* je format koji se sastoji iz niza znakova u kome se navode konverzije koje treba da se izvršavaju u toku ispisivanja podataka. To znači da kompajler ispisuje na ekranu sve što smo zadali u formatu do pojave procenta (*%*), posle čega se čita odgovarajući znak i onda se ispisuju promenljive onim redom kako je u formatu zadato.

Ukoliko želimo da ispišemo rezultat promenljive *a* na ekran koristimo sledeći oblik funkcije *printf*:

```
printf("Vrednost promenljive a je:  %d", a);
```

Kao što vidimo, opet koristimo *%d* za format ispisa celobrojne promenljive.

SIMBOLI KONVERZIJE PRI RADU SA PRINTF I SCANF

Svaki tip podatka ima sopstveni simbol konverzije koji se koristi pri ulazno izlaznim operacijama u C programskom jeziku

Na Slici 3.1 su opisani simboli konverzije koji se koriste kod ulazno/izlaznih funkcija.

Simbol	Tip Podatka	Osobine izlazne informacije
c	char	Jedan znak
d	int	Ceo dekadni broj
u	int	Ceo dekadni broj bez znaka
o	int	Ceo oktalni broj bez znaka
x,X	int	Ceo heksadekadni broj bez znaka
s	string	String – niz karaktera
f	float,double	Dekadni zapis realnog broja
e,E	float,double	Eksponencijalni zapis
g,G	float,double	Kraci zapis, izmedju %f i %e

Slika 3.1 Simboli konverzije kod ulazno izlaznih funkcija [izvor: autor]

U nastavku su dati neki od primera korišćenja simbola konverzije:

- *%d* - prikazati kao dekadni ceo broj
- *%6d* - prikazati kao dekadni ceo broj u polju od bar 6 znakova
- *%f* - prikazati kao broj sa pokretnim zarezom
- *%6f* - prikazati kao broj sa pokretnim zarezom u polju od bar 6 znakova
- *%.2f* - prikazati kao broj sa pokretnim zarezom sa dve decimalne cifre
- *%6.2f* - prikazati kao broj sa pokretnim zarezom u polju od bar 6 znakova sa dve decimalne cifre.

Pogledajmo sada sledeći primer:

```
#include <stdio.h>
void main( )
{
   char str;
   int i;
   printf( "Unesi vrednost :");
   scanf("%c %d", &str, &i);
   printf( "\nUneli ste: %c %d ", str, i);
}
```

Ovde se naravno podrazumeva da funkcija *scanf()* očekuje da ste uneli podatke u istom formatu koji je definisan korišćenjem oznaka *%c* i *%d*.

FUNKCIJE ZA RAD SA KARAKTERIMA: GETCHAR() I PUTCHAR()

Funkcija getchar() učitava prvi sledeći dostupan karakter, dok funkcija putchar() konvertuje ASCII kod u karakter, i taj karakter štampa na ekran

Za učitavanje i ispis karaktera se koriste dve funkcije:

- Funkcija *int getchar(void)* učitava sledeći dostupan karakter sa ekrana i vraća rezultat koji je tipa ***int*** (konvertuje karakter u **ASCII** kod). Ova funkcija čita samo jedan karakter, ali ju je moguće koristiti u petlji ukoliko je neophodno učitati niz karaktera sa ekrana.
- Funkcija *int putchar (int c)* konvertuje celobrojni **ASCII** kod u karakter i taj karakter štampa na ekran.

Naredni primer demonstrira upotrebu funkcija *getchar* i *putchar*:

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

   printf( "Unesi vrednost :");
   c = getchar( );

   printf( "\nUneli ste: ");
   putchar( c );

   return 0;
}
```

Nakon kompajliranja i pokretanja programa, program čeka na korisnika da unese neki tekst, a nakon unosa teksta i pritiska na **Enter**, učitava se samo prvi karakter unetog teksta. Na ekranu se štampa sledeći rezultat (za uneti tekst: ***this is test***):

```
Enter a value : this is test
You entered: t
```

Poglavlje 4

Tipovi podataka

PREGLED TIPOVA PODATAKA U C-U

Tipovi podataka u C-u mogu biti osnovni i složeni. Osnovni tipovi predstavljaju bazu za građenje složenih tipova (nizova, struktura, klasa, itd...)

Tipovi podataka u C-u su dati sledećom tabelom, Slika 4.1:

Tip	Memorija	Opseg Vrednosti
char	1	-128 ... +127
int	2 ili 4	-32768 ... +32768 ili -2,147,483,648 to 2,147,483,647
float	4	3.4E-38... 3.4E+38
double	8	1.7E-308...1.7E+308
long double	10	3.4E-4932 ... 1.1E+4932
void	-	-

Slika 4.1 Osnovni tipovi podataka u C jeziku [1]

Osnovna lista tipova može se proširiti uvođenjem sledećih modifikatora za tip ***int*** i ***char,*** Slika 4.2.

Modifikator	Tip	Memorija	Opseg Vrednosti
short	int	2	-32768 ... +32768
long	int	4	-2,147,483,648 ... +2,147,483,647
unsigned	int	2	0 ... 65535
unsigned	short		
unsigned	long int	4	0 ... 4,294,967,295
unsigned	char	1	0 ... 255

Slika 4.2 Prošireni tipovi podataka u C jeziku [1]

Da bi smo dobili informaciju o veličini memorije u bajtovima koju zauzima neki tip podataka možemo da koristimo funkciju *sizeof*.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

U nastavku je primer korišćenja funkcije *sizeof()* za određivanje veličine memorije koju zauzima tip ***int***:

```
#include <stdio.h>
int main()
{
```

```
   printf("Velicina tipa podatka int je : %d \n", sizeof(int));
   return 0;
}
```

DEFINISANJE NOVIH TIPOVA PODATAKA

U cilju definisanja novih tipova koristi se naredba typedef za kojom slede naziv osnovnog a zatim naziv novodefinisanog tipa podatka

Da bi se u programu uveli tipovi podataka koji nisu standardni deo C/C++ jezika, koristi se naredba typedef**,** čiji je opšti oblik:

```
typedef ime_tipa1 ime_tipa2;
```

Definisanje novog tipa podatka i njegovo korišćenje u C-u može biti ostvareno na sledeći način:

```
typedef int Ceo_broj;
Ceo_broj a,b,x,y=0;
/* prethodne programske linije imaju isto dejstvo kao int a,b,x,y=0; */
```

pri čemu se definiše novi tip *Ceo_broj* koji je ekvivalentan standardnom tipu ***int***. Pogledajmo još jedan primer definisanja novog tipa podatka:

```
#include <stdio.h>
int main()
{
   typedef long long int LLI;
   printf("Velicina memorije za podatak tipa long long int" \
        ": %ld \n", sizeof(LLI));
   return 0;
}
```

Rezultat programa biće:

Velicina memorije za podatak tipa long long int : 8

O definisanju novih tipova će biti više reči u lekciji o strukturama (*struct*).

NABROJIVI TIP ENUM

Korišćenjem nabrojivih tipova poboljšava se čitljivost programa i njegovo lakše razumevanje

Nabrojivi tip (enumerated type) se definiše kao uređen skup nabrojanih imena, koja predstavljaju konstante nabrojivog tipa. Rezervisana reč koja se koristi da se definiše nabrojivi tip je ***enum***, a sintaksa ima sledeći oblik:

```
enum type_name{ value1, value2,...,valueN };
```

pri čemu *type_name* predstavlja ime nabrojivog tipa, dok *value1*, *value2*,...., *valueN* predstavlja skup (ili opseg) vrednosti nabrojivog tipa *type_name*. Ukoliko se ne navede drugačije *value1* će inicijalno biti jednako *0*, *value2* će biti *1* i tako redom. Naravno, programer uvek može da izvrši dodelu vrednosti na način koji njemu najviše odgovara.

Pogledajmo jedan prost primer:

```
enum suit
{
   club=0,
   diamonds=10,
   hearts=20,
   spades=3
};
```

Pošto u programskom jeziku C ne postoji logički tip (***bool***, ***boolean***) mi ga možemo definisati primenom nabrojivog tipa, i to na sledeći način:

```
enum boolean {
    false,
    true
};
enum boolean check;
```

U prethodnom primeru smo definisali novi nabrojivi tip ***boolean***, i deklarisali smo promenljivu *check* koja je tipa ***enum boolean***.

Poglavlje 5

Promenljive i konstante

DEKLARACIJA PROMENLJIVIH

Promenljiva je objekat jezika koji ima ime i kome se može menjati sadržaj u toku izvršavanja programa, za razliku od konstanti koje dobijaju vrednost pre izvršavanja i ostaju nepromenjene

Deklaracija promenljive specificira tip podatka, i može da sadrži listu više od jedne promenljive. Opšti oblik deklaracije je sledeći:

```
type variable_list;
```

dok su u nastavku dati primeri deklaracije promenljivih različitih tipova:

```
int i, j, k;
char   c, ch;
float  f, plata;
double d;
```

Prva linija predstavlja deklaraciju promenljivih *i*, *j* i *k*, a istovremeno se šalje poruka kompajleru da u memoriji odvoji prostor za te tri promenljive celobrojnog tipa (***int***).

Promenljive mogu biti i inicijalizovane prilikom deklaracije, pri čemu se inicijalizacija sastoji iz operatora = i konstantnog izraza:

```
type variable_name = value;
```

Neki od primera inicijalizacije pri deklaraciji su dati u nastavku:

```
// deklaracija inicijalizacija d i f.
int d = 3, f = 5;
// deklaracija i inicijalizacija z.
byte z = 22;
// promenljiva x dobija vrednost 'x'.
char x = 'x';
```

Imena promenljivih su sastavljena od slova i cifara. Prvi znak mora biti slovo ili znak podvučeno (npr, _a). U jeziku C postoji razlika između malih i velikih slova (case sensitive).

KONVERZIJA PROMENLJIVIH

Ako se u aritmetičkom izrazu pojave operandi različitih tipova tada se jedan konvertuje tako da odgovara operandu drugog tipa

Ako se u nekom aritmetičkom izrazu pojave operandi različitih tipova tada se jedan konvertuje tako da odgovara operandu drugog tipa. Osnovni tipovi podataka imaju sledeći hijerarhijski poredak (desni tip se konvertuje u tip koji je levo od njega),

long double < double < float <unsigned long long < long long < unsigned long< long < unsigned int < int

Konverzija se realizuje i pri operaciji dodele vrednosti (npr. a = b). Bez obzira kakav je tip izraza desno od znaka dodele, on se pretvara u tip promenljive levo od znaka dodele.

Posmatrajmo primer gde se vrši sabiranje dva različita tipa podatka (***int*** i ***char***) i rezultat se smešta u promenljivu tipa ***int***:

```
int  i = 17;
char c = 'c'; /* ascii vrednost je 99 */
int sum;

sum = i + c;
printf("Vrednost zbira : %d\n", sum );
```

Naime, rezultat će biti 116 jer kompajler implicitno karakter ***'C'*** pretvara u njegov ASCII kod, pa tek onda taj kod sabira sa *i* i na kraju rezultat (*116*) dodeljuje promenljvoj *sum*.

Osim automatske konverzije tipova može se izvršiti i eksplicitna konverzija odnosno cast-ovanje, tj. konverzija vrednosti promenljive iz jednog tipa podatka u drugi. Opšti oblik operacije kastovanja je:

```
(tip) izraz
```

kao što možemo da vidimo u sledećem primeru:

```
int sum = 17, count = 5;
double mean;

mean = (double) sum / count;
printf("Srednja vrednost : %f\n", mean );
```

Rezultat izvršavanja programa će biti:

```
Srednja vrednost : 3.400000
```

Naime, operator konverzije (cast operator) ima prednost u odnosu na deljenje, pa je vrednost promenljive *sum* prvo konvertovana u tip ***double*** pa je tek onda podeljena sa vrednošću promenljive *count*, Kao rezultat dobijamo realan broj tipa ***double***.

DEFINISANJE KONSTANTI

Definisanje konstanti počinje pretprocesorskom naredbom #define za kojoj slede simboličko ime i vrednost konstante. Definisanje konstanti je moguće izvršiti korišćenjem ključne reči const

Konstante u C-u mogu biti definisane pomoću direktive #define pretpocesora jezika C. Definicija konstante počinje direktivom:

```
#define identifikator vrednost
```

za kojom slede simboličko ime i vrednost konstante. Vrednost konstante može biti ceo ili realan broj, kao i tekstualni podatak (niz karaktera, tj, string). Pogledajmo sledeći primer:

```
#include <stdio.h>

#define LENGTH 10
#define WIDTH  5
#define NEWLINE '\n'

void main()
{
   int area;

   area = LENGTH * WIDTH;
   printf("vrednost povrsine : %d", area);
   printf("%c", NEWLINE);
}
```

Rezultat je:

```
vrednost povrsine: 50
```

Osim pretprocesorske direktive, moguće je koristiti i prefix ***const*** u cilju deklarisanja konstanti. Opšti oblik korišćenja ključne reči ***const*** je dat u nastavku:

```
const type variable = value;
```

Poglavlje 6

Tipovi operatora

PODELA TIPOVA OPERATORA

Operator je simbol koji ukazuje kompajleru da treba da izvrši neku matematičku ili logičku operaciju. Operacije se vrše nad objektima koji se nazivaju operandi

Operatori jezika C/C++ sa mogu svrstati u nekoliko funkcionalnih grupa i one su prikazane u ovom odeljku. Pošto je najveći deo operatora isti kao i u ostalim programskim jezicima (npr. Java), o njima neće biti previše reči i detalja.

Aritmetički operatori

- negativni predznak ili oduzimanje (-)
- sabiranje (+)
- množenje (*)
- deljenje (/)
- moduo tj, celobrojni ostatak pri deljenju (%)

Operatori dodele vrednosti

- elementarna dodela vrednosti (=)
- aritmetička operacija sa dodelom vrednosti(-=, +=, *=, /=, %=)
- operacija nad bitovima sa dodelom vrednosti (<<=, >>=, &=, |=, ^=)

Operatori poređenja

- manje (<)
- manje ili jednako (<=)
- veće (>)
- veće ili jednako (>=)
- jednako (==)
- nije jednako (!=)

Logički operatori

- logičko "i" (&&)
- logičko "ili" (||)
- logička negacija (!)

Operatori umanjenja i uvećanja

- unarno dekrementiranje - umanjenje vrednosti za 1 (--)
- unarno inkrementiranje (uvećanje vrednosti za 1 (++)

Operatori nad bitovima

- pomeranje ulevo (<<)
- pomeranje udesno (>>)
- logičko "i" za niz bitova (&)
- logičko "ili" za niz bitova (|)
- isključivo logičko "ili" za niz bitova (^)
- komplement (~)

Operatori specijalne namene

- operator zarez (,)
- operator grananja (? :)
- cast operator (tip)
- operator sizeof()

U okviru operatora specijalne namene nalaze se dva operatora za rad sa pokazivačima tako će se u ovoj lekciji studenti upoznati i sa osnovama deklaracije i upotrebe pokazivača.

PRIORITET I ASOCIJATIVNOST OPERATORA

Prioritet operatora definiše način grupisanja u nekom složenom izrazu

Prioritet operatora definiše način grupisanja u nekom složenom izrazu, pa stoga utiče na način kako će se neki izraz sračunati. Neki operatori imaju veći prioritet u odnosu na ostale, npr. operator množenja ima veći prioritet od operatora sabiranja, itd.

Operator	Asocijativnost
() [] -> . ++ - -	Sa leva na desno
+ - ! ~ ++ - - (tip) * & sizeof	Sa desna na levo
* / %	Sa leva na desno
+ -	Sa leva na desno
<< >>	Sa leva na desno
< <= > >=	Sa leva na desno
== !=	Sa leva na desno
&	Sa leva na desno
^	Sa leva na desno
|	Sa leva na desno
&&	Sa leva na desno
||	Sa leva na desno
?:	Sa desna na levo
= += -= *= /= %=>>= <<= &= ^= |=	Sa desna na levo
,	Sa leva na desno

Slika 6.1.1 Prioritet i asocijativnost operatora [6]

PRIMERI UPOTREBE OPERATORA

Operatori poređenja su takođe binarni operatori jer uvek vrše poređenje dva operanda. Logički operatori omogućavaju programu da donese odluku zasnovanu na više mogućih ishoda

Prosti primeri korišćenja logičkih operatora:

```
int rezultat;
float  a=5.2;
char b='0';
rezultat = (a == 5. || a > 5.1) && (b != '\n');
/* rezultat = ( 0 || 1) && (1) = (1) &&; (1) = 1 */
```

Primeri korišćenja operatora poređenja:

```
int rezultat;
float  a=5.2, b=11.4;
rezultat = a  >=  b;  /* rezultat=0 */
rezultat = a  <  b;  /* rezultat=1 */
rezultat =  a  !=  b;  /* rezultat=1 */
```

Primer korišćenja operatora umanjenja i uvećanja:

```
int rezultat, a=3, b=3;
rezultat = a * --b; /* rezultat = 6 */
/*  umesto prethodne linije mogu se pisati sledeće dve linije:
 b = b - 1;  rezultat =  a * b;   */
b++;   /*  b = 3  */
rezultat = a-- * b; /* rezultat = 9 */
/*  umesto prethodne linije mogu
    se pisati sledeće dve linije:
 rezultat =  a * b;  a =  a - 1; */
```

6.1 Uvod u pokazivače

INDIREKTNI I ADRESNI OPERATOR

*Adresni operator & vraća adresu operanda odnosno promenljive uz koju stoji u izrazu. Indirektni operator * vraća vrednost promenljive na adresi na koju pokazuje pokazivač*

Pokazivači su možda jedna od najtežih oblasti jezika C (ali i C++). Zato ih odmah sada uvodimo i opisujemo šta znače, dok ćemo kroz naredne lekcije opisati njihovu primenu u

funkcijama i nizovima. C obezbeđuje dva operatora za rad sa pokazivačima, i to (a) adresni operator & i (b) indirektni *.

Šta je ustvari pokazivač? Pokazivač je promenljiva koja sadrži adresu druge promenljive. Drugim rečima, pokazivač je promenljiva koja pokazuje na adresu druge promenljive, odnosno pokazuje na drugu promenljivu. Promenljiva na koju pokazuje pokazivač može biti bilo koji tip podatka kao npr. objekti i strukture, pa čak može biti i pokazivač (pokazivač na pokazivač).

*	operator indirektnog (posrednog) pristupa
&	adresa

Primena: operator a

Slika 6.2.1 Operatori za rad sa pokazivačima [2]

Adresni operator &

Adresni operator & je unarni operator koji kao rezultat vraća memorijsku adresu operanda odnosno promenljive uz koju stoji u izrazu. Na primer, ukoliko je *var* celobrojna promenljiva, onda *&var* predstavlja njenu adresu. Operator & se čita kao ***"adresa od"*** što znači da će *&var* biti pročitano kao ***"adresa promenljive var"*** .

Indirektni operator *

Indirektni operator * je komplement operatora &. Operator * je unarni operator i on vraća vrednost promenljive na adresi na koju pokazuje pokazivač.

Kao što smo već spomenuli pokazivač (pointer) je promenljiva čija vrednost predstavlja adresu neke druge promenljive. Kao pri radu sa bilo kojom drugom promenljivom ili konstantom, prvo je neophodno deklarisati pokazivače da bi ih mogli koristiti.

DEKLARACIJA I PRIMENA POKAZIVAČA

Vrednost pokazivača je ustvari heksadecimalni broj koji predstavlja memorijsku adresu na kojoj će biti smeštena promenljiva na koju on pokazuje

Osnovni oblik deklaracije pokazivača (pokazivačke promenljive) je:

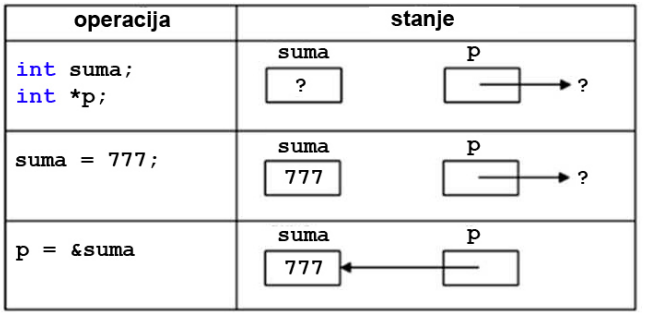
```
type *var_name;
```

Ovde se *type* odnosi na tip podatka promenljive, i to mora biti validni C/C++ tip podatka (osnovni ili složen), dok je *var_name* ime pokazivačke promenjive. Pri deklaraciji se koristi znak zvezdica *"*"* (asterisk) da bi se ukazalo da se radi o pokazivačkoj promenljivoj. U nastavku su dati primeri deklaracija pokazivača:

```
int *ip;    // pokazivač na integer
double *dp; // pokazivač na double
float  *fp; // pokazivač na float
char   *ch  // pokazivač na character
```

Bez obzira da li pokazivač pokazuje na promenljivu tipa ***int***, ***double***, ***float*** ili ***char***, vrednost pokazivača je ustvari heksadecimalni broj koji predstavlja memorijsku adresu na kojoj će biti smeštena promenljiva tipa ***int***, ***double***, ***float*** ili ***char***, na koju pokazivač ***"pokazuje"*** . Jedina razlika između pokazivača na različite tipove podatka je ta da će promenljive na koje pokazuju biti različitog tipa.

Slika 7.2 daje prikaz operacija s pokazivačem na celobrojnu promenljivu.



Slika 6.2.2 Primer deklaracije i upotrebe pokazivača [izvor: autor]

Deklarisali smo pokazivač *ptr* i promenljivu *suma*. Deklaracijom promenljive sum se rezerviše prostor u memoriji koji je određen nekom adresom. Operacijom *&suma* mi ustvari dobijamo tu adresu, a naredbom

```
ptr = &suma;
```

vrednost adrese dodeljujemo pokazivaču *ptr*. Kako sada da preko pokazivača *ptr* vidimo koja je vrednost na adresi? Pa jednostavno primenimo operator * uz pokazivač, i vrednost na koju pokazuje pokazivač prosledimo printf funkciji:

```
printf("Vrednost od var : %d\n", var);
printf("Vrednost od ptr : %x\n", ptr);
printf("Vrednost od *ptr : %d\n", *ptr);
```

Poglavlje 7

Upotreba razvojnog okruženja Visual Studio

RAD U VISUAL STUDIO-U (20 MIN)

U nastavku će biti ukratko opisan način pisanja programa korišćenjem integrisanog okruženja Visual C++

U nastavku će biti opisano integrisano razvojno okruženje (Integrated Development Environment- IDE) modula ***Visual C++*** programa ***Visual Studio Community*** , i biće prikazane opcije pisanja koda, izmena koda, kompajliranja, linkovanja i izvršavanja C/C++ programa u jednom takvom okruženju.

Visual Studio Community 2022 se može besplatno preuzeti sa sajta: https://visualstudio.microsoft.com/vs/ , dok se sa istog sajta mogu preuzeti i starije verzije, kao što su ***VS2017*** ili ***VS2019*** , i to sa sledećeg linka https://visualstudio.microsoft.com/vs/older-downloads/.

Jedan prost C/C++ projekat, koji se sastoji iz samo jednog fajla sa izvornim kodom je korišćen kao demonstracioni primer, tako da studenti treba samo da se fokusiraju na rad u ***Visual Studio*** razvojnom okruženju umesto na kod koji u ovom delu kursa može samo da ih zbuni. Radi potpunosti detalja, na kraju odeljka je dat i izvorni kod (source code) primera.

Napomena: Preporuka je da studenti sa smera za Računarske Igre koriste isključivo ***Visual Studio*** razvojno okruženje. Studenti sa ostalih smerova, ukoliko to žele, mogu da koriste druga dva dozvoljena razvojna okruženja za kreiranje C/C++ aplikacija ali njihov opis nećemo razmatrati u okviru ove lekcije. Uputstva za neophodna podešavanja i kreiranje osnovnih projekata se mogu naći na sledećim linkovima:

- za **Netbeans**:

https://netbeans.apache.org/kb/docs/cnd/quickstart.html

https://netbeans.apache.org/kb/docs/cnd/index.html

- za **CodeBlocks**:

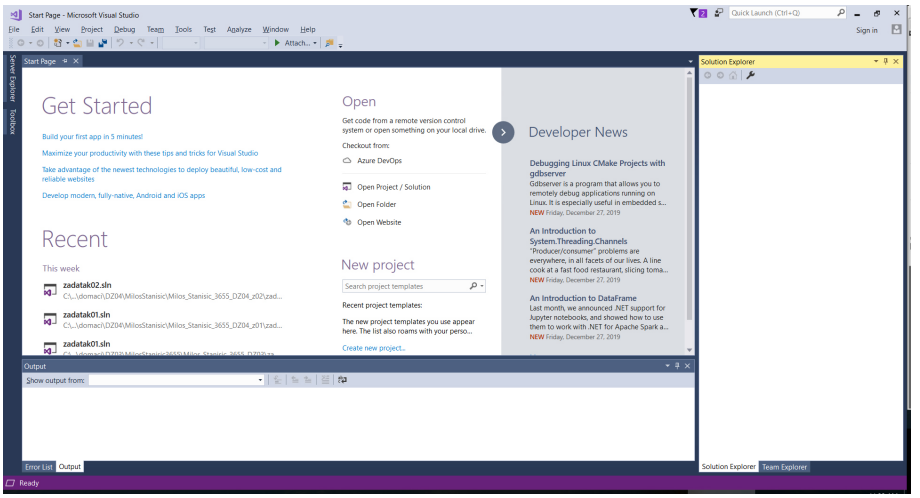
http://wiki.codeblocks.org/index.php/Creating_a_new_project

http://www.cplusplus.com/doc/tutorial/introduction/codeblocks/

KORAK 1 – POKRETANJE APLIKACIJE

Korak 1: Pokrenuti Visual Studio 2019/2022 aplikaciju sa desktop-a ili iz start menija

Korak 1: Pokrenuti ***Visual Studio 2019/2022*** aplikaciju sa desktop-a ili iz start menija. Otvoriće se glavni prozor aplikacije koji može izgledati kao na Slici 7.1:



Slika 7.1 Izgled prozora nakon koraka 1 [8]

Napomena: Detaljno uputstvo za kreiranje projekta u programskom paketu ***Visual Studio*** (modula ***Visual C++)*** se može naći na sledećim veb linkovima:

https://docs.microsoft.com/en-us/cpp/windows/walkthrough-creating-a-standard-cpp-program-cpp?view=vs-2019

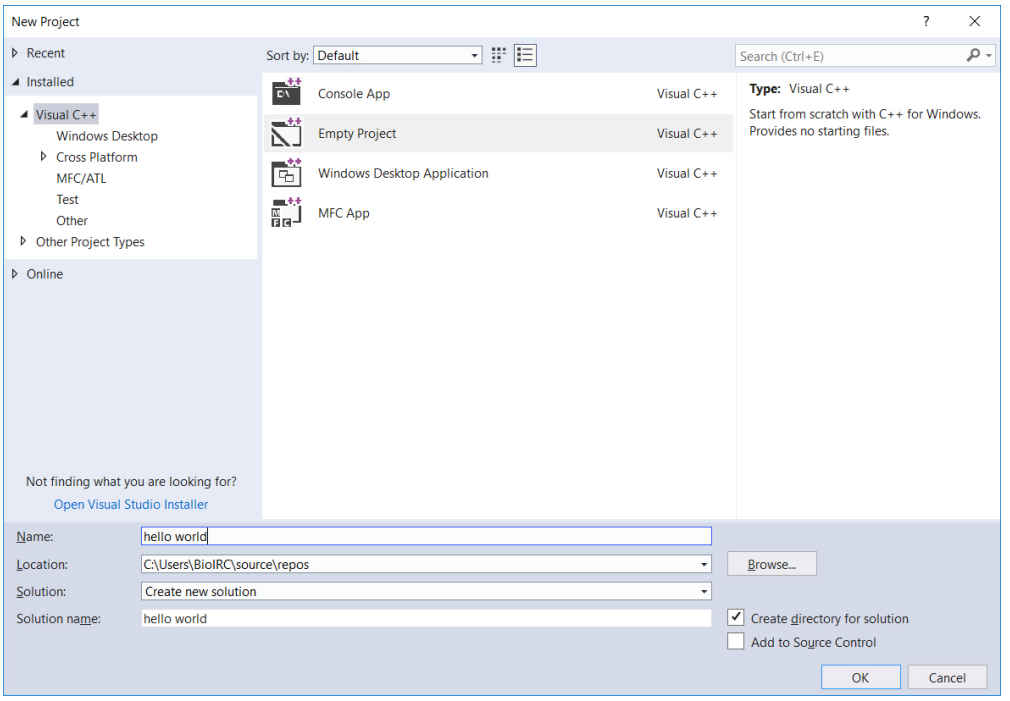
https://docs.microsoft.com/en-us/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019

https://docs.microsoft.com/en-us/cpp/get-started/tutorial-console-cpp?view=vs-2019

KORAK 2 – KREIRANJE NOVOG PROJEKTA KONZOLNE APLIKACIJE

Korak 2: Iz menu bar-a, kliknuti na File-New -Projects... da bi se otvorio New Project dijalog prikazan na Slici-2

U **New Project** dialogu prikazanom na slici ispod, selektovati **Visual C++** u okviru **Installed Template** okna a zatim **Empty Project** koji se nalazi u središnjem prozoru. Zatim u **Name** box-u treba upisati naziv projekta (npr, *hello world* kao što je prikazano), izabrati lokaciju ili direktorijum u kome će biti snimljeni fajlovi novo-kreiranog projekta klikom na dugme **Browse...**

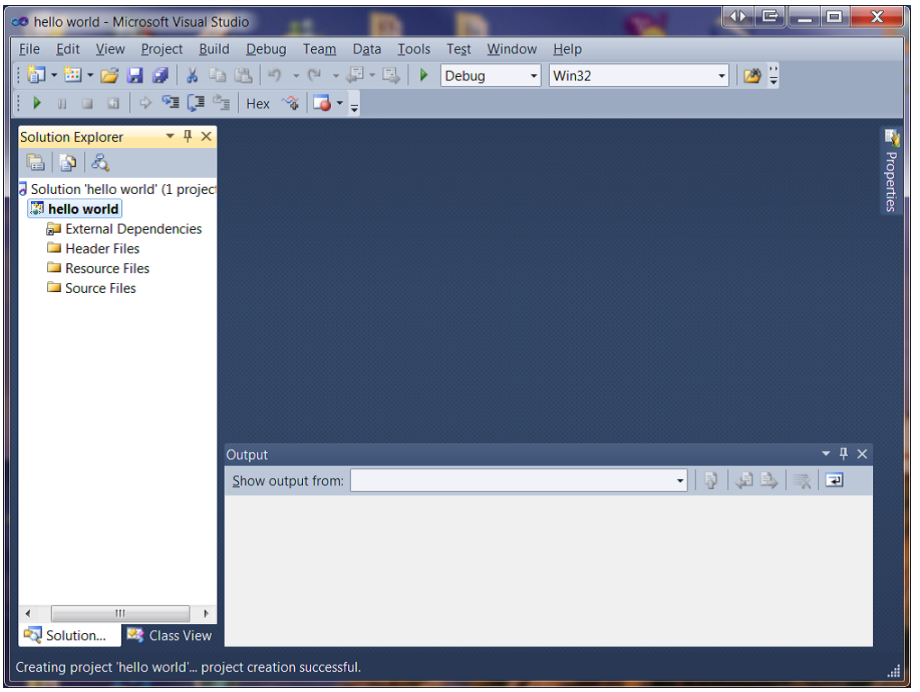


Slika 7.2 Prozor New Project [8]

Zatim kliknuti na dugme **OK** i možete da pristupite narednom koraku

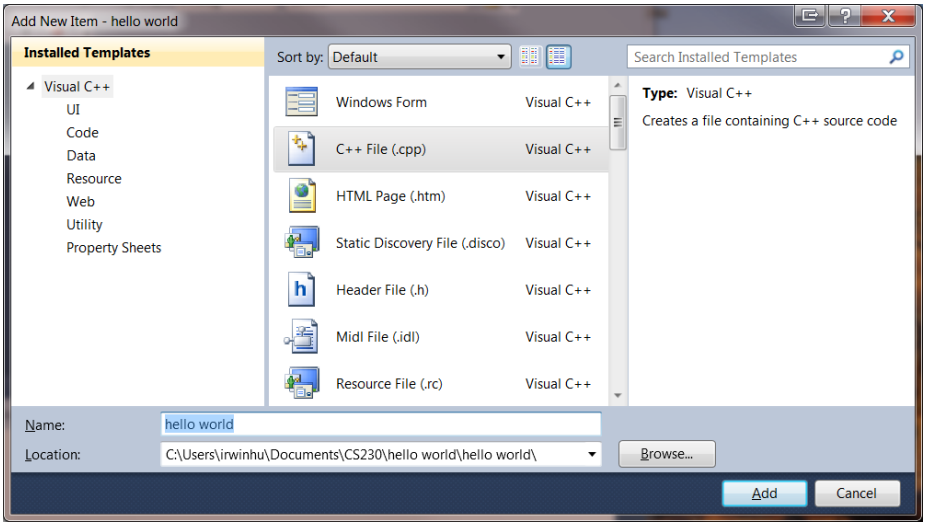
KORAK 3 – DODAVANJE IZVORNIH FAJLOVA U PROJEKAT

Korak 3: Sada će programsko okruženje Visual Studio aplikacije imati sledeći izgled (Slika-3)



Slika 7.3 Izgled programa Visual Studio nakon kreiranja praznog projekta [8]

Postaviti kursor miša na direktorijum **Source Files** okviru **Solution Explorer** okna koje se nalazi na levoj strain aplikacije. Pritiskom na desni taster miša otvoriće se popup meni, gde treba kliknuti na **Add** pa onda **New Item**... nakon čega će se otvoriti **Add New Item – *hello world*** dijalog:



Slika 7.4 Izgled dijaloga Add New Item [8]

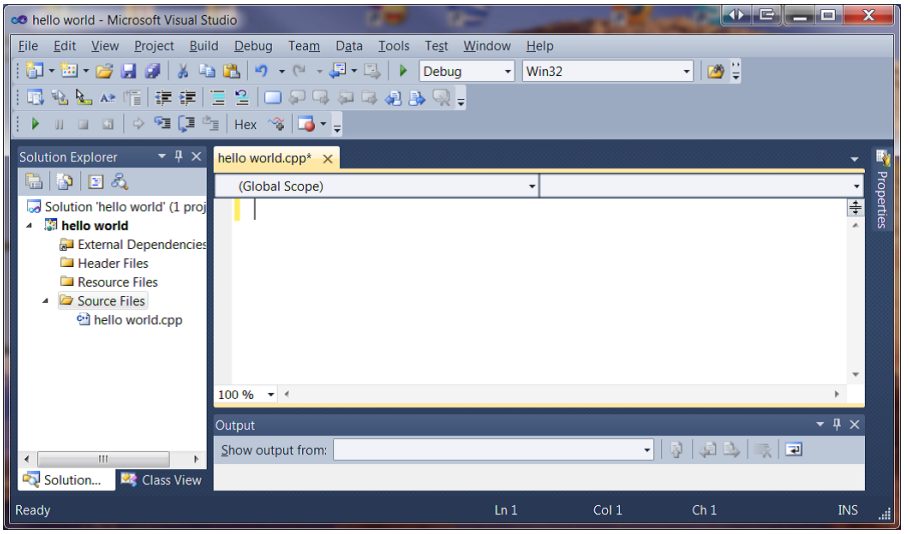
Selektovati **C++ Files (.cpp)** klikom na opciju u središnjem prozoru i izabrati proizvoljno ime fajla (npr, *hello world*, isto ime koje smo koristili za ime projekta). Pritisnuti dugme **Add** da bi ste prešli na sledeći korak.

Napomena: Za rad sa programskim jezikom C treba promeniti ekstenziju fajla ***.cpp*** u ***.c*** .

KORAK 4 – PISANJE KODA U EDITORU RAZVOJNOG OKRUŽENJA

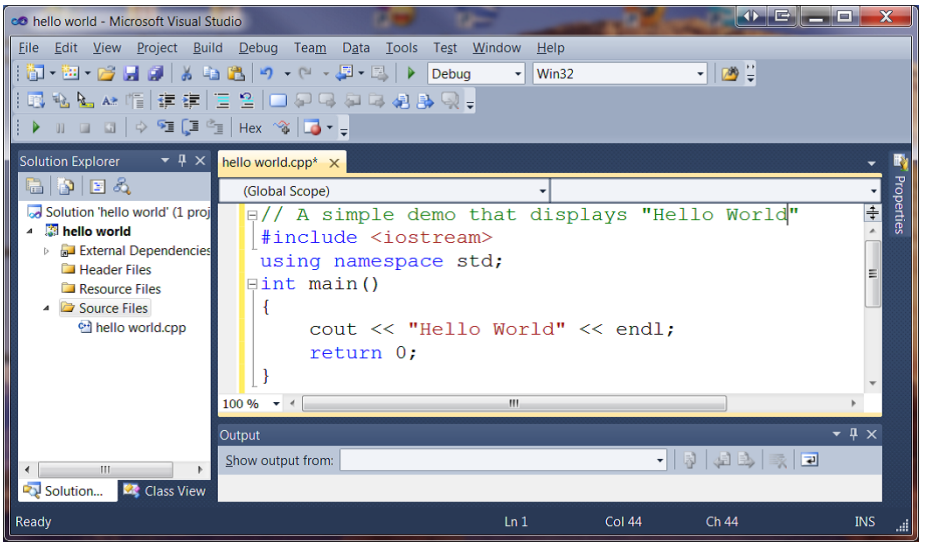
Korak 4: Na ekranu će biti prikazan prozor kao na Slici-5. U središnjem delu programa se nalazi tekst editor u kome kucate vaš C/C++ kod

Korak 4: Na ekranu će biti prikazan prozor kao na Slici 7.5. U okviru prozora **Solution Explorer** može se primetiti 1) direktorijum **Source Folder** (proveriti da li je aktivan tab **Solution Explorer Class View** taba) u kome se nalazi *hello world.cpp* fajl koji smo upravo kreirali i 2) prazan tekstualni editor sa naslovom ***hello world.cpp*** u kome možete kucati vaš C/C++ izvorni kod.



Slika 7.5 Otvaranje editora duplim klikom na ime fajla “hello world” [8]

Nakon unosa koda prozor će imati sledeći izgled:



Slika 7.6 Izgled tekstualnog editor-a nakon kucanja koda [8]

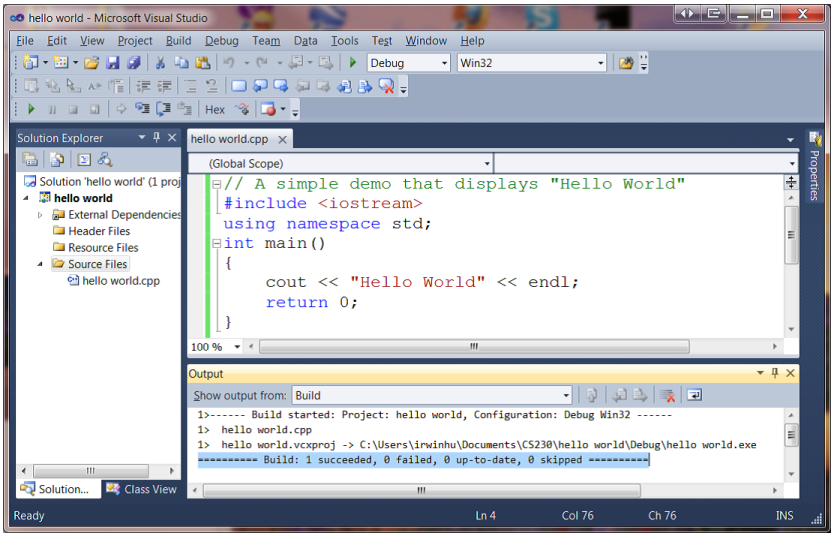
Napomena: U slučaju rada sa projektom koji se sastoji iz više izvornih fajlova, neophodno je u direktorijum **Source Files**, odnosno **Header files** dodati nove izvorne fajlove (***.c***) ili fajlove zaglavlja (***.h***). Pokušati kompajliranje fajlova ***add.c*** i ***main.c*** u okviru ***Sekcije 4 – Program sa više fajlova*** .

KORAK 5 – KOMPAJLIRANJE I IZVRŠAVANJE APLIKACIJE

Korak 5: Pokretanje programa se vrši klikom na Debug iz menija pa onda klikom na Start Without Debugging

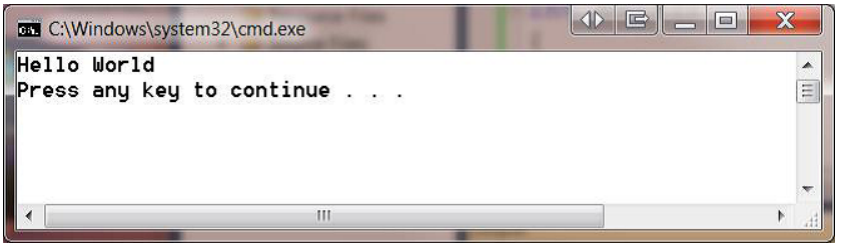
Korak 5: Da bi ste kompajlirali, povezali (linkovali) i izvršili program kliknuti na **Debug** u meni baru, a zatim na **Start Without Debugging** u okviru padajućeg menija. Ukoliko ne postoji greška u vašem kodu biće prikazana sledeća poruka u okviru prozora **Output** kao što je prikazano na Slici 7.7.

=== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped == =



Slika 7.7 Izveštaj o uspešnom kompajliranju [8]

Takođe, u posebnom prozoru (konzoli - C:\Windows\system32\cmd.exe) biće prikazan izlaz programa:



Slika 7.8 Izgled konzole nakon startovanja programa [8]

U nastavku je dat izvorni koda za prethodno obrađeni primer:

Ime fajla: ***Hello World.cpp***

```
#include <iostream>
Using namespace std;
int main()
{
 cout << "\Hello World!" << endl;
 return 0;
}
```

Napomena: Umesto prethodnog C++ koda moguće je zalepiti (paste) bilo koji C program sa predavanja i probati ponovo celokupan postupak do izvršavanja programa.

KORIŠĆENJE KOMANDNE LINIJE

Pogledajte sledeće linkove za kompajiranje programa iz konzole

Dodatna objašnjenja za kompajliranje C programa korišćenje komandne linije možete pronaći na linku

- **Compiling a C Program on the Command Line** https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2013/bb384838(v=vs.120)?redirectedfrom=MSDN,

dok za opis kompajliranja C++ programa korišćenjem komandne linije možete posetiti sledeći link:

Compiling a Native C++ Program on the Command Line - https://docs.microsoft.com/en-us/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line?redirectedfrom=MSDN&view=msvc-160.

Poglavlje 8

Vežbe

PRIMERI 1 (10 MIN)

Ulaz / izlaz sa ASCII kodovima

U nekom od prethodnih primera, uneli smo karakter u program ali on nije zapamćen u obliku ASCII koda jer smo koristili konverziju *%c*. Prilikom štampanja smo takođe koristili konverziju *%c*, pa je izvršeno štampanje istog tog karaktera. Ukoliko prilikom štampe koristimo konverziju *%d* biće oštampan ASCII kod karaktera:

```
#include <stdio.h>
int main(){
    char var1;
    printf("Uneti karakter: ");
    scanf("%c",&var1);
    printf("Uneli ste %c.\n",var1);
/* \n prebacuje kursor u sledeci red(radi isto kao enter). */
    printf("ASCII vrednost je %d",var1);
    return 0;
}
```

Izlaz programa je

```
Uneti karakter: g (enter)
Uneli ste g.
ASCII vrednost je 103
```

Kada unosemo *'g'* čuva se ASCII kod *103* umesto karaktera *g*. U zavisnosti od tipa konverzije vrši se ili štampanje karaktera ili ASCII koda. U nastavku je dat primer konverzije ASCII koda u karakter prilikom štampe.

```
#include <stdio.h>
int main(){
    int var1=69;
    printf("Karakter cije je ASCII jednak 69: %c",69);
    return 0;
}
```

Izlaz prethodnog programa je:

```
Karakter cije je ASCII jednak 69: E
```

ASCII vrednost karaktera *'A'* je *65*, 'B' je *66* i tako dalje do *'Z'* čiji je ASCII kod *90*. Slično tome, ASCII vrednost karaktera *'a'* je *97*, *'b'* je *98*, ..., a slova 'z' je *122*.

PRIMERI 2 (10 MIN)

Varijacije štampanja realnih i celobrojnih konstanti u C-u

Primer. Varijacije štampanja realnih i celobrojnih konstanti u C-u:

```
#include<stdio.h>
int main()
{
    printf("Slucaj 1:%6d\n",9876);
/*  Stampa broj desno poravnat, u polju sirine 6  */
    printf("Slucaj 2:%3d\n",9876);
/* Stampa broj desno poravnat u polju sirine 3, ali posto
   broj ima 4 cifre onda nece biti desno poravnat  */
    printf("Slucaj 3:%.2f\n",987.6543);
/* Stampa broj zaokruzen na dve decimale */
    printf("Slucaj 4:%.f\n",987.6543);
/* Stampa broj zaokruzen na 0 decimala, tj. zaokruzen na ceo broj */
    printf("Slucaj 5:%e\n",987.6543);
/* Stampa broj u eksponencijalnom formatu(scientific notation) */
    return 0;
}
```

Izlaz programa je:

```
Slucaj 1:  9876
Slucaj 2:9876
Slucaj 3:987.65
Slucaj 4:988
Slucaj 5:9.876543e+002
```

PRIMERI 3 (10 MIN)

Upotreba ulazno izlaznih operacija nad karakterima

Primer upotrebe *putchar* i *getchar* funkcija za rad sa karakterima

```
#include <stdio.h>
void main()
{
 int c1, c2;
 c1 = getchar();
 printf("------------\n");
 c2 = getchar();
 printf("c1 = %d, c2 = %d\n",c1, c2);
 printf("c1 = %c, c2 = %c\n",c1, c2);
```

```
 putchar(c1); /* isto je kao i printf("%c",c1); */
 putchar(c2); /* isto je kao i printf("%c",c2); */
 putchar('\n');
 /* Za ispisivanje karaktera a */
 putchar('a');
 /* dozvoljeno je : printf("abc"); printf("a"); */
 /* nije dozvoljeno : printf('a'); putchar('abc'); putchar("abc"); */
}
```

U prethodnom primeru vršimo učitavanje karaktera iz konzole primenom funkcije getchar, smeštamo ih u promenljive c1 i c2. Vrednosti koje se nalaze u c1 i c2 prikazujemo na dva različita načina: primenom printf i putchar. Ostalo se može zaključiti na osnovu komentara koji su u kodu.

PRIMERI 4 (15 MIN)

U nastavku su dati razni primeri aritmetičkih operacija nad promenljivama istog ili različitog tipa, kao i primeri sa konverzijom karaktera u njegov ASCII kod

Primer 1: Zbir dva broja istog tipa

```
#include <stdio.h>
void main()
{
   int x,y;   // deklaracija ide na pocetku
   x=23; // inicijalizacija vrednosti
   y=56; // inicijalizacija vrednosti
   printf("Njihov zbir je %d\n",x+y);  // stampanje vrednosti
}
```

Primer 2: Zbir dva broja različitog tipa

```
#include <stdio.h>
void main()
{
  int x=23;float y=5.6;
  printf("Njihov zbir je %f\n",x+y);
}
```

Primer 3: Množenje dva velika broja

```
#include <stdio.h>
void main()
{
  long x=2345678,y=67890;
  printf(" Njihov proizvod je %ld\n",x*y);
}
```

Primer 4: Štampanje karaktera i njegovog ASCII koda

```
#include <stdio.h>
void main()
{
 char c='A';
 printf(" %c %d\n",c,c);
}
```

PRIMERI 5 - ADRESNI I INDIREKTNI OPERATOR (15 MIN)

*Adresni operator & određuje adresu promenljive, dok indirektni operator * određuje podatak koji se nalazi na adresi na koju pokazuje pokazivač*

Vec smo spomenuli da se svaka promenljiva čuva na memorijskoj lokaciji koju određujemo korišćenjem adresnog operatora &. U nastavku je dat primer koji štampa adresu definisanih promenljivih:

```
#include <stdio.h>

int main()
{
   int  var1;
   char var2[10];

   printf("Adresa promenljive var1: ");
   printf("%x\n",&var1);

   printf("Adresa promenljive var2: ");
   printf("%x\n",&var2);

   return 0;
}
```

Nakon kompajliranja i izvršavanja programa, na ekranu se dobija sledeći rezultat:

```
Adresa promenljive var1: 0xbfebd5c0
Adresa promenljive var2: 0xbfebd5b6
```

Primer: Definisati pokazivač na celobrojnu promenljivu kojoj je dodeljena neka proizvoljna vrednost. Odštampati adresu promenljive i njenu vrednost korišćenjem pokazivača:

```
#include <stdio.h>

int main ()
{
   int  var = 20;   /* deklaracija stvarne promenljive*/
```

```
   int  *ip;        /* deklaracija pokazivača*/

   ip = &var;  /* smesti adresu promenljive var u pokazivačku promenljivu*/

   printf("Adresa promenljive var je: %x\n", &var  );

   /* adresa sačuvana u pokazivaču */
   printf("Adresa smestena u pokazivaču ip: %x\n", ip );

   /* pristupi vrednosti primenom pokazivača */
   printf("Vrednost *ip promenljive: %d\n", *ip );

   return 0;
}
```

Nakon izvršavanja programa dobija se sledeći rezultat:

```
Adresa promenljive var: bffd8b3c
Adresa smestena u pokazivacu ip: bffd8b3c
Vrednost *ip promenljive: 20
```

ZADACI ZA PROVERU ZNANJA (30 MIN)

Pokušajte sami:

1. Šta će biti rezultat sledećeg koda?

```
#include<stdio.h>
void main()
{
    char c = 'A'+255;
    printf("%c", c);
}
```

A - A
B - B
C – greška u toku izvršavanja programa - proboj gornje granice
D – kompajlerska greška

2. Šta će biti rezultat sledećeg koda.

```
#include<stdio.h>
void main()
{
   printf("%d", -11%2);
}
```

A - 1
B - -1

C - 5.5
D - -5.5
3. Pronaći grešku u svakom od sledećih iskaza i objasniti kako greška može biti korigovana.
a) Sledeći iskaz treba da odštampa karakter ’c’.
printf("%s\n", ’c’);
b) Sledeći iskaz treba da odštampa 9.375%.
printf("%.3f%", 9.375);
c) Sledeći iskaz treba da odštampa prvi karakter stringa "Monday".
printf("%c\n", "Monday");
d) printf(""A string in quotes"");
e) printf(%d%d, 12, 20);
f) printf("%c", "x");
g) printf("%s\n", ’Richard’);
4. Napisati iskaz za svaku od sledećih stavki:
a) Odštampati 1234 uz desno poravnanje, u polju širine 10.
b) Odštampati 123.456789 u ekponencijalnom obliku sa tačnošću na tri decimale.
c) Učitati vrednost tipa double korišćenjem promenljive number.
d) Odštampati 100 u oktalnom obliku.
e) Učitati string u niz karaktera pod nazivom string.

i) Odštampati 3.333333 korišćenjem tipa long double u polju širine 20 karaktera sa preciznošću na 3 decimale 3.
5. Pronaći grešku (ili greške) u svakoj od sledećih linija koda. Opisati kako se svaka greška može korigovati.
a) printf("%s\n", ’Happy Birthday’);
b) printf("%c\n", ’Hello’);
c) printf("%c\n", "This is a string");
d) Sledeći iskaz treba da oštampa poruku "Bon Voyage":
printf(""%s"", "Bon Voyage");
e) char day[] = "Sunday";
printf("%s\n", day[3]);
f) printf(’Enter your name: ’);
g) printf(%f, 123.456);
h) Sledeći iskaz treba da oštampa karaktere ’O’ i ’K’:
printf("%s%s\n", ’O’, ’K’);
i) char s[10];
scanf("%c", s[7]);
6. Napisati program koji za uneti karakter ispisuje njegovu numeričku vrednost. Uraditi ovo korišćenjem scanf i printf. (10 min)

Poglavlje 9

Zadaci za samostalan rad

ZADACI ZA SAMOSTALNO VEŽBANJE

Korišćenjem materijala sa predavanja i vežbi za ovu nedelju uraditi sledeće zadatke:

Zadatak 1. Napisati program koji ispisuje poruku ***"Zdravo /*\"*** (svi karakteri su vidljivi prilikom ispisa). (10 min)

Zadatak 2. Napisati C program koji će da ispisuje ***“Dobrodošli na C/C++”*** 5 puta. (10 min)

Zadatak 3. Napisati program koji za unete stranice A i B, računa površinu i obim pravougaonika i ispisuje ih na standardnom izlazu. (10 min)

Zadatak 4. Napisati program koji za unet poluprečnik, računa površinu i obim kruga. (10 min)

Zadatak 5. Napisati program koji učitava podatke za poluprečnik r osnove kupe i izvodnicu s. Prikazuju se podaci za B, M, P i V prave kupe (10 min)

Zadatak 6. Napisati program koji za uneti karakter ispisuje njegovu numeričku vrednost. Uraditi ovo korišćenjem scanf i printf. (10 min)

Zadatak 7. Definisati PI kao konstantu preprocesora; Napisati program koji za uneti poluprečnik izračunava i prikazuje obim i povšinu kruga korišćenjem konstante PI. (10 min)

Zadatak 8. Napisati program koji ispisuje vrednost broja Pi sa 5 decimalnih mesta. (Pomoc: pogledati sadrzaj zaglavlja math.h pomocu man komande). (10 min)

Zadatak 9. Napisati program koji računa godišnju kamatu na iznos depozita u banci. Unosimo količinu novca i kamatnu stopu na mesečnom nivou, a aplikacija nam kaže koliko smo novca zaradili od kamate za godinu dana. Razmisliti koje tipove podataka je optimalno koristiti za ovaj problem. Oštampati rezultat na 3 i 5 decimala, kao i u eksponencijalnom obliku. (10 min)

Zadatak 10. Napisati program u koji se unosi broj od 1 do 7, a on ispisuje na konzoli dan u nedelji. Ako je unet neki drugi broj, program ispisuje grešku. (10 min)

Zadatak 11. Kreirati nabrojivi tip za dane u nedelji i ispitati da li je u pitanju radni dan ili vikend. (10 min)

Zadatak 12. Napisati program u kojem se prvo deklarise promenljiva slovo, a zatim se njena vrednost inicijalizuje na A i ispisuje na ekran. (10 min)

Poglavlje 10

Domaći zadatak

PRAVILA ZA DOMAĆI ZADATAK

Detaljno proučiti pravila za izradu domaćih zadataka

Svaki student dobija od asistenta sopstvenu kombinaciju domaćeg zadatka.

Onlajn studenti bi trebalo mejlom da se najave, kada budu želeli da krenu sa radom na predmetu i prikupljanjem predispitnih obaveza.

Odgovarajući Visual Studio (NetBeans ili CodeBlocks) projekat koji predstavlja rešenje domaćeg zadatka smestiti u folder CS130-DZ01-Ime-Prezime-BrojIndeksa. Zipovani folder CS130-DZ01-Ime-Prezime-BrojIndeksa poslati predmetnom asistentu (lazar.mrkela@metropolitan.ac.rs) u mejlu sa naslovom (subject)CS130-DZ01, inače se neće računati.

Studenti iz Niša predispitne obaveze predaju asistentima u Nišu (sofija.ilic@metropolitan.ac.rs, mihajlo.vukadinovic@metropolitan.ac.rs, i uros.lazarevic@metropolitan.ac.rs).

Student tradicionalne nastave ima 7 dana, od dana kada je dobio mail sa domaćim zadatkom, da uradi i pošalje rešenje za maksimalan broj poena. Ukoliko student pošalje domaći nakon tog roka, najviše može da ostvari 50% od maksimalnog broja poena.

Studenti onlajn nastave imaju rok da predaju rešene domaće zadatke 10 dana pre termina ispita u ispitnom roku u kome polažu CS130 C/C++ programski jezik.

Vreme izrade: 1,5h.

Zaključak

REZIME

Na osnovu svega obrađenog možemo izvesti sledeći zaključak:

Svaki C/C++ program mora imati jednu i samo jednu glavnu funkciju, funkciju *main()*. Instrukcije programa se pišu između dve velike zagrade, a instrukcije se u principu završavaju sa znakom tačka-zarez.

Neke informacije su numeričke, druge su tekstualne. Osnovni tipovi podataka su celi brojevi, decimalni brojevi i tekstovi. Međutim, svi tipovi podataka imaju zajedničku karakteristiku a to je veličina tipa podatka koja se meri u bajtovima. Veličina tipa podatka istovremeno predstavlja i njegov opseg, odnosno najmanji i najveći broj koji može da bu smešten u taj tip podatka.

Promenljive se koriste u dve svrhe. One obezbedjuju pristup odgovarajućoj informaciji, a takođe rezervišu deo memorije koja je neophodna da se smesti neka informacija. Neophodno je deklarisati promenljivu pre njenog korišćenja. Možete koristiti adresni operator &, da bi ste odredili adresu promenljive. Svrha promenljive je da se pomoću nje skladišti neki podatak u memoriji. Stoga je prvi logični korak nakon kreiranja promenljive, da se specificira informacija koja će biti smeštena na adresu rezervisanu od strane promenljive.

Najveći deo računara ima za cilj da obavlja složene proračune i operacije. Računar, osim mogućnosti da skladišti ogromnu količinu podataka, takođe ima mogućnost da izračuna operacije mnogo brže nego što to mi možemo. C podržava rad sa svim aritmetičkim operacijama.

REFERENCE

Korišćena literatura

[1] Jeff Kent , C++: Demystified: A Self-Teaching Guide, McGraw-Hill/Osborne, 2004.

[2] Nenad Filipović, Programski jezik C, Tehnički fakulet u Čačku, Univerzitet u Kragujevcu, 2003.

[3] Milan Čabarkapa, C - Osnovi programiranja, Krug, Beograd, 2003.

[4] Paul J Deitel, Harvey Deitel, C - How to program, 7th edition, Pearson, 2013.

[5] http://www.tutorialspoint.com/cprogramming/index.htm

[6] http://www.codingunit.com/category/c-tutorials

[7] www.learncpp.com/

[8] https://docs.microsoft.com/en-us/visualstudio/ide/create-new-project?view=vs-2017