VEŽBA 1 - Okruženje Microsoft Visual Studio 2015

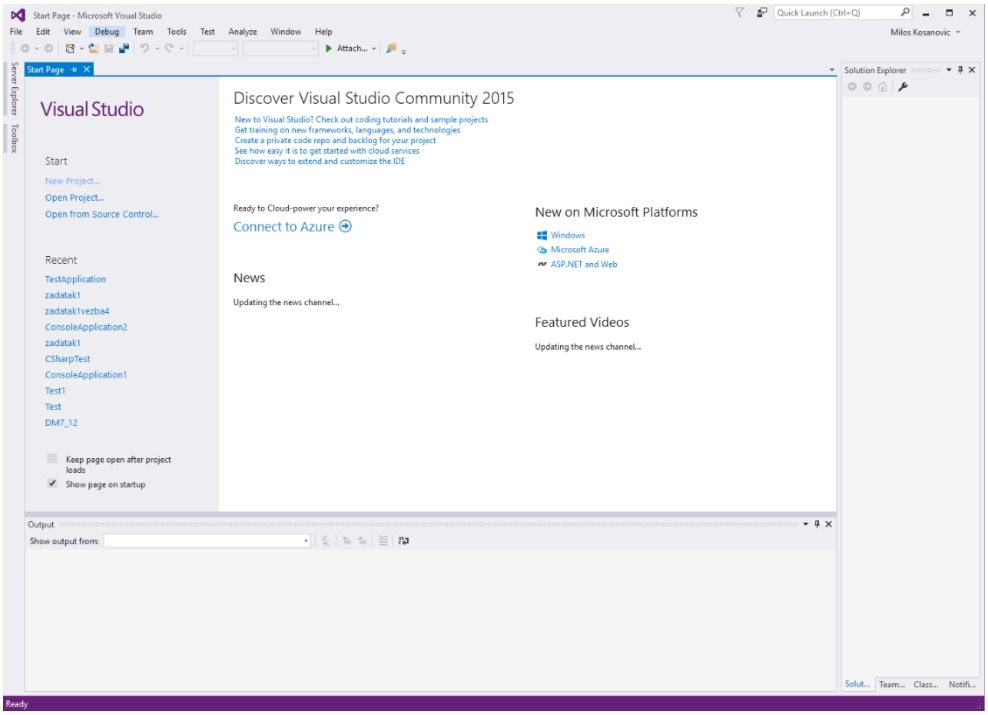
Microsoft Visual Studio 2015 predstavlja integrisano okruženje za razvoj softvera. Sastoji se od sledećih celina:

- *Visual C++*,
- *Visual Basic*,
- *Visual C#*
- *MSDN Library*.
- *Druge…*

Visual studio C++ je okruženje koje omogućava razvoj i pisanje programa u programskom jeziku C++, a samim tim i u programskom jeziku C.

Pokretanje Visual Studio 2015 okruženja

Okruženje *Visual Studio 2015* se može otvoriti klikom na taster *Start* i izborom stavke *Microsoft Visual Studio 2015* iz liste programa. Nakon startovanja otvoriće se osnovi prozor okruženja *Visual Studio 2015* koji je prikazan na slici 1.

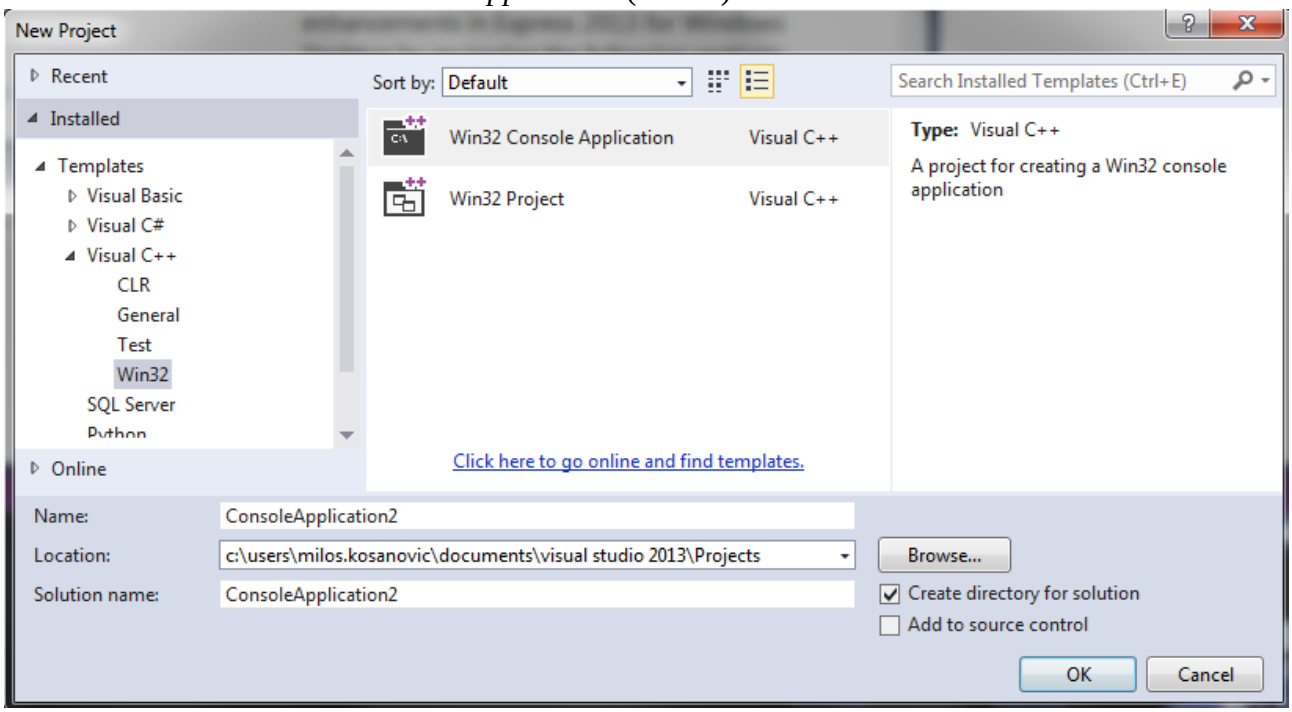


Slika 1. Izgled osnovnog prozora po otvaranju Visual C++-a

Kreiranje projekta za konzolnu aplikaciju

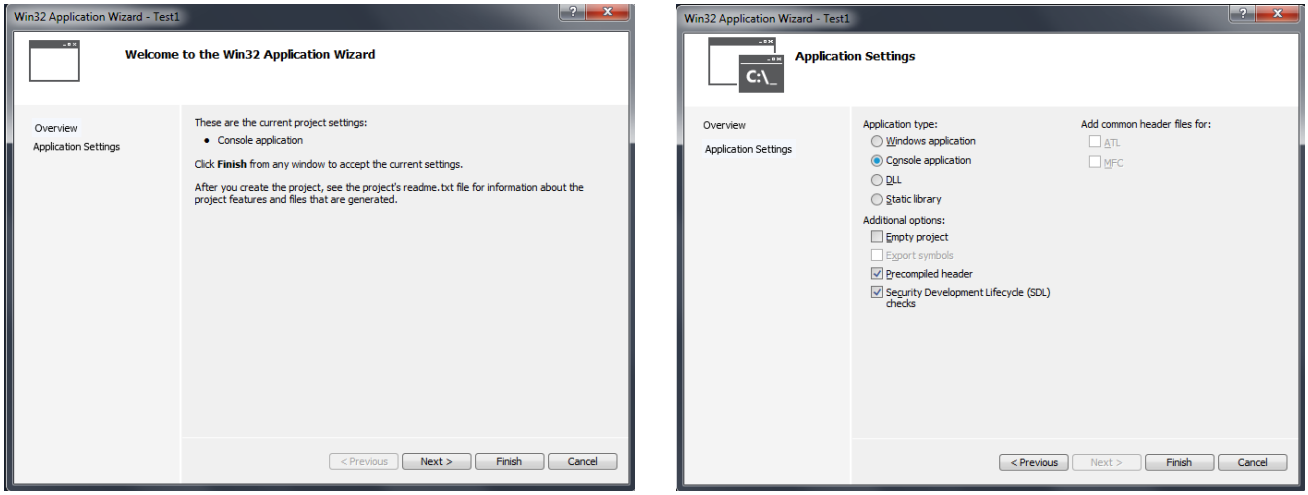
Da bi se napisao, kompilirao, izvršio i eventualno debagirao C program, u *Visual Studiju* je potrebno kreirati projekat koji sadrži odgovarajuće datoteke sa izvornim kodom programa. Za potrebe pisanja programa u C-u biće korišćen najjednostavniji tip projekta koji obezbeđuje pravljenje konzolnih aplikacija. Konzolne aplikacije su programi koji se izvršavaju u komandnom (DOS) prozoru i koriste standardni ulaz i izlaz.

Kreiranje projekta se vrši startovanjem stavke iz menija *File/New*, izborom kartice *Project* i stavke *Visual C++ i zatim Win32 Console Application* (slika 2).



Slika 2. Izgled prozora za kreiranje novog projekta

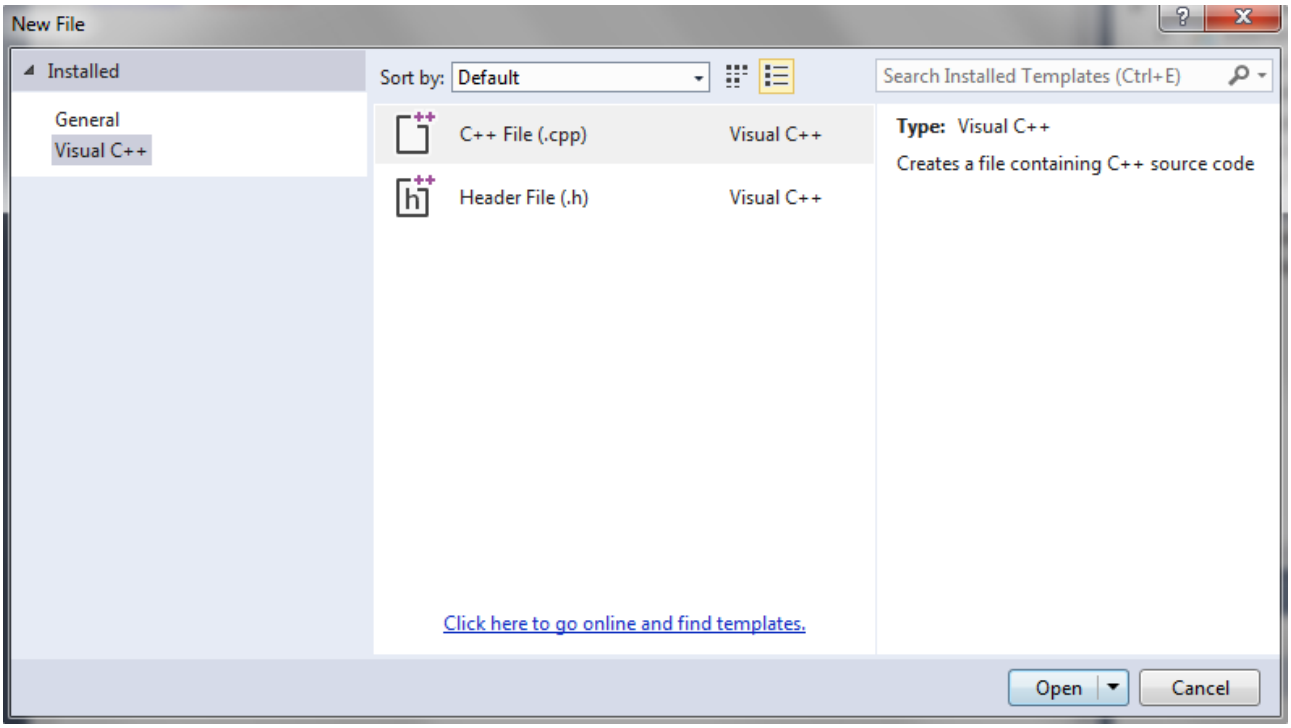
Pri kreiranju projekta potrebno je uneti naziv projekta u polje *Project name*, i eventualno promeniti direktorijum na disku gde će projekat biti kreiran (polje *Location*). Po unosu naziva projekta aktiviraće se taster OK. Klikom na ovaj taster pojaviće se prozor za izbor tipa konzolnog projekata koji se želi kreirati (slika 4). U ovom slučaju ne treba ništa menjati jer je već odabrana opcija za kreiranje praznog projekta (*An empty project*). Da bi se kreirao projekat potrebno je kliknuti na taster *Finish*. Ukoliko želite možete kliknuti i na dugme *next*. U tom slučaju će Vam biti prikazan sledeći prozor sa dodatnim opcijama.



Slika 3. Izgled prozora za izbor tipa konzolnog projekta koji će biti kreiran

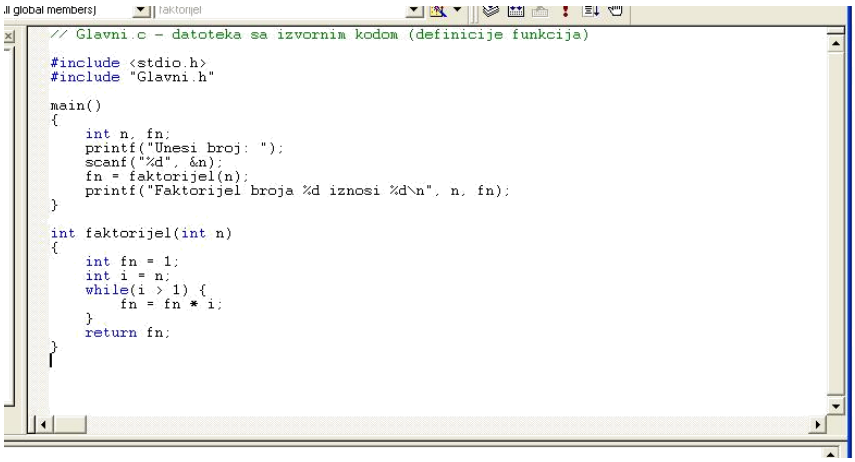
Kreiranje i dodavanje datoteka za izvorni kod programa

Nakon kreiranja praznog projekta za konzolnu aplikaciju potrebno je kreirati i dodati u projekat datoteke koje će se koristiti za unos izvornog koda programa. U VS2015 obično je za Vas već kreiran fajl koji se zove isto kao vaš projekat. Ukoliko nije, potrebno je kreirati i dodati datoteku na sledeći način. Pozivate stavku iz menija *File/New File* i stavke Visual *C++*. Zatim možete izabrati između *C++ File* ili *Header File* (slika 4). Potrebno je uneti željeni naziv datoteke u polje *File name* i kliknuti na taster *OK*.



Slika 4. Izgled prozora za kreiranje i dodavanje datoteka za izvorni kod programa

Po kreiranju i dodavanju nove datoteke u projekat potrebno je uneti odgovarajući sadržaj (ukucati program). Za to se koristi centralni deo glavnog prozora (slika 5). *Visual Studio* poseduje neke napredne opcije za prikaz izvornog koda programa kao što su bojenje ključnih reči i komentara.



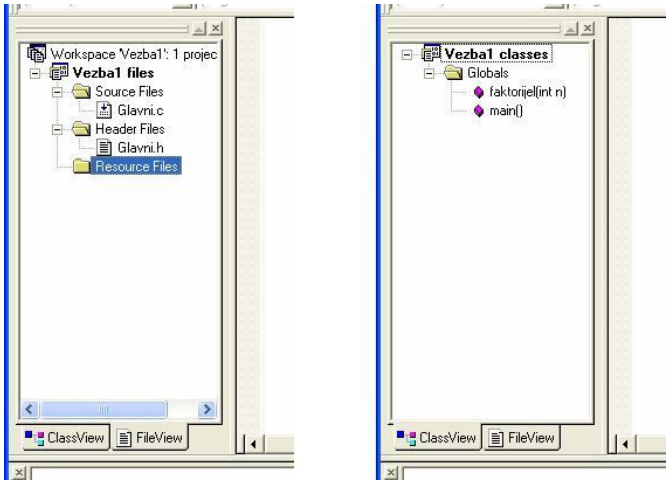
```
// Glavni.c - datoteka sa izvornim kodom (definicije funkcija)

#include <stdio.h>
#include "Glavni.h"

main()
{
    int n, fn;
    printf("Unesi broj: ");
    scanf("%d", &n);
    fn = faktorijel(n);
    printf("Faktorijel broja %d iznosi %d\n", n, fn);
}

int faktorijel(int n)
{
    int fn = 1;
    int i = n;
    while(i > 1) {
        fn = fn * i;
    }
    return fn;
}
```

Slika 5. Deo za unos i prikaz sadržaja datoteka sa izvornim kodom

Prikaz i otvaranje datoteka i funkcija koje projekat sadrži

Za prikaz datoteka koje su uključene u projekat koristi se panel sa leve strane osnovnog prozora *Visual C++*-a. Ovaj panel sadrži dva kartice *ClassView* i *Solution Explorer*. Kartica *Solution Explorer* omogućava pregled i otvaranje datoteka koje su dodate u projekat. Datoteke su grupisane po tipu koji je određen ekstenzijama (*Source Files* - *.cpp datoteke, *Header Files* - *.h datoteke). Duplim klikom na naziv datoteke, sadržaj odgovarajuće datoteke će biti prikazan u centralnom delu prozora *Visual C++*-a. Naziv datoteke koja je trenutno otvorena prikazan je u naslovnoj liniji glavnog prozora.

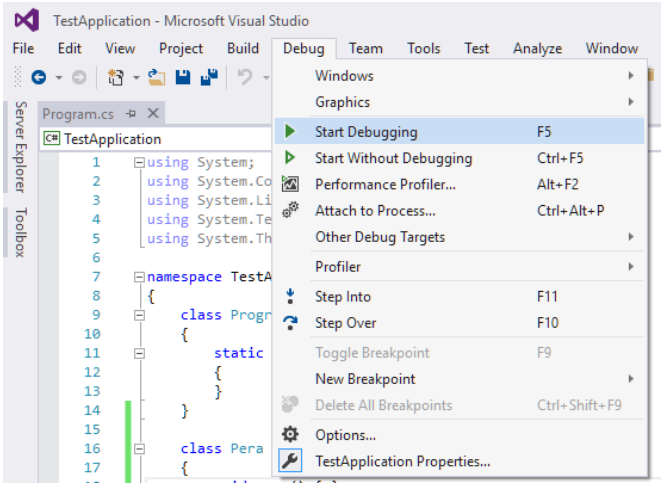
Kartica *ClassView* je izvorno namenjena za prikaz informacija o klasama definisanim u C++ programu. Kako programski jezik C ne podržava rad sa klasama u prikazu vezanom za ovu karticu pobrojane su samo funkcije definisane u datotekama koje projekat sadrži. Duplim klikom na naziv funkcije otvara se odgovarajuća datoteka i postavlja se na početak odgovarajuće funkcije. Na slici 6 ilustrovani su *FileView* i *ClassView* prikazi.



Slika 6. Prikaz datoteka i funkcija koje projekat sadrži

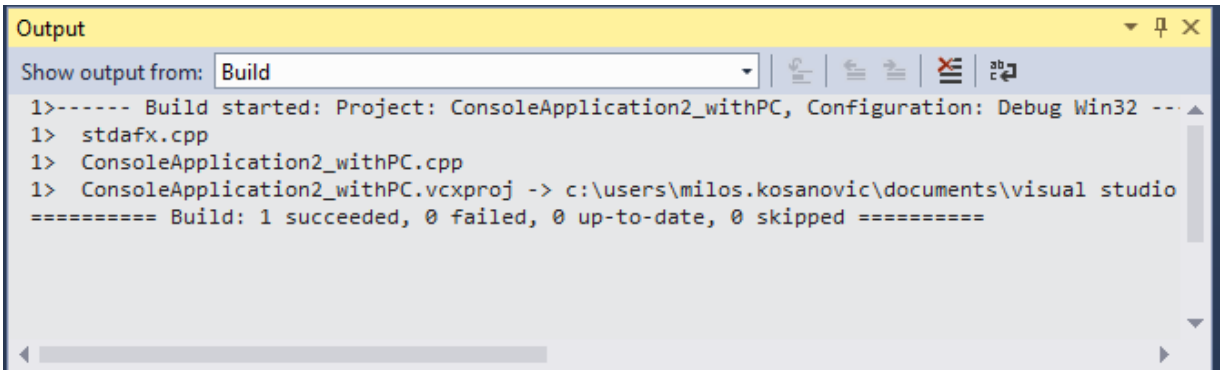
Kompajliranje programa i izveštaj o greškama

Nakon unosa koda programa u odgovarajuće datoteke moguće je izvršiti kompajliranje, povezivanje i izvršavanje programa. U ovu svrhu se koriste stavke iz menija *Build* ili iz odgovarajuće prečice sa alatima (vidi sliku 7).



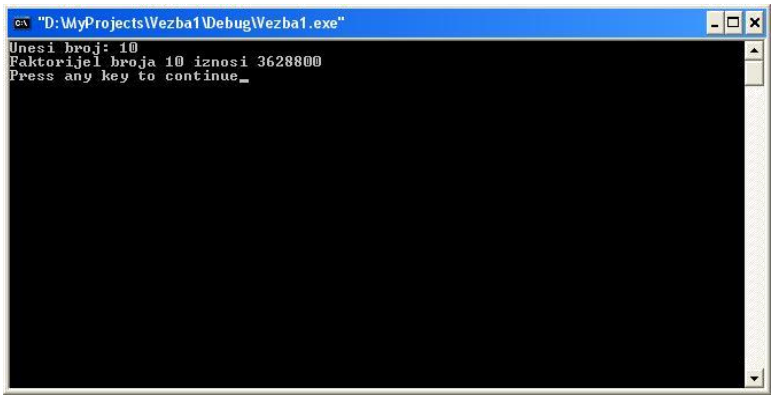
Slika 7. Alati za kompajliranje programa

Najjednostavniji način za iniciranje procesa kompiliranja svih datoteka sa izvornim kodom, kreiranje izvršne (EXE) datoteke programa i njeno startovanje je korišćenjem stavke *Build/Start Without Debug*, kombinacije tastera ***Ctrl+F5*** sa tastature ili odgovarajuće prečice (ikonica "▶"). Ukoliko program nije prethodno preveden ili je izmenjen od vremena kada je poslednji put preveden, pojaviće se prozor sa pitanjem da li treba izvršiti prevođenje (kompajliranje) programa. U oba slučaja potrebno je kliknuti na dugme Yes nakon čega će uslediti kompajliranje i povezivanje (linkovanje) izvršnog programa. Ukoliko je program uspešno preveden automatski će se izvršiti njegovo startovanje, u suprotnom u donjem delu osnovnog prozora će biti prikazan izveštaj o otkrivenim greškama (slika 8). Za svaku otkrivenu grešku ili upozorenje pored koda i opisa, navedena je datoteka i linija koda u kojoj je greška nađena. na liniju sa opisom greške u ovom izveštaju automatski se otvara odgovarajuća datoteka i vrši postavljanje kursora na problematičnu liniju.



Slika 8. Prikaz izveštaja o greškama i upozorenjima

Nakon ispravljanja grešaka potrebno je ponovo izvršiti startovanje procesa prevođenja i izvršenja programa. Na slici 9 prikazan je prozor u kome se izvršava program. Nakon završetka izvršenja programa biće ispisan tekst "Press any key to continue". Pritiskom na neki taster prozor u kome se program izvršavao će biti zatvoren. 6565



Slika 9. Izgled prozora u kome se izvršava program

Prekompajlirani hederi

Prekompajlirani hederi predstavljaju C i C++ heder fajlove koji se pre kompajliranja pripreme i prebace u *intermediate* formu koju kompajler može brže da kompajlira. Ova forma se naziva još i prekompajlirani heder. Korišćenje ovakvih hedera može skratiti vreme kompajliranja, naročito kada se primeni na velike heder fajlove. Problem kompajliranja velikog broja uključenih fajlova postoji i kod drugih programskih jezika i svaki od njih ga rešava na neki svoj način. Demonstrirajmo primerom kako se to rešava u C++ jeziku. Uzmimo za primer projekat koji ima dodata dva fajla: heder fajl header.h i fajl sa izvornim kodom source.cpp:

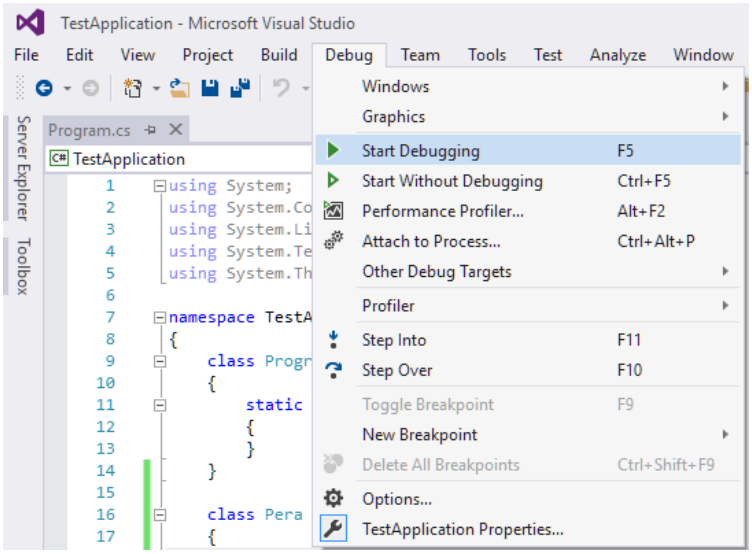
```
//header.hpp
...
//source.cpp
#include "header.h"
...
```

U slučaju da je opcija za prekompajlirane hedere uključena, kompajler će generisati prekompajlirani heder fajl *header.pch*. Kada sledeći put budete kompajlirali fajlove, ukoliko se vreme zadnje promene fajla header.h nije promenilo, računar će preskočiti kompajliranje i iskoristiti već postojeći header.pch fajl. U suprotnom novi fajl heaader.pch će biti kreiran, a stari obrisan.

Prilikom korišćenja Visual Studia 2015 postoji fajl stdafx.h koji predstavlja prekompajlirani heder za sve one biblioteke koje želite da uključite u vaš program, a koje se retko menjaju. To znači da sve *include* direktive treba pisati u ovom fajlu.

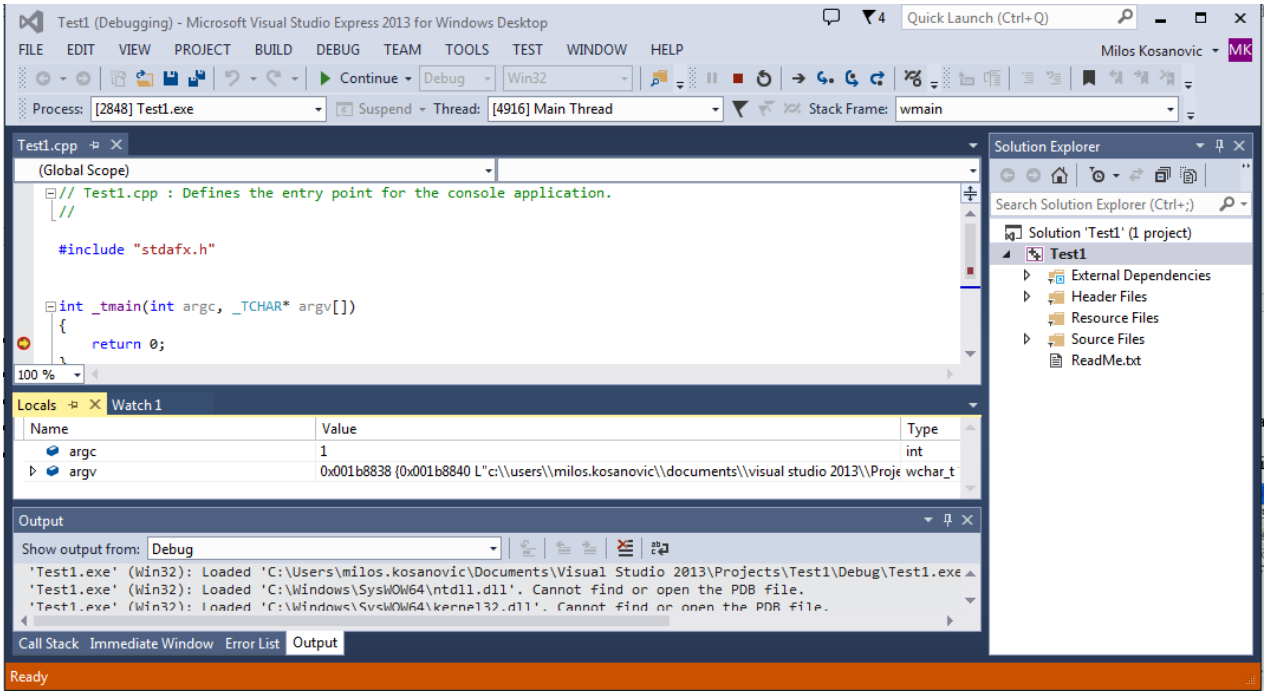
Debagiranje programa

Pored sintaksnih grešaka koje se otkrivaju u toku prevođenja, program može da sadrži i logičke greške koje sprečavaju da se izvršava na način koji je programer predvideo. Ovakve greške se nazivaju bug-ovi, a proces njihovog uklanjanja debugging. Proces debagiranja podrazumeva postavljanje prekidnih tačaka (*breakpoints*) u kojima će se izvršenje programa privremeno prekinuti u cilju očitavanja vrednosti pojedinih promenljivih i sl. Postavljanje (ili uklanjanje) prekidnih tačaka vrši se postavljanjem kursora na odgovarajuću liniju i klikom na dugme u obliku šake (vidi sliku 11). Druga varijanta za dodavanje (ili uklanjanje) prekidne tačke je desni klik na odgovarajuću programsku liniju i izbor stavke Insert/Remove Brakepoint iz padajućeg menija. Za startovanje izvršenja programa u ovom modu koristi se stavka iz menija Build/Start Debug/Go, taster F5 ili odgovarajuća prečica iz linije sa alatima (vidi sliku 10).



Slika 10. Alati za debagiranje programa

Pri debagiranju, program će se izvršavati normalno sve dok se ne dođe do neke od zadatih prekidnih tačaka. U tom trenutku se zaustavlja izvršenje i postaje aktivno okruženje *Visual C++*-a prekonfigurisano za debagiranje. Na slici 11 prikazan je izgled okruženja u modu za debagiranje.



Slika 11. Okruženje za debagiranje programa

Za kontrolu izvršenja koriste se komande menija *Debug*. Značenja pojedinih komandi su sledeća:

- *Go* (taster *F5*) – nastavlja izvršenje do naredne prekidne tačke.
- *Restart* (kombinacija tastera *Ctrl+Shift+F5*) – resetuje izvršenje programa u modu za debagiranje.
- *Stop Debugging* (kombinacija tastera *Shift+F5*) – prekida debagiranje programa.
- *Step Into* (taster F11) – ulazi u funkciju koja čeka na izvršenje. Ukoliko je u pitanju neka druga naredba izvršava je i prelazi na narednu liniju u kodu.
- *Step Over* (taster F10) – izvršava funkciju ili naredbu i prelazi na narednu liniju u kodu.
- *Step Out* (kombinacija tastera Shift+F11) – izvršava tekuću funkciju do kraja i izlazi na nivo iz koga je pozvana.
- Run to Cursor (kombinacija tastera Ctrl+F10) – izvršava program do linije koda na koju ukazuje položaj kursora.

Po zaustavljanju izvršenja moguće je očitati vrednosti pojedinih promenljivih i prikaz konteksta (funkcija) iz kojeg se došlo do prekidne tačke. Očitavanje vrednosti promenljivih se može izvršiti na nekoliko načina:

1. Zadržati pokazivač miša iznad naziva promenljive negde u kodu, što će rezultovati ispisivanjem vrednosti pored pokazivača.
2. Dodavanjem naziva promenljive čije se praćenje želi u panelu *Watch* (desno ispod koda).
3. Direktno, ukoliko je promenljiva automatski izlistana u spisku promenljivih koje se trenutno koriste (panel levo ispod koda).

Samostalni rad studenta

Zadatak 1: Napisati i pokrenuti program.

```
void main() {
      printf("Moj prvi program u C jeziku\n");
}
```

Zadatak 2: Napisati i pokrenuti program.

```
void main()
{
      int razlika= 100 -90;
      printf("Razlika 100 -90= %d\ n", razlika);
}
```

Gde je definisana i šta radi naredba printf?

Zadatak 3: Napisati i izvršiti program. Probajte da debagirate program. Kolika je vrednost promenljive i nakon pre ulaska u petlju (linija 5) i nakon izlaska iz petlje (linija 10)?

```
void main()
{
      int zbir = 0, i;
      for (i=0; i<10; i++)
      {
            zbir = zbir + i;
            printf("i=%d  zbir=%d\n", i, zbir);
      }
      printf("\nzbir=%d\n", zbir);
}
```

Odgovor:

Zadatak 4: Napisati sledeći program, pokrenuti ga i zatim odgovoriti na pitanja.

```
void main(){
      int a = 3;
      int b = 2;
      int c = 3 / 2;
      printf("%d \n", c);
}
```

Zašto je rezultat deljenja 1?

Zadatak 5: Napisati sledeći program, pokrenuti ga i zatim odgovoriti na pitanja.

```
void main(){
	int a = 3;
	int b = 0;
	a = a / b;
	printf("%d \n", a);
}
```

Zašto program ne radi? Objasnite šta se desilo?

Zadatak 6: Nacrtati algoritam za sledeće zadatke:

A. Sastaviti dijagram toka i napisati program kojim se izračunava vrednost funkcije:

$$y = \begin{cases} x, & x < 2 \\ 2, & 2 \le x < 3 \\ x - 1, & x \ge 3 \end{cases}$$

B. Nacrtati algoritam koji štampa vaše ime 1000 puta.
C. Nacrtati algoritam koji računa N5
D. Nacrtati algoritam koji računa N^x

Pitanja za odbranu vežbe

1) Šta je Visual studio 2015?
2) Kako se kreira novi projekat? Koje vrste projekta postoje?
3) Šta je konzola?
4) Kako se dodaje nova biblioteka u projekat?
5) Kako se dodaje novi fajl u projekat?
6) Šta je *ClassView?*
7) Šta je Solution Explorer? File view?
8) Koji sve fajlovi postoje ili se kreiraju u toku bildovanja projekta?
9) Šta je kompajliranje programa?
10) Šta je linkovanje programa?
11) Šta je debagiranje programa?
12) Kako se dodaje breakpoint?
13) Šta su to prekompajlirani hederi?
14) Čemu služi fajl stdafx.h?
15) Čemu služi fajl stdio.h?
16) Nakon zaustavljanja programa u debug modu, na koji način se sve može očitati vrednost neke promenljive?
17) U čemu je razlika između naredbi Step Into i Step Over u debug modu?

VEŽBA 2 – Tipovi Podataka, naredba scanf, printf, operatori

Elementi programskog jezika - tokeni

Objasnimo osnovne pojmove bilo kog programskog jezika:

- **Identifikator**- je bilo koji niz slova, cifara i znaka _ koji ne počinje cifrom i ne pripada skupu ključnih reči. Identifikator se koristi za imenovanje promenljivih, funkcija, konstanti, složenih tipova podataka i labela.
- **Ključna reč** – skup reči koje su rezervisane i imaju specijalno značenje u programskom jeziku C. To su reči:

Tabela 1 - Ključne reči u programskom jeziku C

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

- **Separator** – odvaja delove koda,
  - {, } – početak i kraj bloka koda
  - (,) – početak i kraj liste parametara
  - ; – kraj naredbe
  - // – linijski komentar
  - /*, */ – komentar bloka naredbi
- **Literal** – konstantna vrednost, u slučaju tekstualno literala ona predstavlja niza karaktera koja se piše između navodnika. Primer “primer”. Slovni literal se piše sa jednim navodnikom. Brojni literali se pišu bez posebnih znakova.
- **Operator** (Operator) – operacija koja se obavlja sa 1 do 3 promenljive. Svaki operator ima određeni prioritet u odnosu na neki drugi operator

Dodatni pojmovi

- **Promenljiva** (*Variable*) – ime ili referenca na zapamćenu vrednost (obično u memoriji)
- **Tip podatka** (*Data type*) – određuje veličinu promenljive u memoriji, koje vrednosti ona može da uzme, koje operacije su dozvoljene
- **Izraz** (*Expression*) – kombinacija literalnih vrednost, promenljivih, operatora i funkcija
- **Opseg važenja** (*Scope*) – deo programa u kome je ime nekog entiteta (promenljive ili funkcije) validno i može se iskoristiti za pristup tom entitetu
- **Deklaracija (definicija)** promenljive – određivanje tipa promenljive prilikom čega se rezerviše memorijski prostor za taj tip promenljive. Primer: int broj (rezerviše 4B), char p (rezerviše 1B)
- **Inicijalizacija** promenljive – Dodeliti promenljivoj inicijalnu vrednost

Tipovi podataka

Tipovi podataka se dele na osnovne tipove podataka (built-in type) i kreirane tipove podataka (custom type). Svaki tip podataka definiše:

- Količinu memorije koju će promenljiva zauzeti u radnoj memoriji
- Maksimalnu i minimalnu vrednost koju promenljiva može da ima
- Tip operacija koje su dozvoljene sa ovim tipom promenljive
- Kod složenih tipova podataka definiše i članove i metode klase koju taj tip promenljive sadrži, kao i klase koje nasleđuje
- Mesto u memoriji gde će promenljiva biti zapamćena u trenutku izvršavanja (*run time)*

Tabela 2 – Karakteristike osnovnih tipova podataka za mašinu sa 16-bitnim procesorom

Tip promenljive	Ključna reč	Broj bajtova	Opseg
Character	char	1	-128 do 127
Integer Number	int	2	-32768 do 32767
Real Number	float	4	3.4E-38 do 3.4E38
Real Number	double	8	-1.7E308 do 1.7E308
Odsustvo tipa podataka	void		
Unsigned character	unsigned char	1	0 do 255
Short Integer	short int	2	-32768 do 32767
Long integer	long int	4	-2147483648 do 21474368647
Unsigned integer	unsigned int	2	0 do 65535
Long double	long double	10	3.4E-4932 do 1.1E4932

Tačan skup vrednosti koje može uzeti neka promenljiva zavisi od tipa procesora na kome se program izvršava. Većina modernih procesora koristi 4 bajta (odnosno ima 32-bitni procesor). Informacije o tipovima podataka koristi kompajler kako bi proverio da li su sve operacije koje se obavljaju u vašem kodu sigurne (*type safe*). Na primer, ako je deklarisana promenljiva tipa int, kompajler dozvoljava da se ona koristi u operacijama sabiranja i oduzimanja. Ukoliko iste ove operacija probamo da izvršimo sa tipom promenljive bool, kompajler će generisati grešku kao u sledećem primeru.

```
int a = 5;
int b = a + 2; //OK
bool test = true;
// Error. Operator '+' cannot be applied to operands of type 'int' and
'bool'.
int c = a + test;
```

Printf naredba

Naredba printf ima sledeći oblik ***printf("format", promenljiva)*** gde je:

- promenljiva – ime promenljive a
- Format – string koji definiše na koji način će ta promenljiva biti odštampana na standardni izlaz. Primer formata se sastoji iz nekoliko delova (flag) i njegov opšti oblik je: **%[flags][width][.precision][length]specifier** i gde je svaki od delova objašnjen u tabeli ispod.

Tabela 3 - Objašnjenje formata naredbe printf

Flags	Description
flags	Može imati neku od vrednosti gde svaka od njih označava način na koji se ispisuje rezultat -, +, #, 0
width	Minimalni broj karaktera koje se štampa. Ako je vrednost manja od njega dodaju se blanko znaci.
.precision	Zavisi od tipa specifikatora. Kod realnih brojeva %f označava broj cifara nakon decimalne tačke.
length	H, l ili L
specifier	Konverzioni znak c, f, s, d

Konverzioni znaci za različite tipove podataka

Konverzioni znaci se koriste prilikom korišćenja naredbi pisanja ili čitanja iz memorije. U memoriji računara svaki podatak je zapamćen kao niz bitova. Ovi konverzioni znaci nam pomažu da određenu grupu bitova prepoznamo i pretvorimo u odgovarajući celi broj, realni broj ili karakter.

Tabela 4 - Konverzioni znaci za različite tipove podataka

Konverzacioni znak ili specifikator	Opis
%c	Jedan karakter znakovnog tipa
%d	Označeni ceo broj
%e	Broj u pokretnom zarezu, e -notacija
%E	Broj u pokretnom zarezu, E -notacija
%f	Broj u pokretnom zarezu, decimalna not.
%i	Označeni ceo broj
%X	Neoznačeni heksadecimalni broj (A do F)
%x	Neoznačeni heksadecimalni broj (a do f)
%s	Karakterni niz

Escape karakteri

Escape karakteri se koriste za definisanje nekih specijalnih znakova koje inače nije moguće napisati u string konstantama. Ovo je lista specijalnih znakova:

Tabela 5 - Lista specijalnih znakova koji se koriste za formatiranje ispisa rezultata

Escape sequence	Description	Representation
\'	single quote	byte 0x27
\"	double quote	byte 0x22
\?	question mark	byte 0x3f
\\	backslash	byte 0x5c
\0	null character	byte 0x00
\a	audible bell	byte 0x07
\b	backspace	byte 0x08
\f	form feed - new page	byte 0x0c
\n	line feed - new line	byte 0x0a
\r	carriage return	byte 0x0d
\t	horizontal tab	byte 0x09
\v	vertical tab	byte 0x0b

Operatori

U programskom jeziku C direktno je moguća primena osnovnih matematičkih operatora +, -, *, / i operatora koji ne postoje u drugim programskim jezicima: inkrement i dekrement. Ostale matematičke operatore možemo uključiti iz skupa datoteka zaglavlja koji sadrži datoteku <math.h> i koja se po potrebi u procesu strukturnog programiranja uključuje u izvorni program naredbom #include .

Prioritet Operatora

U sledećoj tabeli možete naći operatore u C jeziku od najvećeg ka najmanjem prioritetu:

Tabela 6 - Tablica operatora poređana po prioritetu operatora

Level	Precedence group	Operator	Description	Grouping
1	Scope	::	scope qualifier	Left-to-right
2	Postfix (unary)	++ --	postfix increment / decrement	Left-to-right
		()	functional forms	
		[]	subscript	
		. ->	member access	
3	Prefix (unary)	++ --	prefix increment / decrement	Right-to-left
		~ !	bitwise NOT / logical NOT	
		+ -	unary prefix	
		& *	reference / dereference	
		new delete	allocation / deallocation	
		sizeof	parameter pack	
		(*type*)	C-style type-casting	
4	Pointer-to-member	.* ->*	access pointer	Left-to-right
5	Arithmetic: scaling	* / %	multiply, divide, modulo	Left-to-right
6	Arithmetic: addition	+ -	addition, subtraction	Left-to-right
7	Bitwise shift	<< >>	shift left, shift right	Left-to-right
8	Relational	< > <= >=	comparison operators	Left-to-right
9	Equality	== !=	equality / inequality	Left-to-right
10	And	&	bitwise AND	Left-to-right
11	Exclusive or	^	bitwise XOR	Left-to-right
12	Inclusive or	|	bitwise OR	Left-to-right
13	Conjunction	&&	logical AND	Left-to-right
14	Disjunction	||	logical OR	Left-to-right
15	Assignment-level expressions	= *= /= %= += -= >>= <<= &= ^= |=	assignment / compound assignment	Right-to-left
		?:	conditional operator	
16	Sequencing	,	comma separator	Left-to-right

Samostalni rad studenta

Zadatak 1. Testirati program na programskom jeziku C za upotrebu specifikatora konverzije u odnosu na očekivane vrednosti celobrojnih promenljivih.

```
#include <stdio.h>
void main()
{
      unsigned neoznacen = -39000;
      printf ("neoznacen = %u, i nije %d\n", neoznacen,neoznacen);
      printf ("Characters: %c %c \n", 'a', 65);
      printf ("Decimals: %d %ld\n", 1977, 650000L);
      printf ("Preceding with blanks: %10d \n", 1977);
      printf ("Preceding with zeros: %010d \n", 1977);
      printf ("Some different: %d %x %o %#x %#o \n", 100, 100, 100, 100,
100);
      printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
      printf ("Width trick: %*d \n", 5, 10);
      printf ("%s \n", "A string");
}
```

Napisati izlaz iz programa i prokomentarisati rezultat tamo gde je potrebno:

Zadatak 2. Sastaviti program na programskom jeziku C za formatiranje ulaza u pokretnom zarezu.

```
void main()
{
      /* deklaracija podataka */
      float f_pro;
      double d_pro;
      /*dodela vrednosti*/
      f_pro = 106,11;
      d_pro = -0,0000654;
      /*štampanje vrednosti promenljivih */
      printf ("Promenljiva f_pro=%2f\n", f_pro);
      printf ("Promenljiva d_pro=%.11f\n", d_pro);
      printf ("Promenljiva f_pro=%e\n", f_pro);
      printf ("Promenljiva d_pro=%G\n", d_pro) ;
}
```

Izlaz iz programa je:

Zadatak 3. Sastaviti program na programskom jeziku C za unos proizvoljnog karaktera i za štampanje ASCII koda tog karaktera. Unesite i zapišite ASCII kod svih slova u vašem imenu.

```
#include<stdio.h>
void main()
{
      char ch;
      printf("Unesite proizvoljan karakter.\n");
      scanf("%c",&ch); /*naredba za unos podataka*/
      printf("ASCII kod unetog karaktera %c je %d\n",ch,ch);
}
```

Prepišite izlaz iz programa:

Zadatak 4. Napisati program na C jeziku za štampanje dva cela broja x i y u kao i njihovog inkrementa i dekrementa koristeći prefiksni i postfiksni operator.
Napomena: Operator inkrementa se koristi da bi se povećala vrednost promenljive za jedan. Operator dekrementa se koristi da bi se smanjila vrednost promenljive za jedan. Pravila po kojima se primenjuje postfiksni i prefiksni zapis operatora dekrementa na celobrojne promenljive i izraze su potpuno analogna pravilima upotrebe operatora inkrementa i mogu se sagledati iz navedenog primera.

```
void main()
{
      int x,y;
      x=10;       y=10;
      printf("Vrednost izraza ++x je   %d\n",++x);
      printf("Vrednost izraza y++ je   %d\n",y++);
      printf("Nakon inkrementiranja vrednost za x je %d\n",x);
      printf("Nakon inkrementiranja vrednost za y je %d\n",y);
      printf("Vrednost izraza --x je   %d\n",--x);
printf("Vrednost izraza y-- je   %d\n",y--);
      printf("Nakon dekrementiranja vrednost za x je %d\n",x);
      printf("Nakon dekrementiranja vrednost za y je %d\n",y);
}
```

Upisati rezultate rada programa i objasniti razliku između prefiksnog i postfiksnog operatora:

Zadatak 5. Šta označavaju vitičaste zagrade u jeziku C? Zašto koristimo ove zagrade pri definiciji main (ili bilo koje druge) funkcije

Zadarak 6. Opišite razliku između 7,”7”,’7’?

Zadatak 7. Napišite vrednosti sledećih izraza na osnovu tablice prioriteta operatora:

1. ((5 == 5) && (3 > 6))
2. ((5 == 5) || (3 > 6))
3. a=2; b=7; c = (a>b) ? a : b; Koliko je c?
4. a=2, b=7, a += 3 + b; Koliko je a i b?
5. a=2, b=7, a += 3 + b++, b=1; Koliko je a i b?
6. a=2; a>>2 || a=+3 | a

```

```

Napomena: Ukoliko u toku izrade zadataka dobijete sledeću grešku:

```
error C4996: 'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS.
```

Neophodno je u header fajl stdafx.h definisati sledeću pretprocesorsku konstantu pre nego što dodate bilo koju biblioteku, odnosno odmah nakon naredbe #pragma once:

#define _CRT_SECURE_NO_WARNINGS

Pitanja za odbranu vežbe

1. Šta je promenljiva? Primer.
2. Šta je tip podatka? Primer
3. Šta je opseg važenja promenljive? Primer
4. Šta je operator? Primer
5. Šta je izraz? Primer
6. Koji su osnovni tipovi podataka? Koliko bajtova i koje vrednosti može da uzme bilo koji od njih?
7. Šta su konverzioni znaci? Primer
8. Šta su escape karakteri? Primer
9. Šta su aritmetički operatori? Primer
10. Šta su logički operatori? Primer
11. Šta su unarni operatori? Primer
12. Šta su binarni operatori? Primer
13. Šta je operator dodele? Primer
14. Čemu služi sizeof operator? Primer
15. Šta je to prioritet operatora? Primer
16. Šta su to prefiksni operatori? Primer
17. Šta su to postfiksni operatori? Primer
18. Šta su to operatori nad bitovima? Primer
19. Šta je operator jednakosti? Primer
20. Šta su to konstantne? Navedi primer za celobrojnu i karakternu konstantu.

VEŽBA 3 –Promenljive, If naredba, pretprocesorske direktive, scanf naredba

Pretprocesorske direktive

Pred-procesorske direktive predstavljaju linije u kodu koje imaju predznak (#) koji se još naziva i heš (hash). Ove linije nisu deo programa već direktive namenjene procesoru. Pred-procesor proverava kod pre nego što kompajliranje koda uopšte počne i izvršava sve direktive. Tek nakon toga se vrši kompajliranje programa.

Direktive se mogu naći samo u jednoj liniji koda. Čim pred-procesor naiđe na karakter za novi red on to označava kao kraj te direktive pa se na kraju direktive ne stavlja znak (;). Primer naredbe je:

#define TABLE_SIZE 100
#include <stdio.h>

Ove direktive se najčešće koriste za definisanje konstanti ili promenu načina kompajliranja koda, u zavisnosti od hardverske ili softverske platforme za koju se program piše. Direktiva #include <stdio.h> u zaglavlju programa uključuje stdio.h biblioteku koja je deo paketa C kompajlera i sadrži informacije o ulazno/izlaznim funkcijama za računar. Naziv potiče od STanDard Imput/Output.

Drugo mesto gde se koriste predprocesorske direktive je pisanje makro funckija ili skraćeno **makroa.** Prednost, a u jedno i mana makroa je što su oni neutralni u odnosu na tip odataka koji im se prosledi. Ono što predprocesor radi je da traži odgovarajuće makroe u kodu i onda direktno kopira ili vrši zamenu naziva makroa sa njegovim kodom. Primer makroa:

```
      #define    squared(x)  x*x
```

Primer poziva makroa u kodu:

```
      int x = squared(3);   // rezultat je x= 3 * 3 = 9
      int x = squared(2+1); // rezultat je x = 2 + 1 * 2 + 1 = 5
```

Naredbe za ulaz - Scanf

```
int scanf(<format>, <lista promenljivih>)
Scanf („%d", x);
Scanf(„%d%f%lf", x, y, z);
```

Povratna vrednost je negativan broj ako je funkcije neuspešna, u suprotnom broj promenljivih koji je uspešno upisan. Format je literal koji sadrži konverzione znake opisane u tabeli 4. Svakoj promenljivoj iz liste odgovara jedna pojedinačna ulazna konverzija. Konverzioni znaci se koriste prilikom korišćenja naredbi pisanja ili čitanja iz memorije. U memoriji računara svaki podatak je zapamćen kao niz bitova. Ovi konverzioni znaci nam pomažu da određenu grupu bitova prepoznamo i pretvorimo u odgovarajući celi broj, realni broj ili karakter.

Naredba if then else

```
if (uslov)
      naredba
else
      naredba
```

```
if (uslov)
      naredba;
else if (uslov)
      naredba;
else
      naredba;
```

Ukoliko struktura sadrži više naredbi neophodno je pisati oznaku za blok naredbi {}.

Matematičke funkcije iz biblioteke math.h

- Double pow(double x, double a)
- double sin(double x)
- double cos(double x)
- double sqrt(double x)
- double floor(double x)
- double log(double x)

Primer korišćenja matematičkih funkcija:

```
#include <math.h>
void main(void) {
      int n = 5;                  double x = 32.0, r1, r2, r3;
      r1 = pow(x, 1./n);          r2 = pow(x, 2.);
      r3 = sqrt(x);
      printf("%d_ti koren broja %lf je: %lf\n",n, x,rezultat);
}
```

Opseg važenja promenljivih

Opseg važenja ili ***scope*** u bilo kom programskom jeziku predstavlja deo koda u kome promenljiva postoji ili važi. Van tog dela koda promenljivoj se ne može pristupiti i kažemo da ona van tog dela koda ne važi. Postoje tri mesta gde se promenljiva može deklarisati u programskom jeziku C.

1. U okviru neke funkcije i tada se naiva **lokalna promenljiva**
2. Van svake funkcije i tada se naziva **globalna promenljiva**
3. U deklaraciji funkcije kao jedan od parametara i tada se naziva **formalni parametar** funkcije

Samostalni rad studenta

Zadatak 1 Napišite logički izraz kojim možete da proverite da li je uneti karakter c:

- Malo slovo (Answer: c>='a'&& c<='z')
- Veliko slovo (Answer: c>='A'&& c<='Z')
- cifra (Answer: c>='0'&& c<='9')
- beli znak (uključujući blanko, tab i novi red) (Answer: c=='\n'|| c=='\t'|| c=='')

Zadatak 2 Sastaviti program na programskom jeziku C za upotrebu funkcije rand() kojom se uključuje generator slučajnih brojeva i štampa celobrojni slučajni broj i njegova dvostruka vrednost. Štampati i slučajan borj između 0 i 9. Štmpati slučajan broj između 10 i 19.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void main() {
      int slucajan, dvostruki, x, y;
      srand (time(NULL));        /* initialize random seed: */

      slucajan = rand();
      dvostruki = slucajan*2;
      x = rand() % 10;      /*slucajan izmedju 0 i 9*/
      y = rand() % 10 + 10;      /*slucajan izmedju 10 i 19*/
      printf("Slucajan broj je %d\n", slucajan);
      printf("Dvostruki slucajan broj je %d\n", dvostruki);
      printf("Jos neki slucajni brojevi %d, \t %d \n", x, y);
}
```

Prepišite izlaz iz programa. Funkcija rand() je deo koje biblioteke?

Zadatak 3 Proveriti i testirati strukturu *if* naredbe:

```
void main() {
      int x, y, z;     x = 10;          y= 3;           z = ( x / y ) * y;
      if (x ==  z)
            printf ( "Vrednosti x i z su jednake\n");
      if ( x  <=  z)
            printf ("Vrednost x je < ili = od vrednosti z \n");
      if ( x  >= z)
            printf ( "Vrednost x je > ili = od vrednosti z \n" );

      printf ( "Vrednost x je %d\n", x );
      printf ( "Vrednost z je %d\n", z );
}
```

Prepisati rezultat:

Zadatak 4 Napisati program na C jeziku za rešavanje kvadratne jednačine $x^2 + x - 2 = 0$ korišćenjem obrasca:

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

i štampanje vrednosti za x1 i x2.

```
#include<math.h>
void main()
{
      double a,b,c;
      double x1,x2,pom;
      a = 1;      b = 1;      c = -2;
      pom = sqrt(b*b-4*a*c);
      x1 = ( (-b) + pom ) / 2 * a;
      x2 = ( (-b) - pom ) / 2 * a;
      printf("Vrednost prvog korena x1 = %f\n",x1);
      printf("Vrednost drugog korena x2 = %f\n",x2);
}
```

Izmeniti program tako da se vrši provera da li su rešenja kvadratne jednačine kompleksna. Napisati navedene izmene kao i rezultate rada programa u prvom i drugom slučaju:

Zadatak 5 Napisati isti program ali u opštem obliku kada se veličine a, b i c učitavaju sa tastature i u kome se korisnik obaveštava ukoliko su rešenja jednačine kompleksna.

Zadatak 6 Napisati program na C jeziku za izračunavanje ukamaćene vrednosti ako je poznata mesečna kamatna stopa, period oročavanja u mesecima i iznos glavnice. Štampati dobijene rezultate i proveriti matematičku postavku programa. **Napomena:** Koristiti pow funkciju iz biblioteke math.h koja izračunava x na y. Kamata se računa kao glavnica * (1+kamata/100)^{period}.

```
void main( )
{
      double stopa, period, glavnica;
      printf( "Unesite mesecnu kamatnu stopu: " ) ;
      scanf( "%lf", &stopa ) ; /* ulaz u pokretnom zarezu */
      /* konverzija u procente */
      stopa = stopa / 100.0;
      printf( "Unesite glavnicu: " );
      scanf( "%lf", &glavnica );
      printf( "Unesite vreme orocavanja u mesecima: " );
      scanf( "%lf", &period ) ;
      printf( "Ukamacena vrednost je = %.2f\n",
            glavnica * pow( (1.0+stopa), period));
}
```

Proveriti izlaz iz programa za sledeće vrednosti: Kamatna stopa: 1, glvnica: 100, oročavnje: 3 meseca. Koliko će iznositi krajnja vrednost?

Zadatak 7 Napisati program kojim se izračunava vrednost funkcije:

$$y = \begin{cases} x, & x < 2 \\ 2, & 2 \le x < 3 \\ x - 1, & x \ge 3 \end{cases}$$

Zadatak 8 Objasnite zašto su ovi izrazi netačni:

a) `#include <stdio.h>;`

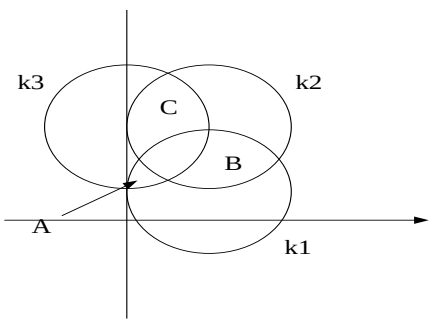
b) `void main(int arg1){ return arg1-1 }`

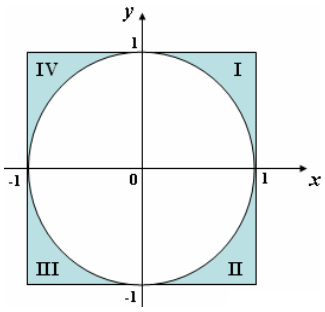
c) `int 2nd_value=10;`

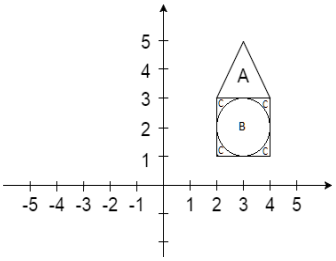
```
d) #define MESSAGE = "Happy new year!"
   printf("%s",MESSAGE);
```

Zadatak 9 Uradite broj zadatka koji dobijate kada vaš broj indeksa podelite po modulu 5. Radite samo jedan od sledećih zadataka. Zadatak nije neophodno prepisati u izveštaj.

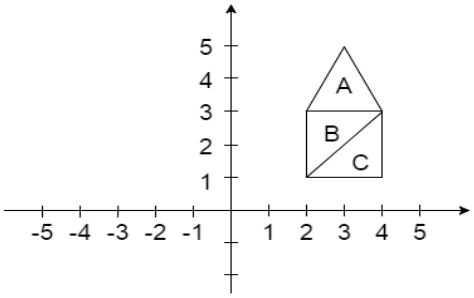
Napisati program koji za unetu tačku sa koordinatama (x, y) proverava da li se ona nalazi u oblastima označenim na slici. Na izlazu prikazati oblast kojoj tačka pripada. Ukoliko ne pripada ni jednoj oblasti, ispisati poruku „tačka ne pripada ni jednoj oblasti“. Tačke na pravama i krugovima mogu pripadati bilo kojoj od oblasti. Jednačina prave kroz dve tačke data je sledećom jednačinom: $y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$. Jednačina kruga je data sledećom jednačinom: $(x - x_1)^2 + (y - y_1)^2 = R^2$.



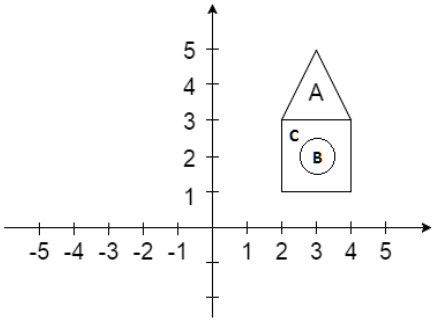




Slika 3



Slika 5



Slika 4

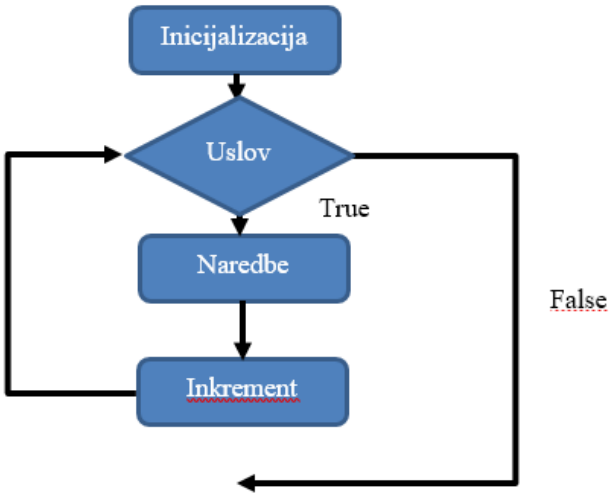
Pitanja za odbranu vežbe

1. Šta su pred-procesorkse direktive?
2. Koja je razlika između pred-procesorske direktive i programske naredbe?
3. Čemu služi pred-procesorska direktiva include?
4. Čemu služi pred-procesorska direktiva define?
5. Čemu služi programska naredba scanf? Primer.
6. Koji je format naredbe if? Primer.
7. Šta označava ključna reč void?
8. Koji nazivi promenljivih nisu dozvoljeni u programskom jeziku c?
9. Šta su to ključne reči u bilo kom programskom jeziku?
10. Šta su to logički izrazi? Primer.
11. Šta su globalne promenljive?
12. Šta su lokalne promenljive?
13. Kako se u C programskom jeziku pišu komentari?
14. Šta je to oblast važenja ili scope?

VEŽBA 4 – for petlja, while petlja

For petlja

Upravljačke informacije kod for petlje, za razliku od while petlje, su smeštene na jednom mestu i to na vrhu iteracije. Petlja for je definisana, što znači da unapred znamo koliko puta će petlja da se izvrši, a njena osnovna konstrukcija je:



- **Izraz 1 Inicijalizacija** – inicijalizacije vrednosti brojača. i = 0
- **Izraz 2 Uslov** – na primer upoređenje brojača sa graničnom vrednošću. i < 10
- **Izraz 3 Inkrementirnje** – ažuriranje brojača pri svakoj iteraciji. i ++

Prema tome, opšti oblik naredbe for je:

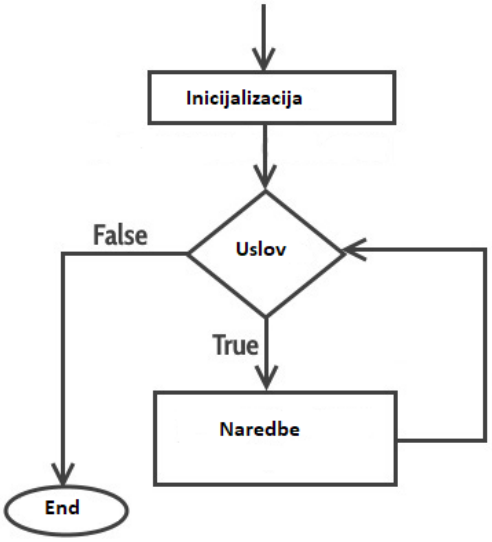
```
for ( izraz1, izraz2, izraz3 )
          naredba
```

Neki od izraza u for petlji može ostati i prazan i tada se za taj izraz smatra da ima vrednost true.

While petlja

Upravljačku strukturu while karakterišu četiri osobine:

- petlja se ponavlja sve dok izraz koji se testira ne postane netačan (logička 0);
- petlja je sa *ulaznim uslovom* : odluka o još jednom prolasku kroz telo petlje se donosi pre izvršenja naredbi petlje;
- while petlja mora sadržati promenu vrednosti izraza tako da on postane tačan posle određenog broja iteracija, kako petlja ne bi postala beskonačna;
- jedna naredba se posmatra kao deo upravljačke strukture while petlje, bilo ona prosta ili složena.



Sintaksa while i do while petlje je:

```
While (uslov) {
      Naredba
}
```

```
do{
      naredba
}
While(uslov)
```

Samostalni rad studenta

Zadatak 1: Napisati program na C jeziku za prikaz neparnih brojeva manjih ili jednakih 20 korišćenjem for petlje.

Zadatak 2. Napisati program na C jeziku za izračunavanje zbira celih brojeva upotrebom while petlje. Izračunavanje se prekida kada se unese '0';

```
void main()
{
      long sum = 0L;
      int num;
      printf("Unesite broj za sumiranje Ili 0 za izlaz\n");
      scanf("%ld", &num);
      while (num != 0)
      {
            sum = sum+num;
            printf("Unesite naredni broj za sabiranj ili 0 za quite ::::::::
%d \n ", num);
            scanf("%ld", &num);
      }
      printf("Zbir unetih brojeva je %ld.\n", sum);
}
```

Zadatak 3. Napisati program u C jeziku za permutovanje cifara celog broja korišćenjem while naredba: (npr. 54321 u 12345)

Postupak:
celi broj 54321 delimo po modulu 10 - rezultat je 1
celi broj 54321 delimo sa 10 - rezultat je 5432
celi broj 5432 delimo po modulu 10 - rezultat je 2
celi broj 5432 delimo sa 10 - rezultat je 543
celi broj 543 delimo po modulu 10 - rezultat je 3
celi broj 543 delimo sa 10 - rezultat je 54
celi broj 54 delimo po modulu 10 - rezultat je 4
celi broj 54 delimo sa 10 - rezultat je 5
celi broj 5 delimo po modulu 10 - rezultat je 5
Petlja se prekida jer broj postaje nula !!!

```
void main()
{









}
```

Rezultat za uneti broj 8612 je:

Zadatak 4. Napisati program na C jeziku za određivanje srednje vrednosti n realnih pozitivnih brojeva –pomoću while petlje. Testirati program za vrednosti n=4, brojevi su redom 34,345, 234,2876, 45,98, 453,32. Koliki je rezultat?

Zadatak 5. Razmotrite izraz : double ans = 18.0/squared(2+1). Izračunati vrednost izraza ako je funkcija makro squared() definisana kao:

A. `#define squared(x)  x*x`

B. `#define squared(x)  (x*x)`

C. `#define squared(x)  (x)*(x)`

D. `#define squared(x)  ((x)*(x))`

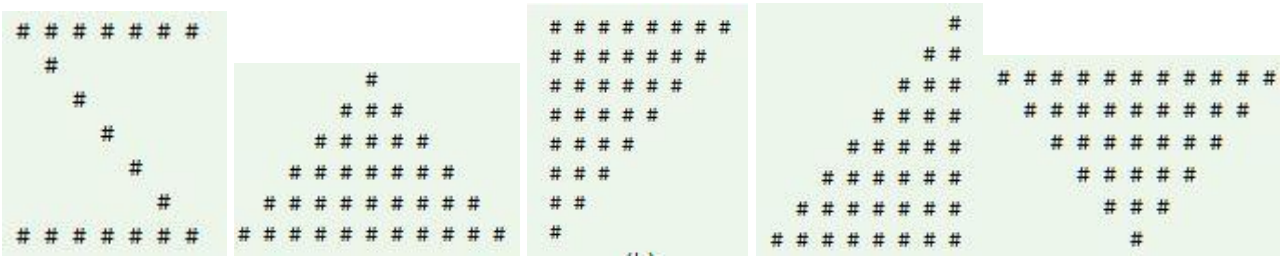
Zadatak 7 Napisati algoritam i program na programskom jeziku C koji cifre nekog proizvoljno unetog broja rotira za k pozicija u desno.

Zadatak 8 Napisati algoritam i program na programskom jeziku C koji računa zbir kvadrata cifara nekog proizvoljno unetog broja n.

Zadatak 9. Šta je mrtva petlja? Napisati primer mrtve petlje za for i while naredbe i objasniti.

Zadatak 10. Uradite broj zadatka koji dobijate kada vaš broj indeksa podelite po modulu 5. Radite samo jedan od sledećih zadataka.

Napisati program koji će za različito uneto n (za primer na slici uneto n je 11) štampati na standardni izlaz figuru na odgovarajućoj slici dobijenoj kada vaš broj indeksa podelite po modulu 5.



Slike 1, 2, 3, 4 i 5

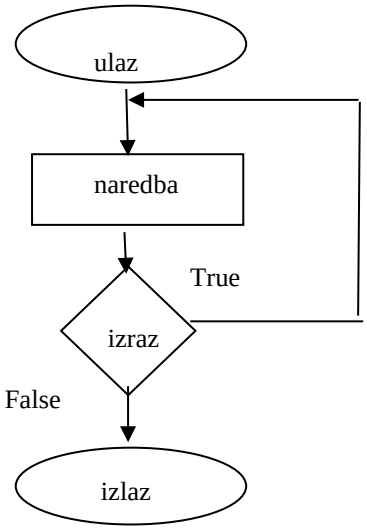
Pitanja za odbranu vežbe

1. Šta je mrtva petlja?
2. Primer mrtve petlje sa for naredbom.
3. Primer mrtve petlje sa while naredbom.
4. Koji je format naredba for?
5. Koji je format naredbe while?
6. Šta je predprocesorksa direktiva?
7. Šta je makro?
8. Šta u programskom jeziku označava ključna reč define?
9. Šta u programskom jeziku označava ključna reč include?
10. U čemu je razlika između naredbe i predprocesorske direktive?

VEŽBA 5 – do while petlja, switch case

Petlja sa ulaznim uslovom *do while*

U slučaju do while petlje obavezno izvršavanje bar jedne iteracije se postiže tako što je upravljački izraz petlje na samom dnu petlje. Tako se uslov petlje proverava nakon izvršenih naredbi tela petlje. Strukturni dijagram toka može se prikazati kao:



Zadatak 1. Napisati program za izračunavanje n! primenom do while strukture:

```
#include <stdio.h>
void main()           /* Program za izracunavanje faktorijela */
{
      int i, fak;
      long n ;
      i = 1; n = 1;
      printf("lzracunavanje n!\nUkucajte broj ? ");
      scanf("%d", &fak);
      do{


      } while (i <= fak);
      printf("%d! = %ld\n", fak,n);
}
```

Koliko puta se izvrši naredba n *= i. Napisati isti program koristeći samo for petlju:

Bezuslovno grananje : naredbe *break* i *continue*

Prekid izvršenja naredbi tela petlje se može realizovati i pre nego što uslovni izraz dobije logičku netačnu vrednost. Naredba kojom je moguće realizovati iskakanje iz upravljačke strukture se realizuje korišćenjem ključne reči **break**. Ova naredba inicira sledeća dva procesa:

- Prekid izvršenja naredbi tela petlje u upravljačkim strukturama while, for i do while iskakanjem iz tela petlje. Programski tok se nastavlja neposredno na prvoj naredbi iza naredbe upravljačke strukture.
- Preskakanje preostalih naredbi unutar višestrukog grananja, koje se ostvaruju naredbom switch,

Prekid izvršenja trenutnog ciklusa petlja, preskakanja svih ostalih naredbi i prelazak na sledeću iteraciju sa obavlja naredbom **continue.**

Naredba višestrukog grananja – Switch Case

Naredba višestrukog grananja je ekvivalentna upotrebi naredbe if then else if. Koristi se kada vrednost nekog izraza može imati više od dve vrednosti. Sintaksa narede je:

```
      switch (promenljiva)
      {
            case literarna vrednost: naredba; break;
            case literarna vrednost: naredba; break;
            case literarna vrednost: naredba; break;
            default naredba; break;
      }
```

Zadatak 2. Primer programa u kome se koristi naredba break u kombinaciji sa naredbom switch za izračunavanje broja samoglasnika u delu proizvoljnog teksta. Testirati program na svom imenu i prezimenu. Da li bi bilo razlike u radu programa ukoliko bi obrisali sve break naredbe?

```
void main()
{
      char ch;
      int a_ct,e_ct,i_ct,o_ct,u_ct, ostalo;
      a_ct=e_ct=i_ct=o_ct=u_ct=ostalo=0;
      printf("Unesi tvoje ime i prezime; Unesi # za izlaz.\n");
      while((ch=getchar())!= '#')
      {
            switch (ch)
            {
            case 'a' : a_ct++; break;
            case 'A' : a_ct++; break;



            default: ostalo++ break;
            }        /* kraj switch */
      }     /* dok petlji nije kraj */
      printf("Broj samoglasnika: A E I O U\n");
      printf("%4d %4d %4d %4d %4d\n", a_ct,e_ct,i_ct,o_ct,u_ct);
}
```

Izlaz iz programa je:

Zadatak 3. Sastaviti program na C jeziku za rešavanje sistema linearnih jednačina primenom Kramerovog pravila:

$$x_1 = \frac{\Delta_1}{\Delta}, x_2 = \frac{\Delta_2}{\Delta}, x_3 = \frac{\Delta_3}{\Delta}, \cdots\cdots, x_n = \frac{\Delta_n}{\Delta}$$

i praktično rešiti sistem jednačina:

$$x_1 + x_2 + 2x_3 = 9$$
$$4x_1 - x_2 + 3x_3 = 11$$
$$-2x_1 + 2x_2 + x_3 = 5$$

gde je : $\Delta = \begin{vmatrix} 1 & 1 & 2 \\ 4 & -1 & 3 \\ -2 & 2 & 1 \end{vmatrix}, \Delta_1 = \begin{vmatrix} 9 & 1 & 2 \\ 11 & -1 & 3 \\ 5 & 2 & 1 \end{vmatrix}, \Delta_2 = \begin{vmatrix} 1 & 9 & 2 \\ 4 & 11 & 3 \\ -2 & 5 & 1 \end{vmatrix}, \Delta_3 = \begin{vmatrix} 1 & 1 & 9 \\ 4 & -1 & 11 \\ -2 & 2 & 5 \end{vmatrix}$

```
void main()
{
      int x1, x2, x3;          int a1, a2, a3;          int b1, b2, b3;
      int c1, c2, c3;          int r1, r2, r3;

      scanf("%d %d %d %d", &a1, &b1, &c1, &r1);
      scanf("%d %d %d %d", &a2, &b2, &c2, &r2);
      scanf("%d %d %d %d", &a3, &b3, &c3, &r3);

      int d =  a1*b2*c3 + a2*b3*c1 + a3*b1*c2 - c1*b2*a3 - c2*b3*a1 - c3*b1*a2;
      int d1 = r1*b2*c3 + r2*b3*c1 + r3*b1*c2 - c1*b2*r3 - c2*b3*r1 - c3*b1*r2;
      int d2 = a1*r2*c3 + a2*r3*c1 + a3*r1*c2 - c1*r2*a3 - c2*r3*a1 - c3*r1*a2;
      int d3 = a1*b2*r3 + a2*b3*r1 + a3*b1*r2 - r1*b2*a3 - r2*b3*a1 - r3*b1*a2;

      x1 = d1/d;               x2 = d2/d;               x3 = d3/d;

      printf ("determinante: %d %d %d %d", d, d1, d2, d3);
      printf ("\npromenljive:  %d %d %d\n", x1, x2, x3);
}
```

Proveriti rešenja ovog sistema linearnih jednačina. Rešenje treba da bude: x_1=1, x_2=2 i x_3=3.

Zadatak 4. For petlja i do while petlja se mogu transformisati u while petlju. Transformišite sledeće primere

a) Prevesti dati izraz koristeći while petlju:

```
      int i , ret = 1;
      for ( i = 2; i <= n; i++)
            ret *= i ;
```

b) Prevesti dati izraz koristeći for petlju

```
int i=1;   int z = 0;
do {
      z = z + i;
      i++;
}
while (i <= 10 );
```

c) Prevesti dati izraz koristeći while petlju

```
int i=1, z = 0;
do {
      z = z + i;
      i++;
      while (i <= 10 );
}
return n;
```

Zadatak 5. Napisati program sa menijem:

1- Program matematika
2- Program finansija
3- Program zabave
4- Exit

upotrebom naredbe switch i case. Korisnik unosi broj od 1 do 4. Nakon unosa program štampa ime odabrane opcije i nudi mogućnost za novi izbor sve dok se ne izabere opcija 4 (Exit).

```
void main()
{









}
```

Zadatak 6: Napisati program koji ispisuje sve prirodne brojeve manje od n koji su prosti. Broj je prost ukoliko je deljiv samo sa 1 i sa samim sobom.

```
```

Zadatak 7: Napisati šta vraća sledeći kod? Šta bi vratio kod ukoliko bi uklonili naredbu break i zašto?

```
int main(int argc, const char * argv[]) {
      printf("TEST");
      int i;
      for (i = 0; i<5; i++)
      {
            switch (i % 3) {
            case 2: printf(" %d ", i++); break;
            case 1: printf(" %d ", ++i); break;
            default: printf(" %d ", 0); break;
            }
      }
      return 0;
}
```

```
Odgovor:
```

Zadatak 8: Uradite broj zadatka koji dobijate kada vaš broj indeksa podelite po modulu 5. Radite samo jedan od sledećih zadataka.

8.1 Napisati program kojim se štampaju svi trocifreni brojevi (ako ih ima) koji su jednaki sumi faktorijela svojih cifara.

8.2 Napisati program koji slučajno bira cele brojeve iz intervala od 1 - 1000 i zaustavlja se kada je izabran broj 500. Prikazuje poruku u kom je pokušaju odabran broj 500, i da li je bilo više slučajno odabranih brojeva iz intervala [1,499] ili intervala [501, 1000]. Koristiti funkciju rand() iz biblioteke stdlib.h.

8.3 Napisati program koji slučajno bira cele brojeve iz intervala 0 - 999, prikazuje izabran broj i prekida biranje brojeva kada je izabran broj 100, ili je izvršeno biranje više od 1000 puta. Koristiti funkciju rand() iz biblioteke stdlib.h.

8.4 Napisati funkciju koja štampa ceo broj N u binarnom zapisu, a zatim računa i štampa binarni broj i zbir svih cifara binarnog broja.

8.5 Napisati program kojim unosimo tekst do pojavljivanja znaka '.' pri čemu tekst unosimo slovo po slovo. Prikazati broj pojavljivanja svih samoglasnika i izraziti tu vrednost procentualno u odnosu na sve unete znake.

Pitanja za odbranu vežbe:

1. Čemu služi naredba continue? Primer.
2. Čemu služi naredba break? Primer.
3. Koji je format naredbe switch case?
4. Čemu služi naredba defaul u switch case naredbi?
5. Koji uslov mora da bude ispunjen da bi se tačka nalazila van u i na nekoj kružnici?
6. Koji uslov mora da bude ispunjen da bi se tačka nalazila ispod iznad i na nekoj pravoj?
7. Koji uslov mora da bude ispunjen da bi se neka tačka nalazila u nekom pravougaoniku?

VEŽBA 6 –Funkcije

1. Potprogrami i struktura programa

Potprogrami predstavljaju mehanizam koji direktno podržava funkcionalnu dekompoziciju kao jednu od osnovnih metoda strukturnog programiranja. Oni su samostalni segmenti programskog koda koji se pozivaju radi obavljanja konkretno specificiranog zadatka. Potprogram može biti neograničeno puta pozivan iz istog ili u okviru različitih programa. Programski jezik C sadrži module (biblioteke) u kojima se nalaze grupe deklaracija promenljivih i deklaracija potprograma kao jedinice fizičke dekompozicije. Ovakav pristup za direktnu posledicu ima stvaranje opštih programskih rešenja i mogućnost višestrukog korišćenja istog programskog koda u različitim softverskim proizvodima. Svaki potprogram se sastoji od:

- **Deklaracije** ili potpisa kojim se definiše interfejs potprograma prema programu, koja uključuje:
  - ime potprograma,
  - **listu parametara,** koja može biti i prazna,
  - i **tip povratne vrednosti**, Rezervisana reč **void** se koristi kao oznaka za povratnu vrednost kada funkcija ne vraća ništa.
- **Tela potprograma** koje se sastoji od deklaracija i naredbi.

Postoje dva tipa potprograma:

- ***FUNKCIJA**:* je segment programskog koda koji na osnovu nijednog, jednog ili više argumenata, uvek daje kao rezultat **jednu tačno određenu** vrednost koja se naziva vrednost funkcije.
- ***PROCEDURA*** je tip potprograma koji takođe može prihvatiti nijednu, jednu ili više vrednosti u obliku argumenata. Na osnovu prihvaćene vrednosti rezultat procedure može biti **nijedna ili više** vrednosti.

Definicija funkcije u opštem slučaju ima sledeći oblik:

```
povratni_tip    ime_funkcije    ( lista_argumenata )
{
     naredba
     naredba
     …
     return ime_promenljive
}
```

Izvršavanje se može prekinuti naredbama **return** ili **exit**.

Poziv funkcije

Prilikom kreiranja funkcije definiše se šta ta funkcija treba da radi. Da bi se funkcija izvršila ona se mora pozvati. Kada program pozove funkciju kontrola se iz glavnog programa prenosi na pozvanu funkciju. Funkcija se izvršava sve dok ne dođe do kraja ili do naredbe return. Tada funkcija završava izvršavanje i vraća kontrolu glavnom programu.

Da bi funkcija bila pozvana potrebno je da se navede njeno ime i proslede konkretne vrednosti parametara kako je predviđeno u njenoj deklaraciji. Ukoliko funkcija vraća vrednost, nju treba upotrebiti ili zapamtiti u odgovarajućoj promenljivoj kao što je ilustrovano sledećim primerom:

```
Int min(int a, int b)                          void main()
{                                              {
      if (a>b)                                       int x = 3;
            return b;                                int y = 8;
      else                                           int rez = min(3, 1);
            return a;                                int rez2 = min(x, y);
}                                              }
Prenos parametara funkciji
```

Postoje dva načina da prenesemo parametre funkciji.

1. Prenos parametara **po vrednosti** podrazumeva da sve što se desi u funkciji nema nikakvog uticaja na vrednost promenljivih u glavnom programu. Prilikom poziva funkcije prave se kopije prosleđenih parametara koje postoje samo dok funkcija ne završi sa izvršavanjem.

2. Prenos parametara **po referenci** podrazumeva da se sve promene promenljivih koje su na ovaj način prosleđene vide i u glavnom programu. Ovaj prenos se može koristiti kako bu funkcija vratila više od jedne

Main metoda i prenos argumenata preko komandne linije

Metoda ili funkcija main je glavna ulazna tačka u C program bilo da se radi o konzolnoj ili windows aplikaciji, dok biblioteke i servisi ne zahtevaju postojanje main metode. Kada se aplikacija startuje main metoda je prva koja se poziva i počinje sa izvršavanjem. Jedna C aplikacija može imati samo jednu ulaznu tačku, odnosno samo jednu main funkciju.

Main funkcija može imati kao povratnu vrednost void ili int. Obično se ova povratna vrednost koristi kako bi se signaliziralo da li je program uspešno izvršen ili je došlo do neke greške pri čemu će program u tom slučaju vratiti 0.

Main funkcija može imati dva parametra. Prvi parametar je tipa int i označava broj parametara koji su navedeni prilikom poziva funkcije iz konzolne linije. Drugi parametar je tipa char* [] i predstavlja niz stringova koji zapravo predstavljaju vrednosti navedenih parametara. Na primer ukoliko iz konzole pozovemo program sa zad saberi 1. Vrednost prvog parametra će biti 3, a vrednost drugog će biti niz koji će sadržati 3 elementa „niz“, „saberi“, i „3“.

Rekurzija

Rekurzija je pojava kada funkcija poziva samu sebe kao što je to prikazano sledećim primerom.

```
void recursion() {
   recursion(); /* function calls itself */
}
int main() {     recursion();     }
```

U programskom jeziku C rekurzivno pozivanje funkcije je dozvoljeno, ali se naročito mora voditi računa o definisanju kada se rekurzivni poziv funkcije završava. U protivnom postoji opasnost od kreiranja mrtve petlje.

Rekurzivni poziv funkcije predstavlja jedan od osnovnih programerskih metoda za rešavanje određenih klasa problema, naročito matematičkih. Primeri primene predstavljaju računanje faktorijela broja, fibonačijevog niza, implementaciju *quick sort* algoritma i druge.

Samostalni rad studenta

Zadatak 1: Poziv funkcija bez argumenata

```
#include<stdio.h>
#define IME "Visa tehnicka skola"
#define ADRESA "Beogradska 20"
#define MESTO "18000 Nis"
#define LIMIT 65
void zvezde();              /* deklaracija funkcije bez argumenata */

void main()
{
      zvezde();                    /* poziv korisnicke funkcije */
      printf("%s \n",IME);         /*poziv funkcije iz st. biblioteke*/
      printf("%s \n",ADRESA);
      printf("%s \n",MESTO);
      zvezde();
}

/*definicija korisnickih funkcija*/
void zvezde()          /*funkcija nema argumenata*/
{
      int brojac;
      for ( brojac=1; brojac <= LIMIT; brojac++)
            printf("%c",'*');
      printf("\n");
}
```

1.1 Sagledati tok glavnog programa i funkcije i utvrditi u kojoj programskoj liniji nastaju "tačke prekida" glavnog program i kada se kontrola prebacuje na izvršavanje funkcije.

1.2 Šta označava ključna reč void?

1.3 Zašto je neophodno prvo deklarisati funkciju zvezde()? Šta bi se desilo da nismo deklarisali funkciju? Da li je moguće napisati funkciju tako da se deklaracija i definicija funkcije obave zajedno? Kako?

1.4 Koristeći isti program umesto konstanti IME, ADRESA i MESTO uneti vase podatke. Kompajlirati, linkovati i izvršiti ovako modifikovan program.

1.5 Modifikovati funkciju zvezde tako da je moguće kao parametar proslediti broj zvezda koje treba odštampati. Pozvati 3 puta funkciju sa različitim parametrima iz glavnog programa.

Zadatak 2. Poziv funkcija sa 2 argumenata: Sastaviti program na C jeziku za izračunavanje minimuma dva cela broja. U glavnom programu obezbediti štampanje oba broja i njihovog minimum. Funkcija treba kao parametar da primi dva broja i da vrati manji od njih.

```
#include<stdio.h>
int imin ( int n, int m);        // Deklaracija funkcije

void main()                              /* glavni program */
{
      int broj1, broj2, vrati, rez;
      printf("Unesite 2 proizvoljna broja: \n");
      vrati=scanf("%d %d",&broj1,&broj2);
      rez = imin(broj1,broj2);
      if (vrati == 2)
      {
            printf("Manji od %d i %d je %d\n",broj1,broj2, imin(broj1,broj2));
            printf("Manji od %d i %d je %d\n",broj1,broj2, rez);
      }
}

int imin( int n, int m )         /* definicija funkcije */
{
      int min;
      if (n<m)
            min=n;
      else
            min=m;
      return min;
}
```

2.1 Šta vraća funkcija scanf?

2.2 Objasnite sledeću naredbu: printf("Manji od %d i %d je %d\n",broj1,broj2, imin(broj1,broj2));

Zadatak 3. Sastaviti program korišćenjem struktura IF...THEN....ELSE IF za izračunavanje funkcije y. U glavnom programu obezbediti štampanje rezultata, a u funkciji definisati vrednosti funkcije po datim uslovima. Brojevi su integer tipa.

$$y = \begin{cases} x1+x2, & x1<x2 \\ x1*x2, & x1=x2 \\ x1-x2, & x1>x2 \end{cases}$$

Napisati program:

Zadatak 4. Napisati program za izračunavanje hipotenuze pravouglog trougla. U glavnom program izvršiti unos vrednosti prve i druge katete i štampati vrednost hipotenuze. U funkciji izračunati vrednost hipotenuze.

```
#include<stdio.h>
float hipotenuza( float n, float m);        // Deklaracija funkcije

void main()                              /* glavni program */
{
      float kat1, kat2, hip;
      printf("Unesite 2 proizvoljna broja: \n");
      scanf("%f %f",&kat1,&kat2);
      hip = hipotenuza(kat1,kat2);
      printf("Hipotenuza je %4.2f\n", hipotenuza(kat1,kat2));
      printf("Hipotenuza je %4.2f\n",hip);
}

float hipotenuza( float n, float m )        /*   funkcija   -   definicija
funkcije */
{
      float rez = sqrt(n*n + m*m);
      return rez;
}
```

Zadatak 5: Izmeniti prethodni program tako da se parametri unose prilikom poziva funkcije preko komandne linije.

Zadatak 6. Sastaviti program za prenos argumenata pozivom po vrednosti.

```
#include <stdio.h>
void promeni(int u, int v)
{     int temp;
      printf("U funk. originalne vrednosti su: u = %d a v= %d.\n", u, v);
      temp=u;                u=v;             v=temp;
      printf("U funk. nakon promene: u = %d a v= %d.\n", u, v);
}

void main ( )
{     int x = 5, y = 10;
      printf("U programu originalne vrednosti: x = %d a y = %d.\n", x, y);
      promeni(x,y);
      printf("U programu nakon poziva funkcije je: x = %d a y = %d.\n", x,
y);
}
```

6.1 Napisati rezultate rada programa. Zašto promenljive x i y nisu zamenile vrednosti u glavnom program, a jesu unutar funkcije? Izmeniti i napisati kod programa funkcije *promeni* tako da promenljive x i y zamene mesta i u glavnom programu. Problem možete rešiti pomoću pokazivača ili referenci.

Zadatak 7: Sastaviti program za izračunavanje sume:

$$S = \sum_{K=5}^{N} \frac{X^K}{K! + N}$$

Sve promenljive su integer veličine. Za računanje stepena broja i faktorijela napisati sopstvene funkcije i njih koristiti prilikom računanja sume.

Zadatak 8: Napisati program koji zasebno računa zbir površina i zbir obima N unetih geometrijskih figura. Površinu i obim jedne geometrijske figure računati u posebnim funkcijama. Vrstu geometrijske figure student bira tako što svoj broj indeksa podeli po modulu 5.

1. Trougao
2. Romb
3. Pravougaonik
4. Valjak
5. Kupa

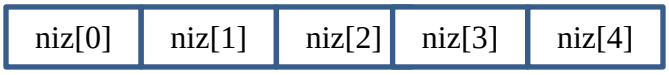
Pitanja za odbranu vežbe

1. Šta je to funkcija?
2. Šta je to procedura?
3. Šta je to povratna vrednost?
4. Šta radi naredba return?
5. Šta je to deklaracija funkcije?
6. Šta je to void?
7. Napisati primer funkcije bez parametra?
8. Napisati primer funkcije sa 2 parametra?
9. Napisati primer funkcije bez i sa povratnom vrednošću?
10. Da li funkcija može da vrati više od jedne vrednosti? Na koji način?
11. Kako se poziva funkcija iz glavnog programa?
12. Šta je to pokazivač? Primer?
13. Šta je to referenca? Primer?
14. Šta je potpis funkcije?
15. Kako glasi potpis funkcije printf?
16. Kako glasi potpis funkcije rand?
17. Kako glasi potpis funkcije pow?

VEŽBA 7 –Nizovi

1. Izvedeni tipovi podataka - nizovi

Niz je homogena struktura u kojoj su svi objekti istog tipa. Objekti u nizu se nazivaju **elementi niza**. Tip niza predstavlja tipa podataka svojih elemenata. Elementi niza mogu biti prosti tipovi podataka kao što su celi brojevi (int), realni brojevi (float) ili karakteri (char), ali niz može da sadrži i složene tipove podataka kao što su objekti, pokazivači ili reference. Za pristup elementu niza koristi se indeks elementa. U programskom jeziku indeks elemenata počinje od nule tako da prvi element niza ima indeks 0, drugi element ima indeks 1 itd. Osnovne operacije nad elementima niza predstavljaju dodavanje elementa u niz, menjanje elementa niza, brisanje elementa niza i sortiranje elemenata niza u rastući ili opadajući redosled.



Slika 6 - Elementi niza

Deklaracija Niza

Svaki niz mora biti deklarisan pre nego što se može koristiti:

```
type variable_name[length_of_array];
```

Ilustrujmo to primerima:

```
double height[10];    float width[20]; int min[9];      char name[20];
```

U C jeziku elementi niza počinju od **pozicije (indeksa) 0**. Maksimalni indeks je **za jedan manji od veličine niza (n-1).** Elementi niza se nalaze ne susednim memorijskim lokacijama. Samo ime niza u stvari predstavlja pokazivač na memorijsku lokaciju prvog elementa niza. Zato je moguće pristupiti bilo kom elementu niza navođenjem odgovarajućeg indeksa tog elementa:

```
M = height[0];
```

U prethodnom primeru promenljiva m će dobiti vrednost prvog elementa niza height.

Inicijalizacija Niza

Inicijalizacija niza predstavlja dodelu inicijalne vrednosti svim njegovim članovima. Sledeći kod ilustruje deklaraciju i inicijalizaciju niza:

```
int myArray1[5] = {1, 2, 3, 4, 5}; //istovremena deklaracija i
inicijalizacija


int myArray2[] = {1, 2, 3, 4, 5}; //istovremena deklaracija i
inicijalizacija
                        //bez navedene veličine niza

int studentAge[4];    studentAge[0]=14;     studentAge[1]=13;
                studentAge[2]=15;     studentAge[3]=16;
```

Dinamička deklaracija niza

Nizove je moguće deklarisati:

- statički – u statičkom delu memorije ili steku. U ovom slučaju nije moguće menjati veličinu niza nakon deklaracije.
- dinamički – u dinamičkom delu memorije ili heap-u

O dinamičkom alociranju memorije ćemo detaljnije učiti na kursu programski jezici II.

Sortiranje Nizova

Postoji nekoliko algoritama koji mogu sortirati niz. Neki od njih su:

- Selection sort – kod ove vrste sortiranja ideja je da se u prvom koraku minimalni element stavi na početak niza. U drugom koraku minimalni elemnt nesrtiranog dela niza dolazi na drugu poziciju i sve tako redom dok se ne sortira ceo niz
- Insertion sort – kod ove vrste sortiranja za svaki element niza traži se njegova pozicija u levom delu niza od njega. Od te pozicije svi elementi niza se zatim pomeraju za jedno mesto u desno kako bi napravili mesto za umetanje tog elementa. Jedan po jedan element niza se zatim umeću u sortirani deo niza sve dok ceo niz ne bude sortiran.
- Bubble sort – Ideja kod ovog sortiranja je da se uvek posmatraju dva susedna elementa. Ukoliko redosled elemenata nije odgovarajući, elemnti razmenjuju mesta. U prvoj iteraciji provera ide s' leva u desno. Do kraja niza nakon čega se dobija delimično uređen niz. Postupak se ponavlja sve dok se niz ne sortira.
- Quick sort – najefikasniji način sortiranja niza zasnovan na rekurziji.

Pokazivači

Pokazivač je promenljiva koja sadrži **adresu objekta ili funkcije**. Rad sa pokazivačkim promenljivama je veoma teško razumeti bez razumevanja koncepta indirekcije, odnosno koncepta posrednog pristupa. Bilo koji objekat ili promenljiva se pamti na tačno određenoj memorijskoj lokaciji. Pokazivači omogućuju indirektni pristup vrednostima programskih promenljivih korišćenjem njihovih memorijskih adresa. U neposrednoj vezi sa pokazivačima su dva specijalna operatora programskog jezika.

Adresni operator referenciranja & ili operator ***adresa-od*** je unarni operator vrlo visokog prioriteta koji daje memorijsku adresu objekta na koga je primenjen. Ovaj operator predstavlja način da se sazna memorijska lokacija gde je taj objekat smešten u memoriji računara. Asocijativnost ovog unarnog operatora je s desna na levo, a opseg vrednosti koji može da ima je skup ne negativnih celih brojeva (unsigned int).

Operator indirekcije ili dereferenciranja * omogućava deklarisanje pokazivačke promenljive i indirektni pristup do vrednosti za koje su korišćene pokazivačke promenljive. To znači da se on primeni na neku adresu u memoriji i vrati ono što je zapisano na toj adresi.

Pokazivači se često koriste u radu sa nizovima kako bi se lakše dolazilo do određenog člana niza. Samo ime niza zapravo jeste **pokazivačka promenljiva koja pokazuje na adresu prvog elementa niza**. Dozvoljene operacije su sabiranje i oduzimanje celobrojnih podataka, dodela vrednosti jednog pokazivača drugom, poređenje dva pokazivača, poređenje pokazivača i nule kao celobrojne vrednosti, oduzimanje vrednosti pokazivača istog tipa.

Niz kao parametar funkcije i povratna vrednost

Niz se kao parametar funkcije može proslediti na 3 načina. Svaki od načina je jako sličan i u suštini prenosi informaciju kompajleru da će pokazivač na neki tip podataka (u ilustrovanom primeru int) biti prosleđen funkciji. Sam kompajler ne vrši (*bounds checking*) odnosno ne proverava dužinu niza na koji pokazuje pokazivač koji se prosleđuje. Na sličan način se može proslediti pokazivač na višedimenzionalni niz.

```
void func(int *param)
void func(int param[10])
void func(int param[])
```

C programski jezik ne dozvoljava da vratite ceo niz kao povratnu vrednost funkcije, ali dozvoljava da vratite pokazivač na neki niz bez specificiranja njegove dužine. Povratna vrednost funkcije bi u tom slučaju izgledala ovako:

```
int * func(){ …}
```

Druga stvar koju treba zapamtiti je da se ne sme vratiti pokazivač na lokalnu promenljivu jer će ona biti uništena na kraju izvršavanja funkcije. Postoje tri načina kako da bezbedno vratite niz iz funkcije:

1. Prosledite niz kao formalni parametar
2. Dinamički alocirate memoriju za niz – u ovom slučaju se mora voditi računa da se sva memorija koja je dinamički alocirana mora i obrisati. Program je neće sam obrisati odnosno osloboditi.
3. Kreirate statičku lokalnu ili globalnu promenljivu koja predstavlja niz. Ovaj način se ne preporučuje i spada u lošu programersku praksu.

```
char *returnArray(char array []){
      char returned [10];
      return &(returned[0]);      // sta vraca funckija?
}

void foo(char *buf, int count) {
    for(int i = 0; i < count; ++i)
        buf[i] = i;
}

char *foo1(int count) {
      char *ret = malloc(count);
      if(!ret)   return NULL;
      for(int i = 0; i < count; ++i)
            ret[i] = i;
      return ret;
}

int main() {
      char arr[10] = {0};
      arr = returnArray(arr);
      foo(arr, 10);               // no need to deallocate, we allocated
                                  // arr with automatic storage duration.
      char *p = foo(10);
      if(p) { free(p);}     //  we  had  dynamically  allocated  it  (i.e.,
malloc or new)
}                     //variant) then we need to call free(p)
```

Samostalni rad studenta

Zadatak 1. Napisati program za unos i izračunavanje zbira članova niza sa 5 elemenata.

```
void main(void)
{
      float broj[5];                          //deklaracija niza
      double rez = 0;  int i;                 //pomocne promenljive

      for(i=0; i < 5; i++)              //uzmi podatke i smesti ih u broj
      {
            printf("Unesite broj %d : ", i+1);
            scanf("%f", &broj[i] );
      }

      for(i = 4; i+1 ; i--)                   //saberi
            rez += broj[i];
      printf("Zbir unesenih brojeva je %lf \n", rez);        //prikazi
rezultat
}
```

1.1 Testirajte program. Navedite unesene vrednosti za niz i rezultat:

1.2 Deklarisati niz od 4 elemenata tipa char i inicijalizovati sve članove niza na vrednost ‘a’. Ilustrovati oba načina inicijalizacije:

1.3 Dodeliti vrednost ‚m‘ drugom i četvrtom elementu niza.

1.4 Šta će se desiti ukoliko pokušamo da petom elementu ovog niza dodelimo vrednost ‘m’

Zadatak 2. Napisati program kojim se određuje redni broj maksimalnog elementa u celobrojnom nizu X od N elemenata.

Rešenje:

Zadatak 3. Primer prenosa argumenata funkcije pokazivačkim promenljivim za realizaciju sabiranja članova niza.

```
#define VELIC 10
long sump(int *pok, int vel);

void main() {
      static int niz[VELIC] = {20,10,5,39,4,16,19,26,31,20};
      long odgovor;
      odgovor = sump(niz, VELIC);
      printf("Suma niza je: %ld.\n", odgovor);
}

long sump(int *ar, int n) {       /* koristi pokazivacku aritmetiku */
      int i ;
      long ukupno = 0;
      for(i = 0; i < n; i++) {
            ukupno += *ar; /* dodaje vrednost na ukupno */
            ar++; /* pomera pokazivac na sledeci elemenat */
      }
      return ukupno;
}
```

Rezultat ovog programa je:

Zadatak 4. Napisati program za štampanje niza i množenje elemenata niza konstantnom vrednošću.

```
#include <stdio.h>
#define VELICINA 5

void main()
{
      double dip[VELICINA] = {20.0, 17.66, 8.2, 15.3, 22.22};
      int i;
      for(i = 0; i < VELICINA; i++)
            dip[i] *= mnozi;

      printf("\n");
      for(i = 0; i < n; i++)
            printf("%8.3f ", dip[i]);
      printf ("\n");
}
```

Napisati rezultate rada programa:

Zadatak 4.1. Izmeniti kod iz prethodnog zadatka tako da se svaki element niza uveća (umesto pomnoži) za neku konstantnu vrednost koja se unosi sa tastature.

Zadatak 4.2 Izmeniti kod iz prethodnog zadatka tako da se štampanje niza i množenje niza obavi u funkciji.

```
#include <stdio.h>
#define VELICINA 5

void prikaz_niza(double niz[], int n);
void uvecaj_niz(double broj, double *pok);

void main()
{




}

void prikaz_niza(                    )
{



}

void uvecaj_niz(                )
{



}
```

Zadatak 5. Razmotriti rad programa pisanog u C programskom jeziku, koji sortira dati vektor ar = {7,3,9,2,11,20,17,19,6,4} korišćenjem metode "bubble sort" algoritma. Program se najpre sastoji u pisanju funkcije "swap" koja vrši zamenu vrednosti dve promenljive i smešta ih u dve memorijske lokacije. Zatim tu funkciju koristimo da bi izvršili sortiranje datog vektora. "Bubble sort" ispituje svake dve ćelije vektora i ako nisu sortirane – zamanjuje im mesta. Ako se ovaj postupak izvrši n-1 put nad svim ćelijama vektora, on će biti kompletno sortiran.

```
#include <stdio.h>
void swap (int *a, int *b);

void main(){
      int ar[10] = {7, 3, 9, 2, 11, 20, 17, 19, 6, 4};
      int i,j,n;

      printf("Vektor pre sortiranja:\n");
      for(i=0;i<10;i++)
            printf("ar[%d]=%d\n",i,ar[i]);

      n=10;            /*broj clanova u vektoru koji se sortira*/
      for(i=0;i<n-1;i++)
            for(j=0;j<n-1;j++)
            {
                  if(ar[j]>ar[j+1])
                        swap(&ar[j],&ar[j+1]);
            }

      printf("\nVektor posle sortiranja:\n");
      for(i=0;i<10;i++)
            printf("ar[%d]=%d\n",i,ar[i]);
}

void swap ( int *a,int *b)
{
      int temp;
      temp=*a;
      *a=*b;
      *b=temp;
}
```

Prepisati rezultate rada programa:

Zadatak 6: Napisati program u programskom jeziku C koji određuje broj elemenata niza koji se javljaju samo jednom. Sve takve elemente smestiti u novi niz.

```
void main()
{
	int niz[1000];
	int i, j, n, b = 0;
	printf("Unesi n:");
	scanf_s("%d", &n);
	for (i = 0; i < n; i++)
	{
		printf("Unesi %d clan niza:", i + 1);
		scanf_s("%d", &niz[i]);
	}

	for (i = 0; i < n; i++){
		for (j = 0; j < n; j++)









		if (b == 0)
			printf("%d\t", niz[i]);
		b = 0;
	}
}
```

Zadatak 7: Napisati program u programskom jeziku C koji za dva uneta niza A i B proverava da li ovi nizovi imaju zajedničkih elemenata. Ukoliko zajednički elementi postoje, od njih se formira novi niz C i na kraju zadataka štampa na standardni izlaz. Primer A = { 1, 2, 3 ,4 ,5}; B = {2, 7, 4, 9, 8}; U ovom slučaju C će biti {2, 4};

```
void main()
{









}
```

Zadatak 8: Student radi samo jedan zadatak i to onaj koji se dobija kada se njegov broj indeksa podeli po modulu 5.

8.1. Napisati program u programskom jeziku C koji određuje uniju skupova: $C = A \cup B$. Smatrati da su skupovi zadati nizovima An i Bm, čije elemente zadaje korisnik.

8.2. Napisati program u programskom jeziku C koji određuje presek skupova: $C = A \cap B$

8.3. Napisati program u programskom jeziku C koji određuje skup D po formuli $D = (A \cup B) \setminus C$

8.4. Napisati program u programskom jeziku C koji određuje razliku skupova: $C = A \setminus B$

8.5. Napisati program u programskom jeziku C koji određuje skup D po formuli $D = (A \cap B) \setminus C$

Pitanja za odbranu vežbe:

1. Šta su nizovi i zašto i kada ih koristimo?
2. Deklarisati nekoliko nizova sa različitim tipovima podataka.
3. Deklarisati i inicijalizovati niz na 3 načina.
4. Izmeniti vrednost elemenata niza.
5. Navesti osnovne operacija koje se mogu vršiti na jednom nizu.
6. Navesti osnovne operacije koje se mogu vršiti sa dva niza.
7. Šta su pokazivači. Primer.
8. Šta je operator referenciranja. Primer.
9. Šta je operator dereferenciranja. Primer.
10. Navesti algoritme koje možemo koristiti za sortiranje jednog niza.
11. Šta je to dinamičko, a šta statičko deklarisanje niza.
12. Napisati program koji nalazi maksimalni element niza.
13. Napisati program koji nalazi minimalni element niza.
14. Napisati program koji sabira elemente niza.
15. Napisati program koji uvećava sve elemente niza nekom konstantnom vrednošću.
16. Napisati program koji množi sve elemente niza nekom konstantnom vrednošću.
17. Napisati funkciju koja kao parametar ima niz.
18. Napisati funkciju koja kao povratnu vrednost vraća niz.

VEŽBA 8 – Znakovni nizovi

Znakovni niz ili string predstavlja niz znakovnih podataka (jedan ili više). Primer konstantne vrednosti jednog stringa "test". Operacije za rad sa stringovima:

- Konkatenacija
- Poređenje
- Traženje
- Izdvajanje podniza

ASCII kod

ASCII kod sa brojem 32 je blanko simbol ili razmak. Prvih 32 ASCII kodova i svi ASCII kodovi veći ili jednaki 127 predstavljaju specijalne simbole koji se koriste u različite svrhe i najčešće nemaju neko jasno slovno značenje.

U C jeziku ne postoji poseban tip za predstavljanje niza znakovnih podataka, već se koristi niz char tipa. Na kraju svakog znakovno niza stoji null terminated symbol '\0'.

```
char c = ’a’;
char niz[] = „Pera“;
char niz2[] = {'P', 'e', 'r', 'a', '\0'};
```

U oba primera nisu date veličine niza i njih će odrediti prevodilac. Kod drugog načina neophodno je eksplicitno navesti null terminated character.

Naredba scanf za učitavanje karakternih nizova. %c znakovna konverzija (rezultat je tipa char), %s konverzija u znakovni niz (niz znakova između dva blanko znaka što znači da učitani niz ne sadrži bl. znake. Iza pročitanih znakova dodaje se ‘\0’)

U C jeziku nije dozvoljeno napisati:

```
char ime[20];
char prezime[20];
Ime = ''MIlos''
Prezime = ''m.'' + ime;
```

Ako su **s1** i **s2** C "stringovi" program **ne može**:

1. da dodeli vrednost jednog stringa drugom : **s1 = s2;**
2. da ih upoređuje: **. . . s1 < s2 . .**
3. da uradi konkatenaciju u jedan string: **. . . s1 + s2 . . .**
4. da funkcija kao rezultat vrati string.

Funkcije za unošenje i štampanje karakternih nizova su definisane u biblioteci #include <cstdio> ili <cstdio.h>:

- int getc (FILE * stream);
- Int getchar(); ekvivalenta naredba kao getc(stdin); Int c = getchar();, povratna vrednost je int ili EOF koja je definisana u cstdio biblioteci
- int putc (int character, FILE * stream);
- int putchar (int character);
- int fputc (int character, FILE * stream);

- char * gets (char * str) - Čita karaktere sa standardnog ulaza (stdin) i smešta ih kao C string u promenljivu str sve dok se ne dođe do karaktera za novi red (“\r\n”) ili kraj fajla (EOF). Karakter za novi red se ne kopira u str. Terminacioni null karakter („\0“) se automatski dodaje na kraj str promenljive. Ukoliko je funkcija uspešna, vraća str promenljivu. U suprotnom vraća se null pokazivač.
- char * fgets (char * str, int num, FILE * stream);
- int puts(char* str);
- int fputs (const char * str, FILE * stream);

Funkcije za rad sa nizovima karaktera iz biblioteke <cstring>

- char * strcpy (char * destination, const char * source);
- size_t strlen (const char * str);
- char * strcat (char * destination, const char * source);
- int strcmp (const char * str1, const char * str2) – Povratna vrednost je manja od 0 ako je prvi karakter iz str1 koji je različit od njemu odgovarajućeg iz str 2 ima manju int vrednost. Jednaka je 0 ako su stringovi isti
- int fflush (FILE * stream); Bilo koji nezapisan podatak (koji je trenutno baferovan) ce biti istog trenutka zapisa u stream.

Primer:

```
char rec1[20] = "milos", rec2[20] = "kosanovic", rec3[20];
printf("%d\n", strlen(rec1));
printf("%d\n", strlen(rec2));
printf("%s\n", strcat(rec1, rec2));
printf("%d\n", strlen(rec1));

strcpy(rec3, rec1);
printf("%s \n %s\n %s\n", rec1, rec2, rec3);

printf("%d\n", strcmp(rec1, rec2));
printf("%d\n", strcmp(rec1, rec3));
```

Samostalni rad studenta:

Zadatak 1. Sastaviti program na programskom jeziku C za ispisivanje tablice ASCII kodova za sve znakove. Znaci se prikazuju po kolonama. U jednoj koloni se prikazuje po 19 znakova, a u jednom redu se prikazuje po 5 znakova (5x19=95). Prikazaćemo sve ASCII kodove od 32 do 126.

```
#include <stdio.h>
void main()
{
      char c=' ';
      int i;
      printf ("\t\tTablica ASCII kodova \n \n");
      for (i=0; i<95; i++)
            printf("%3d %c     ",c+i,c+i);
            if (i%19 == 0)
                  printf("\n");
}
```

Koji je kod za karaktere ‘a’, ‘A’, ‘#’?

Zadatak 2 Poređati sledeći niz naredbi u odgovarajući redosled tako da program odštampa jednostavnu poruku “Ja sam student prve godine”

```
return 0;
const char msg[] = MSG1;
}
#define MSG1 "All your base are belong to us!"
int main(void){
#include <stdio.h>
puts(msg);
```

Zadatak 3. Napisati program za unos i štampanje karaktera**.** Program učitava i broji karaktere sa ulaza sve dok se ne otkuca znak za kraj unosa EOF. EOF se može generisati kucanjem Control + Z, a zatim enter.

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




}
```

Zadatak 4. Sastaviti program koji iz skupa unetih karaktera (na primer: ''Ja sam rodjen 1990. god u 12. mesecu, i u petak!'') izračunava broj unetih slova, brojeva i ostalih karaktera. Tekst se unosi sa tastature, a program se prekida pritiskom na ctrl+Z, a zatim enter.

```
#include<conio.h>
void  main( )
{









}
```

Zadatak 5. Napisati program konvertuje velika u mala slova. Rečenica se unosi sa tastature a program se prekid unošenjem karaktera %.

```
void main( ) {









}
```

Zadatak 6 Napisati program na programskom jeziku C koji učitava dva znakovna niza, čije se dužine unose kao podatak sa tastature, izvrši nadovezivanje drugog na prvi, okrene "naopako" dobijeni niz i ispiše ga na standardnom izlaznom uređaju. Znakovni nizovi su maksimalne dužine 200 karaktera.

```
#include <stdio.h>    #include <string.h>   #include <stdlib.h>
void main( ) {









}
```

6.1 Da li je moguće deklarisati niz sledećim izrazom int niz[n], gde je n neka promenljiva. Zašto?

```

```

6.2 Napisati primer unetih vrednosti za niz i dobijeni rezultat.

6.3 Napisati koje još naredbe znate za rad sa nizovima i kratko objasniti šta rade:

6.4 Da li je moguće promeniti veličinu već deklarisanog niza. Zašto?

Zadatak 7. Sastaviti program na jeziku C za učitavanje imena gradova uz njihovo uređivanje po abecednom redosledu i ispisivanje rezultata. U svakom redu se učitava po jedno ime sve dok se ne učita prazan red. Za upoređivanje imena gradova koristiti funkciju strcmp).

Zadatak 8: Sledeći kod prijavljuje grešku na liniji 2.

```
  char *str = "string";
  str[0] = 'z';
  printf("%s", str);
```

dok sledeći kod radi bez prijavljivanja greške. Zašto?

```
char str[] = "string";
  str[0] = 'z';
  printf("%s", str);
```

Zadatak 9: Student radi samo jedan zadatak i to onaj koji se dobija kada se njegov broj indeksa podeli po modulu 5.

9.1. Sastaviti program na jeziku C koji određuje i prikazuje dužinu unetog stringa bez korišćenja bibliotečkih funkcija.

9.2. Sastaviti program na jeziku C koji u zadatom stringu određuje redni broj najduže reči. Reči u rečenici su razdvojene blanko znacima (jednim ili više). Prikazati redni broj najduže reči.

9.3. Sastaviti program na jeziku C koji na osnovu unetih naziva ulica određuje i prikazuje da li osobe žive u ulici.

9.4. Sastaviti program na jeziku C koji na osnovu unete reči štampa da reč sadrži malo ili veliko slovo z.

9.5. Sastaviti program na jeziku C koji na osnovu unete reči štampa da li reč sadrži više slova „a“ ili „e“. (Gledaju se i velika slova „A“ i „E“)

Pitanja za odbranu vežbi

1. Šta vraća funkcija getchar().
2. Koji parametar treba proslediti funkciji getc, da bi se ona ponašala ekvivalentno kao funkcija getchar?
3. Koji je karakter za novi red?
4. Šta vrača funkcija strcmp i koji su argumenti ove funkcije?
5. Šta vrača funkcija strcpy i koji su argumenti ove funkcije?
6. Šta vrača funkcija strcat i koji su argumenti ove funkcije?
7. Da li je moguće deklarisati niz sledećim izrazom int niz[n], gde je n neka promenljiva. Zašto?
8. Da li je moguće promeniti veličinu već deklarisanog niza. Zašto?
9. Napisati program koji učitava rečenicu sa tastature.
10. Napisati program koji učitava tekst koji se unosi sa tastature sve dok se ne unese znak za novi red.
11. Napisati program koji učitava tekst koji se unosi sa tastature sve dok se ne unese znak #.

VEŽBA 9 –Matrice

Multidimenzijalni nizovi

U C jeziku mogu postojati i višedimenzijalni nizovi. Probajmo da to ilustrujemo primerom jedne matrice dimenzija 4 x 5:

		Broj kolone (j)				
		0	**1**	**2**	**3**	**4**
Broj vrste (i)	**0**	11	3	5	-9	-6
	1	5	6	-8	7	24
	2	-8	9	2	12	45
	3	10	13	-10	4	5

Neka je ime naše matrice X. Za pristup elementu matrice imamo da koristimo dva indeksa, jedan za red a drugi za vrstu matrice. Izraz bi u ovom slučaju izgledao **X [i] [j]**, gde je i indeks reda, a j indeks kolone. To znači da bi X [0] [0] referencirao vrednost 10, X [2] [1] referencira vrednost 9. Ukratko, multidimenzijalni nizovi se definišu na isti način kao i obični nizovi samo što korite veći broj indeksa za pristup svojim elementima. Mi ćemo za sada raditi samo sa dvodimenzijalnim nizovima. Deklarisanje ovakvog niza se može uraditi ovako:

```
float table [50] [50];
char line [24] [40];
```

Prvi primer definiše matricu (dvodimenzionalni niz) float vrednosti koja ima 50 vrsta i 50 kolona. To znači da će ukupan broj elemenata biti 50x50=2500. Drugi primer deklariše matricu char elemenat sa 24 vrsta i 40 kolona. Ukupan broj elemenata je 24 x 40 = 1920 elemenata.

Pokažimo još jedan primer matrice 3 x 4 sa vrednostima {1, 2, 3, 4, 5, 6, 7, 8, 9, 10. 11, 12, };

```
Values [0][0] = 1 Values [0][1] = 2 Values [0][2] = 3 Values [0][3] = 4
Values [1][0] = 5 Values [1][1] = 6 Values [1][2] = 7 Values [1][3] = 8
Values [2][0] = 9 Values [2][1] = 10 Values [2][2] = 11 Values [2][3] = 12
```

Prvi indeks označava vrstu a drugi kolonu. Prvi indeks može uzeti vrednosti od 0 do 2 što ukupno čini 3 vrst. Drugi indeks može uzeti vrednosti od 0 do 3 što ukupno čini 4 kolone. Ova matrica se može deklarisati i inicijalizovti i na sledeći način:

```
int values [3] [4] = {      {     1, 2, 3, 4 }
                            {     5, 6, 7, 8 }
                            {     9, 10, 11, 12    }
};
```

Ova matrica će u memoriji biti zapamćena u nizu susednih memorijskih lokacija.

Zadatak 1. Sastaviti program na programskom jeziku C koji štampa sadržaj kvadratne matrice 5 * 5 vrstu po vrstu, a potom sporednu dijagonalu, sleva-udesno. Matricu formirati generatorom slučajnih brojeva RAND().

```
#include <stdio.h>
#include <cstdlib.h>
void main(){
      srand (time(NULL));

      int a[5][5], n, i, j;
      for (i=0; i<5; i++)
      {
            /*alocira memoriju za n elemenata vrste koji su tipa int*/
            for (j=0; j<5; j++)
                  a[i][j] = rand()%10;
      }

      for (i=0; i<5; printf("\n"), i++)
            for (j=0; j<5; j++)
                  printf(" %d", a[i][j]);
      printf("\n");

      for (i=4; i>=0; i--)
                  printf(" %d", *(*(a+i)+5-1-i));
            printf("\n");
}
```

Prepisati rezultate rada programa. Napisati na drugi način izraz printf(" %d", `*(*(a+i)+5-1-i)`).

Zadatak 2. Neka je data kvadratna matrica u formatu:

$$\begin{Vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{Vmatrix}$$

Članove matrice uneti u program kao konstantne celobrojne veličine, a kao izlaz iz programa predvideti štampanje ove matrice u pokazanom formatu kao polazne matrice i štampanje nove matrice u istom obliku ali sada unazad od člana $a_{44}, a_{43}, a_{42} a_{41}, \ldots\ldots, a_{11}$.

```
void main()
{









}
```

Napisati rezultate rada programa:

ulazni nizovi:	obrnuti izlazni niz

Zadatak 3. Napisati program na C++ jeziku koji izračunava sumu svih elemenata u svakom redu pojedinačno, pravougaone matrice [4x4].

$$A=\begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{vmatrix} \longleftarrow \longleftarrow \longleftarrow$$

```
void main()
{
	int mat[4][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
	int i,j,s=0;

	printf("\nPolazna matrica je\n");
	for(i=0;i<4; i++)
	{
		for(j=0;j<4; j++)
			printf("%2d ",mat[i][j]);
		printf("\n");
	}

	for(i=0;i<4; i++)
	{




	}
}
```

Napisati rezultat rada programa:

Zadatak 4. Napisati program na C jeziku koji će izračunati sumu elemenata na glavnoj dijgonali proizvoljne matrice dimenzija 5x5.

Zadatak 5. Napisati program na C jeziku koji će izračunati broj elemenata matrice većih od 10. Sve elemente matrice koji su veći od 10 smestiti u novi niz i izračunati njihov zbir.

Zadatak 6. Napisati program na C jeziku koji će izračunati zbir dve matrice.

Zadatak 7: Napisati funkciju na C-u kojom se elementi na sporednoj dijagonali matrice A (n x n) uređuju u rastući redosled, a ostali elementi zamenjuju jedinicama. U glavnom programu učitati kvadratnu matricu A, i prikazati matricu pre i posle poziva funkcije. Za potrebe prikazivanja matrice na ekranu napisati zasebnu funkciju

Zadatak 8: Sastaviti program na jeziku C koji u matrici A_{NxN} vrši zamenu mesta elementima:

1. k-te i l-te vrste,
2. k-te i l-te kolone

Zadatak 9: Student radi samo jedan zadatak i to onaj koji se dobija kada svoj broj indeksa podeli po modulu 5. Sastaviti program na jeziku C koji:

1. Računa zbir elemenata: 3. vrste, a zatim menja vrednost svih elemenata 1. Kolone i postavlja ih na nula.
2. Računa zbir elemenata: Na glavnoj dijagonali, i nakon toga štampa sve elemente koji se na njoj nalaze
3. Računa koliko ima elemenata većih od 10 iznad sporedne dijagonale i sve takve elemente množi sa dva
4. Računa koliko ima elemenata ispod sporedne dijagonale većih od 5 koji
5. Razliku elemenata: Ispod sporedne dijagonale

VEŽBA 10 –Rad sa fajlovima

Upis i čitanje iz fajla

U drugoj laboratorijskoj vežbi smo učili kako da izvršimo upis sa standardnog ulaza, odnosno ispis na standardni izlaz u programskom jeziku C. U ovoj vežbi ćemo naučiti kako da kreiramo, otvorimo, upišemo, pročitamo i zatvorimo tekstualni ili binarni fajl.

Otvaranje fajla

Za otvaranje fajlova se koristi funkcija **fopen.** Svaki fajl se mora otvoriti pre korišćenja i zatvoriti nakon korišćenja kako bi sve promene na fajlu bile zapamćene. Ukoliko fajl ne postoji ova funkcija će ga kreirati, otvoriti, inicijalizovati objekat tipa **FILE**, koji sadrži sve informacije neophodne za kontrolu toka (stream) podataka. Pogledaj mo deklaraciju ove funkcije:

```
FILE *fopen( const char * filename, const char * mode );
```

U ovom primeru filename je string literal i predstavlja putanju i ime fajla. Mode predstavlja pristupnu opciju i može imati sledeće vrednosti:

Tabela 7 – Opis različitih opcija za otvaranje fajla

Mod	Opis
r	Otvara trenutni tekstualni fajl kako bi se izvršilo čitanje.
w	Otvara tekstualni fajl kako bi se izvršio upis. Ako fajl ne postoji , novi fajl će biti kreiran i program će izvršiti upis na početak fajla.
a	Otvara tekstualni fajl za upis. Ako fajl ne postoji novi fajl će biti kreiran. Upis će izvršiti dodavanje sadržaja na već postojeći sadržaj fajla.
r+	Otvara tekstualni fajl za čitanje i pisanje.
w+	Otvara tekstualni fajl za čitanje i pisanje. Sav sadržaj fajla se briše. Ukoliko fajl ne postoji kreira se novi fajl.
a+	Otvara teksualni fajl za čitanje i pisanje. Kreira fajl ukoliko ne postoji. Čitanje će početi na početku fajla, ali pisanje će izvršiti dodavanje sadržaja na kraju fajla.

Ukoliko želite da koristite binarne fajlove onda morate koristiti sledeće opcije:

```
"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"
```

Zatvaranje fajla

Za zatvaranje fajla koristite funkciju fclose(). Deklaracija ove funkcije je:

```
int fclose( FILE *fp );
```

Ova funkcija vraća nulu ukoliko je poziv uspešan ili **EOF** ukoliko postoji greška pri zatvaranju fajla. Ova funkcija ustvari prazni bafer koji se koristi za rad sa fajlom, zatvara fajl i oslobađa memoriju koja se koristila za fajl**. EOF je konstanta definisana u zaglavlju stdio.h fajla i najčešće ima vrednost -1.**

Upis u fajl

Najjednostavnija funkcija za upis jednog karaktera u fajl je:

```
int fputc( int c, FILE *fp );
```

Ova funkcija vrši upis jednog karaktera koji ima vreedsnot c na tok podataka referenciran sa fp. U našem primeru fp će referencirati neki fajl. Funkcija će prilikom upisa izvršiti internu konverziju karaktera u tip unsigned char. Funkcija vraća karakter koji je upisan u slučaju da je funkcija uspešno izvršena, u suprotnom vraća EOF.

Sledeća funkcija se moće koristiti za upis *null-termineted* stringa:

```
int fputs( const char *s, FILE *fp );
```

Ova funkcija vrši upis stringa s u tok podataka referenciran sa fp. Ukoliko je funkcija upešno izvršena vraća se neka nenegativna vrednost. U slučaju greške vraća se EOF. Moćete koristiti i funkciju **int fprintf(FILE *fp,const char *format, ...)** za upis jednog stringa u fajl.Ilustrujmo to primerom:

Čitanje iz fajla

Sledeća funkcija vrši čitanje jednog karaktera u fajl:

```
int fgetc( FILE * fp );
```

Funkcija fgets() vrši čitanje jednog karaktera iz toka podataka referenciranog sa fp. Povratna vrednost funkcije je pročitani karakter ukoliko je ona uspešna. Ukoliko je došlo do greške funkcija će vratiti EOF. Sledeća funkcija omogućava čitanje stringa iz ulaznog toka podataka:

```
char *fgets( char *buf, int n, FILE *fp );
```

Funkcija fgets() čita do n-1 karaktera sa ulaznog toka podataka referenciranog sa fp. Ona kopira pročitani string u bafer buf i dodaje null karakter kako bi završila string. Ako ova funkcija pročita karakter za novi red '\n' ili karakter za kraj fajla EOF pre nego što je pročitala svih n-1 karaktera ona će prekinuti dalje čitanje podataka. Zadnji karakter koji će vratiti će biti znak za novi red.

Za čitanje stringa se može koristiti i funkcija **int fscanf(FILE *fp, const char *format, ...)**. Ona će prekinuti čitanje nakon prvog blanko znaka. Ukoliko je uspena funkcija će vratiti broj argumenata koji su pročitani. Na primer funkcija `fscanf(fp, "%s %s", ime, prezime);` će vratiti vrednost 2 ukoliko je uspešna ili EOF ako je neuspešna. Ilustrujmo ovo primerom:

Zadatak 1: Primer upisa u fajl.

```
#include <stdio.h>
void main()
{
   FILE *fp;

   fp = fopen("test.txt", "w+");
   fprintf(fp, "This is testing for fprintf...\n");
   fputs("This is testing for fputs...\n", fp);
   fclose(fp);
}
```

1.1: Šta će se desiti ukoliko fajl **"test.txt"** ne posotji?

```
Biće kreiran novi fajl.
```

1.2: Gde će se ovaj fajl kreirati? Zašto?

U istom direktorijumu gde se nalaze i source datoteke projekta.

1.3: Šta bi se desilo da je pristupni mod bio **a+**?

1.4: Kojoj biblioteci (fajlu) pripadaju funkcije

Zadatak 2: Primer čitanja iz fajla:

```
void main()
{
   FILE *fp;
   char buff[255];
   fp = fopen("test.txt", "r");
   fscanf(fp, "%s", buff);
   printf("1 : %s\n", buff );

   fgets(buff, 255, (FILE*)fp);
   printf("2: %s\n", buff );

   fgets(buff, 255, (FILE*)fp);
   printf("3: %s\n", buff );
   fclose(fp);
}
```

2.1 Šta radi prethodni zadatak i šta će on ispisati na ekranu:

Zadatak 3: Fajlovi zad3.txt sadrži podatke o poenima koje su studenti osvojili na ispitu iz predmeta programski jezici 1. Za svakog studenta je zapamćen broj poena koje je student dobio na ispitu. Odštampati imena svih studenata koji su položili ispit. Ispit su položili svi studenti koji imaju više od 50 poena. Listu odštampati na standardni ulaz i u fajl rezultati.txt. Primer fajla zad3.txt je:
Pera Peric 30
Ivana Stojicic 65
Marko Knezevic 45
Milutin Milankovic 80

```
void main()
{
	FILE *fp, *fpr;
	char ime[20], prezime[20];
	int i=1, rez=0, r1=0, r3=0;
	fp = fopen("zad3.txt", "r");
	fpr = fopen("rez.txt", "w");
	while (fscanf(fp, "%s", ime) != EOF)
	{
		r1 = fscanf(   /*to do   */ );
		r3 = fscanf(   /*to do   */  );

		if (//to do)
		{
			printf("%d: %s %s %d \n", i, ime, prezime, rez);
			fprintf(fpr, "%d: %s %s %d \n", i, ime, prezime, rez);
//			printf("%d: %d %d \n", i, r1, r3 );
			i++;
		}
	}
	fclose(fp);
}
```

3.1 Gde se nalazi datoteka rez.txt i koji je njen sadržaj nakon izvršenja programa?

3.2 Zašto se u naredbi fscanf(fp, "%s", prezime); promenljiva prezime koristi bez operatora referenciranja &, a promenljiva rez u naredbi fscanf(fp, "%d", &rez) se koristi sa ovim operatorom?

3.3 Da li možete da napišete program tako da se sve tri promenljive ime, prezime i rez čitaju odjednom? Dovoljno je napisati samo deo koda koji se razlikuje od trenutnog rešenja. Koliku će vrednost u tom slučaju vratiti funkcija fscanf?

Zadatak 4: Napisati funkciju koja će za primer fajla iz prethodnog zadatku izračunati prosečni broj poena za ispit koji su studenti polagali.

```

```

Zadatak 5: Fajl zad4.txt sadrži odlomak jednog našeg poznatog romana. Tokom kucanja odlomka greškom je umesto slova “lj” kucano slovo q i umesto slova “nj”, slovo w . Izvršiti ispravku grešaka pri kucanju ovog odlomka i unete ispravke upisati u fajl rez4.txt. Izbrojati koliko je izmena ukupno bilo. Primer sadržaja fajla zad4.txt je: “Vesti o qubicastom cvetu qiqanu mozete naci na wegovm blogu”. Napisati program, šta štampa program na standardni izlaz i sadržaj fajla rez4.txt nakon izvršenja programa.

```
void main()
{
      FILE *fp, *fpr;
      int c, i=0;
      fp = fopen("zad5.txt", "r");
      fpr = fopen("rez5.txt", "w");
      while ((c = fgetc(fp)) != EOF)
      {









      }
      fclose(fp);
      printf(“Bilo je ukupno %d promena”, i);
}
```

Zadatak6: U ulaznoj datoteci se nalaze celi brojevi (svi brojevi su manji od 1000). U izlaznu datoteku prepisati brojeve iz ulazne datoteke koji su deljivi sa 3 i 7.

Zadatak 7: Napisati program u programskom jeziku C koji briše specifičan red txt fajla.

Zadatak 8: Na sadržaj već postojeće datoteke ”dat.txt”, dopisati ”Zdravo svima”.

Zadatak 9: Podeliti vaš broj indeksa po modulu 5. Student radi samo JEDAN zadatak i to onaj koji je dobijen kao rezultat deljenja po modulu. Napisati program koji kao parametar pri startovanju programa dobija imena ulazne i izlazne datoteke.

9.1 Ulazni fajl sadrži spisak studenata (imena i prezimena studenata). Urediti spisak po početnom slovu prezimena studenata. Ukoliko ima studenata čije prezime počinje istim slovom urediti ih na osnovu početnog slova imena. Ako su oba početna slova ista redosled nije bitan. U izlazni fajl upisati ovako uređen spisak studenata, s tim što, ispred imena studenta, treba dodati njegov redni broj.

9.2 U ulaznoj datoteci se nalaze celi brojevi. U izlaznu datoteku prepisati brojeve iz ulazne datoteke koji su pozitivni i prosti.

9.3 U ulaznoj datoteci se nalaze imena gradova u Srbiji U izlaznu datoteku prepisati sve gradove koji se sastoje iz dve reči ili one čije je ime duže od 10 slova.

9.4 Ulazni fajl sadrži spisak studenata (imena, prezimena i broj položenih ispita). Urediti spisak po broju položenih ispita. Ukoliko ima studenata koji imaju jednak broj položenih ispita urediti ih na osnovu početnog slova prezimena. Ako su oba početna slova prezimena ista redosled nije bitan. U izlazni fajl upisati ovako uređen spisak studenata, s tim što, ispred imena studenta, treba dodati njegov redni broj.

9.5 Ulazni fajl sadrži spisak studenata (imena, prezimena i godište rođenja). Za svakog studenta kreirati šifru i upisati je u izlazni fajl. Šifra se sastoji od prva tri slova imena studenta, njegove godine rođenja, zadnja dva slova prezimena studenta i broja samoglasnika u njegovom imenu i prezimenu. Primer: Milos Kosanovic 1986 --> Mil1986ic6

Pitanja za odbranu vežbe

1. Koji je format naredbe za otvaranje fajla?
2. Koji je format naredbe za zatvaranje fajla?
3. Koji je format naredbe za čitanje iz fajla?
4. Koji je format naredbe za upis u fajla?
5. Koji sve modovi postoje prilikom otvaranja fajla?
6. Koji je format naredbe fgetc?
7. Koji je format naredbe fgets?
8. Koji je format naredbe fscanf?