Слободанка Ђенић
Јелена Митић ▪ Светлана Штрбац

Основи програмирања на језику "C"

Збирка примера и задатака
за лабораторијске вежбе из предмета
Основи програмирања

Висока школа електротехнике и рачунарства
струковних студија
Београд, 2019.

Аутори: Слободанка Ђенић, Јелена Митић, Светлана Штрбац

Рецензенти: Ласло Краус, Слободан Обрадовић, Дивна Вучковић

Лектор: Анђелка Ковачевић

Корице: Јелена Лабус (MASSVision)

Издавач: Висока школа електротехнике и рачунарства струковних студија у Београду

Штампа: Развојно-истраживачки центар графичког инжењерства ТМФ, Београд

Тираж: 300

Издање: 13.

ISBN 978-86-7982-149-2

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004.42(075.8)(076)
004.432.2C(075.8)(076)

ЂЕНИЋ, Слободанка, 1965 -
Основи програмирања на језику "C" : збирка примера и задатака за лабораторијске вежбе из предмета Основи програмирања / Слободанка Ђенић, Јелена Митић, Светлана Штрбац. - 13. изд. - Београд : Висока школа електротехнике и рачунарства струковних студија, 2019 (Београд: Развојно-истраживачки центар графичког инжењерства ТМФ). – 189 стр. : илустр. ; 24 cm

Тираж 300. – Библиографија: стр. 189.

ISBN 978-86-7982-149-2

1. Митић, Јелена, 1966 - [аутор] 2. Штрбац, Светлана, 1972 - [аутор]
a) Програмирање – Вежбе b) Програмски језик "C" – Задаци
COBISS.SR-ID 273533452

Предговор

Овај приручник намењен је студентима прве године Високе школе електротехнике и рачунарства струковних студија у Београду, али и свима који су изабрали да науче основе програмског језика *C* и да на тај начин уђу у свет програмирања.

Програмски језик *C* написао је *Dennis Ritchie* 1972. године, са циљем да развије језик погодан за системско програмирање. Успео је у томе. Језик *C* је најпре искоришћен за развој *UNIX* оперативног система, а онда и за развој највећег дела свих осталих оперативних система. И данас је језик *C* незаменљив у области системског програмирања. У међувремену језик је претрпео извесне измене због делова синтаксе, који нису били од почетка јасно дефинисани. Данас се користи *ANSI C*, који је усвојио Амерички институт за стандарде 1988. године

Ево разлога из којих професионални програмери постављају језик *C* у сам врх листе програмских језика:

- ЈЕДНОСТАВНОСТ – у програмима на језику *C* заступљен је релативно мали број наредби, али је могуће коришћење стандардних библиотека, које нуде готова решења за велики број основних програмских задатака. Ове библиотеке иду у пакету развојног окружења. Важи да се на једноставан начин могу написати веома сложени *C* програми.
- ФЛЕКСИБИЛНОСТ – језик *C* се користи за развој системских и апликативних програма, програмских алата, програма за микроконтролере и микропроцесоре, као и за развој других програмских језика (*C++, C#, Java*).
- ПРЕНОСИВОСТ – једном написан програм на језику *C* преносив је, од једног до другог процесора, као и од једног до другог радног окружења.
- МОДУЛАРНОСТ – програм на језику *C* може се писати по логичким целинама (модулима) и свака од тих целина може бити искоришћена и код других *C* програма, ако је то потребно.

Из свих наведених разлога програмски језик *C* је и данас популаран (постоји велики број преводиоца за језик *C,* под разним оперативним системима) и изучава се на основном курсу из програмирања у овој школи, из предмета Основи програмирања.

Учење програмског језика значи много више од једноставног разумевања његове синтаксе. Синтаксу језика *C* чини прописани скуп правила о томе шта је дозвољено а шта не у програмима написаним на овом језику, и то су правила која се могу научити из књига. Значајнији део учења програмског језика је учење неписаних правила, као и логике употребе синтаксе језика, што се може савладати једино кроз праксу (након написаног извесног броја програма који решавају најразличитије програмске задатке).

Већина компоненти програма на језику *C* може се употребити на више начина, неки начини писања програма мање су разумљиви, неки више. Због тога се може чути да је језик *C* тежак за програмере почетнике. Да не би био тежи за учење од било ког другог програмског језика, у првим примерима за вежбу програми су написани на најједноставнији могући начин, док у каснијим примерима има комбиновања појединих компоненти, као и скраћивања записа.

Са овим треба ићи до границе релативно разумљивог текста програма. У фази учења програме треба писати једноставно, а касније су дозвољени сложени оператори и сложене наредбе.

Сваки пример у овом приручнику има:

1. Изворни кôд (текст програма). Свака линија изворног кôда програма има на почетку свој редни број који није део кôда, ту је само ради лакше анализе програма.
2. Анализу програма (по линијама изворног кôда).
3. Слику са излазом програма (у питању су конзолне апликације) по пуштеном програму у рад, све до краја његовог извршавања.
4. Неке практичне савете и евентуалне грешке сличних програма.

У Београду, 2019. године Аутори

Садржај

Вежба 1.

Основне компоненте програма на језику *C*

Главна функција

Стандардна функција форматираног излаза

Контролне секвенце у форматираном излазу

Примери

Изворни кôд програма **primer1_1**

```
1:    /*primer1_1.c*/
2:    /*Prikaz na ekranu jedne linije fiksnog teksta*/
3:
4:    #include  <stdio.h>
5:
6:    main( )
7:    {
8:          printf("\nPozdrav\n\n");
9:    }
```

Анализа програма **primer1_1**

У првом примеру следи анализа сваке линије текста програма. У каснијим примерима биће анализиране само линије са новим компонентама.

Текст у линијама 1. и 2. почиње симболима */** и завршава се симболима ***/**. Текст између поменутих симбола је коментар. Програмери додају коментаре тексту програма да би их учинили читљивијим, себи и другима. Преводилац за језик *C* игнорише коментаре, тако да они не утичу на извршавање програма. Коментари у програмима на језику *C* могу бити написани:

- у посебној линији */*linijski komentar*/*
- у више линија */*blokovski komentar*/*
- у продужетку наредбе или било где унутар наредбе (биће примера).

Коментар унутар коментара није дозвољен: */*/*pogresno napisan komentar*/*/*
У свим примерима даље коментар ће бити коришћен за опис програма.

Линије 3. и 5. су празне и користе се да би текст програма био читљивији. Празне линије користе се за одвајање појединих програмских целина. Преводилац их игнорише, као и коментаре. Поред празних линија користе се и празни знакови, за одвајање појединих делова наредбе.

У линији 4. написана је претпроцесорска директива *#include <stdio.h>*, која укључује у програм стандардну библиотеку *stdio.h*, са директоријума *include*, који је у пакету сваког развојног окружења за језик *C*. Ова библиотека садржи функције улаза/излаза (за сада ће то бити тастатура/екран).

Линија 6. је прва линија дефиниције главне функције *main,* која је неопходна компонента сваког програма на језику C (из ове функције се посредно или непосредно позивају све остале функције програма). Да се ради о функцији види се по малим заградама ( ) написаним после имена функције *main*. Функција је компонента програма која самостално обавља одређени задатак, у оквиру задатка целог програма, при томе опционо користи податке од остатка програма и опционо враћа резултат остатку програма. За извршавање главне функције *main* није потребно писати позив (што је случај

код свих осталих функција), јер ову функцију позива оперативни систем, по покретању програма.

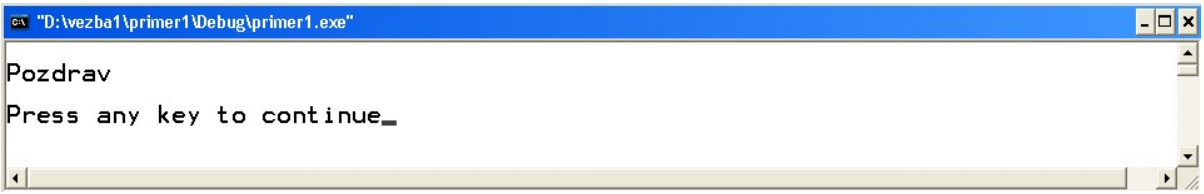
Тело главне функције (као и свих осталих) садржи наредбе по којима ради та функција и пише се између великих заграда, отвореном заградом { у линији 7. почиње тело главне функције, а затвореном } у линији 9. се завршава.

Једина наредба програма (позив стандардне функције *printf()*) смештена између великих заграда, у линији 8. *printf()* је функција стандардне библиотеке *stdio.h*. И за најједноставнији програм на језику *C*, какав је овај, неопходно је укључити функцију излаза на екран, у овом случају је то функција форматираног излаза *printf()*. Ова функција се користи за:

- приказ текста на екрану, онако како је текст написан у формату позива функције, између знака наводника (“ “), приказ нумеричких података на екрану у предвиђеном формату (форматиран приказ биће објашњен у каснијим примерима),
- разне помераје на екрану који се у формату означавају као управљачке секвенце (коса црта уназад и одговарајуће слово), највише се користе секвенца за прелаз у нови ред (*n*) и хоризонтални табулатор (*t*), а остале се могу видети у табели 2. у Прилогу,
- приказ знакова на екрану који имају специјална значења у формату (\\ приказује на екрану знак \, \” приказује на екрану знак ”, погледати табелу 2. у Прилогу).

Дакле, по позиву функције *printf()* у линији 8. следи:

1. прелаз у нови ред (*n*),
2. приказ на екрану задатог фиксног текста *Pozdrav*,
3. два пута прелаз у нови ред (*n**n*), да би порука која се исписује у *DOS* прозору, по извршеном програму (*Press any key to continue*), била одвојена од текста програма (слика 1.1).



Слика 1.1 Излаз програма **primer1_1**

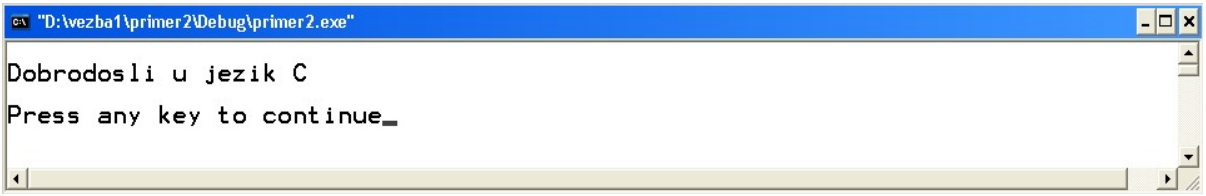
Изворни кôд програма **primer1_2**

```
1:      /*primer1_2.c*/
2:      /*Vise poziva funkcije printf(), prikaz na ekranu jedne linije teksta*/
3:
4:      #include <stdio.h>
5:
6:      main( )
7:      {
8:              printf("\nDobrodosli ");
9:              printf("u jezik C\n\n");
10:     }
```

Анализа програма **primer1_2**

У овом примеру је нешто модификован програм из претходног примера, сада су ту два позива стандардне функције *printf()* (у линијама 8. и 9.), али је излаз и даље текст приказан на екрану у једној линији. Ово је због тога што функција *printf()* нема прелаз у нови ред без секвенце *\n*.

Дакле, из линије 8. изворног кôда овог програма, прелази се у нови ред на екрану, приказује текст *Dobrodosli* и одваја празно место за један знак, онда се из линије 9., у истом реду, приказује текст *u jezik C*, а затим следи два пута прелаз у нови ред (слика 1.2).



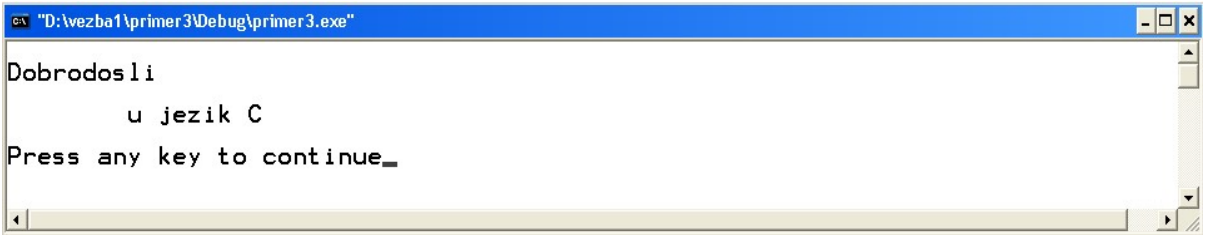
Слика 1.2 Излаз програма **primer1_2**

Изворни кôд програма **primer1_3**

```
1:      /*Primer1_3.c*/
2:      /*Jedan poziv finkcije printf(), prikaz na ekranu vise linija teksta*/
3:
4:      #include <stdio.h>
5:
6:      main( )
7:      {
8:              printf("\nDobrodosli\n\n\tu jezik C\n\n");
9:      }
```

Анализа програма **primer1_3**

За разлику од претходног примера, програм овог примера приказује на екрану текст у више линија, помоћу само једног позива функције *printf()* (из линије 8.). Ово омогућују управљачке секвенце, следећим редом. Функција прво изведе прелаз у нови ред и прикаже текст *Dobrodosli*, онда изведе два пута прелаз у нови ред као и хоризонтални померај (размак од једног хоризонталног табулатора), прикаже остатак текста *u jezik C* у посебној линији и понови два пута прелаз у нови ред (слика 1.3).



Слика 1.3 Излаз програма **primer1_3**

Изворни кôд програма **primer1_4**

```
1:      /*Primer1_4.c*/
2:      /*Prihvatanje imena od korisnika i prikaz na ekranu licnog pozdrava*/
3:
4:      #include <stdio.h>
5:      #define MAX 30
6:
7:      main( )
8:      {
9:              char ime[MAX];
10:
11:             printf("\nVase ime?\n\n");
12:             gets(ime);
13:
14:             printf("\nZdravo %s, dobrodosli u jezik C.\n\n", ime);
15:     }
```

Анализа програма **primer1_4**

У овом примеру је програм који не приказује на екрану само фиксни текст (што је било у предходним примерима) већ текст задат преко тастатуре.

Пример је ту, да би показао у овој, првој вежби, програм који има више од две линије изворног кôда. Није неопходно разумевање овог програма, јер ће касније бити објашњен сличан програм.

У линији 5. је нова претпроцесорска директива *#define*, која се користи за дефинисање симболичке константе. *#define MAX 30* је директива претпроцесору да свуда у тексту програма симбол *MAX* замени бројем *30*. У тексту програма се уместо саме вредности броја пише *MAX*, а евентуална измена захтева измену вредности само у директиви, а не и даље у програму (што је значајно код великог броја појављивања вредности истог броја у тексту програма).

У линији 9. написана је наредба декларације низа *ime* од *MAX (30)* карактера (променљивих типа *char*). Да се ради о низу види се по средњим заградама [ ]. Ово за сада треба прихватити као резервисање простора у меморији, који ће бити искоришћен за чување имена, задатог преко тастатуре.

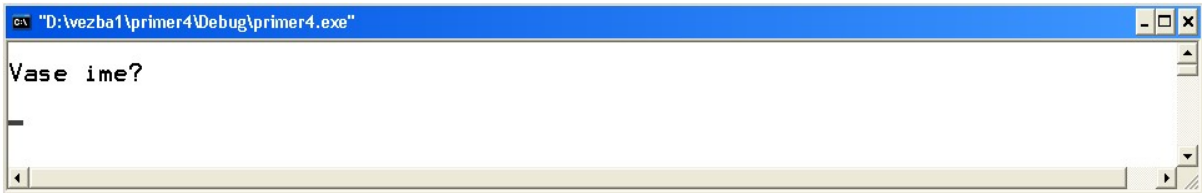
Из линије 11., помоћу позива функције *printf(),* програм пита за име (слика 1.4 а)).

Позив стандардне функције *gets() (*из библиотеке *stdio.h*) даље, у линији 12., значи да програм чека да се преко тастатуре зада име (низ знакова), да по завршеном уносу (који означава притиснут тастер *ENTER*) прихвата име и смешта га у резервисани простор у меморији, који се зове *ime*. У позиву функције пише се само име резервисаног простора у меморији, који ће чувати унети низ знакова.

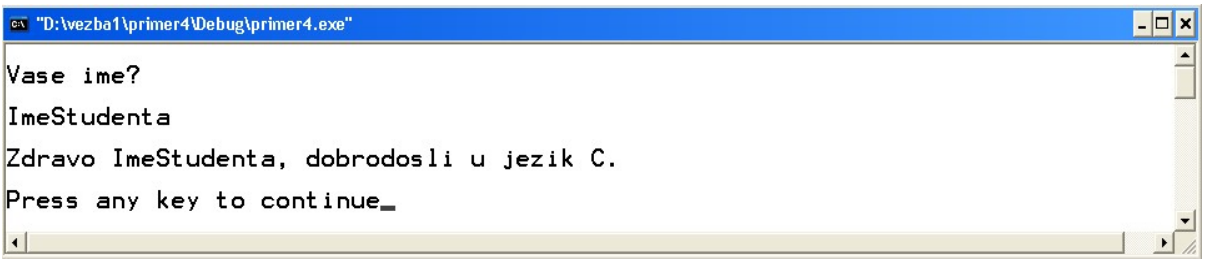
Из линије 14., помоћу позива функције *printf(),*приказује се на екрану фиксни део текста *Zdravo,* онда у одговарајућем формату (%s) садржај бафера *ime*, чије је име написано после зареза, на месту које заузима %s у формату, а онда и фиксни део текста *dobrodosli u jezik C.* (слика 1.4 б)).

Дакле, функција *printf()*:

- приказује на екрану текст, написан између наводника, све сем управљачких секвенци (\ и одговарајуће слово) и формата за поједине типове података (% и одговарајуће слово),
- ради по управљачким секвенцама (прелази у нови ред, хоризонтално помера текст,...),
- приказује на екрану садржаје променљивих чија су имена наведена после наводника, а на местима која заузимају њихови формати редом, унутар наводника.



Слика 1.4 а) Излаз програма **primer1_4** (први део)



Слика 1.4 б) Излаз програма **primer1_4** (други део)

Практични савети

- ✓ Иако су листинзи програма прве вежбе кратки, иза сцене програма се догађа много тога. Да би максимално искористили примере, сваки листинг треба унети прецизно, превести и пробати извршавање програма.

Евентуалне грешке

- ➢ Упозорења (*Warnings*) - са њима ће програм радити али су непожељна, јер ће вероватно условити рад програма на погрешан начин. За свако упозорење, преводилац приказује на екрану поруку која садржи редни број линије изворног кôда у којој је упозорење и даје опис упозорења.
- ➢ Синтаксне грешке (*Errors*) - су грешке које региструје преводилац, када нађе нешто у изворном кôду што не може да преведе. Онда исписује на екрану поруку у којој је линији (или до које је линије) грешка и даје опис грешке. За разумевање ових порука потребна су основна знања програмског језика *C*.

Задаци за припрему вежбе 1. у лабораторији

Задатак 1.

Исправити синтаксне грешке у изворном кôду следећег програма на језику *C*:

```
#include  stdio.h

        main( );
        {
                printf( \nResenja zadataka doneti na )
                printf( laboratorijske vezbe\n );
        }
```

Задатак 2.

Који је минимални број линија, потребан за изворни кôд програма из претходног задатка? Написати исти кôд помоћу минималног броја линија.

Задатак 3.

Написати програм на језику *C* који помоћу једног позива стандардне функције *printf()* исписује на екрану следећи текст:

```
Prva recenica.
        Druga recenica.
                Treca recenica.
```

Задаци за вежбу 1. у лабораторији

Задатак 1.

Вежбати унос, измене, памћење, превођење и извршавање програма на језику *C*, на примерима вежбе 1.

Задатак 2.

Направити намерно грешке у примерима вежбе 1. да би преводилац приказао поруке о њима. Прочитати ове поруке редом, од почетка листе, селектовати сваку од њих у листи излазног прозора и елиминисати их.

Задатак 3.

Пробати да ли ради програм на језику *C* написан овако:

а) *#include <stdio.h> main() {printf("\nPozdrav");}*

б) *#include<stdio.h>*
main() {printf("\nPozdrav");}

Задатак 4.

Написати програм на језику *C* који:

а) помоћу једног позива стандардне функције *printf()* приказује на екрану текст по избору у више линија,
б) помоћу више позива стандардне функције *printf()* приказује на екрану текст по избору у једној линији,
в) садржи коментаре по избору у посебној линији и у продужетку наредбе.

Вежба 2.

Основни типови података у језику *C*

Основни типови променљивих

Декларација и иницијализација променљивих

Дефиниција симболичких константи

Контроле за форматирање улаза и излаза

Примери

Изворни кôд програма **primer2_1**

```
1:      /*primer2_1.c*/
2:      /*Prikaz na ekranu velicina osnovnih tipova podataka u bajtovima*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {    printf("\nchar\t\tzauzima\t%d B memorije", sizeof(char));
8:           printf("\nunsigned char\tzauzima\t%d B memorije",sizeof(unsigned char));
9:           printf("\nint\t\tzauzima\t%d B memorije", sizeof(int));
10:          printf("\nunsigned int\tzauzima\t%d B memorije", sizeof(unsigned int));
11:          printf("\nlong\t\tzauzima\t%d B memorije", sizeof(long));
12:          printf("\nunsigned long\tzauzima\t%d B memorije",sizeof(unsigned long));
13:          printf("\nfloat\t\tzauzima\t%d B memorije", sizeof(float));
14:          printf("\ndouble\t\tzauzima\t%d B memorije.\n\n",  sizeof(double));
15:     }
```

Анализа програма **primer2_1**

Овај програм приказује на екрану величине појединих основних типова у језику *C* (величине простора у оперативној меморији које се користе за чување следећих типова променљивих: *char, unsigned char, int, unsigned int, long, unsigned long, float* и *double)*.

У линији 7., помоћу првог позива функције *printf()*, програм на екрану:

- прелази у нови ред: \n,
- приказује име основног *C* типа: *char,*
- прави размак од два хоризонтална табулатора: *\t\t,*
- приказује текст, написан до знака *\t: zauzima,*
- прави размак од једног хоризонталног табулатора: *\t,*
- на месту на коме је *%d*, у децималном формату приказује резултат примене оператора *sizeof()* на типу *char*, величину у бајтовима меморијског простора који заузима основни тип *char*,
- приказује текст до краја формата*: B memorije*.

За тип *char* програм приказује на екрану текст: *char zauzima 1 B memorije* (слика 2.1).

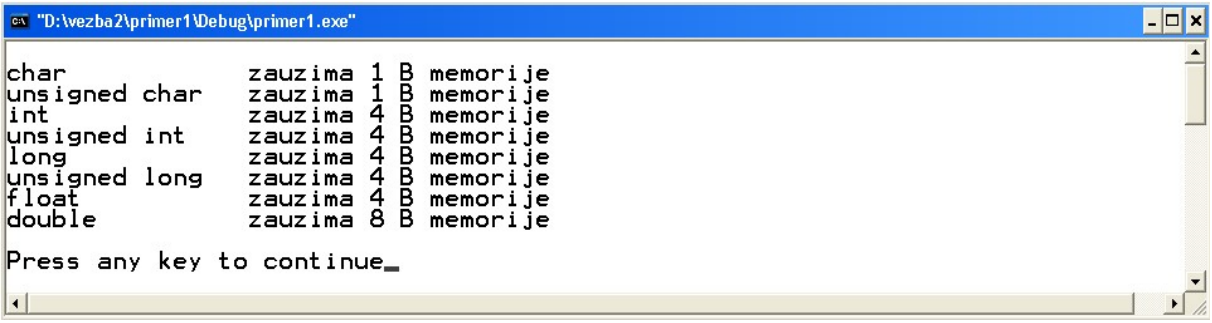
Позиви функције *printf()*, од линије 8. до линије 14. приказују на екрану исто за још неке типове променљивих у језику *C*.

Излаз програма као на слици 2.1 добија се за пуштање програма на 32-битном рачунару (на 16-битном рачунару биће различита величина у бајтовима за типове *int* (2 а не 4) и *unsigned int* (2 а не 4).

На истој слици може се видети да исти број бајтова заузимају: *char* и *unsigned char, int* и *unsigned int, long* и *unsigned long.*

Тип променљиве без префикса *unsigned* користи се за чување одређеног опсега позитивних и негативних вредности бројева, док се *unsigned* тип користи само за чување позитивних бројева (опсег је померен и само позитиван).

Табела 3. у Прилогу има за сваки основни тип променљиве у језику *C*: ознаку, величину у бајтовима и опсег вредности за који је тип употребљив.



Слика 2.1 Излаз програма **primer2_1**

Изворни кôд програма **primer2_2**

```
1:      /*primer2_2.c*/
2:      /*Prikaz na ekranu zadatog celog broja u raznim oblicima*/
3:
4:      #include <stdio.h>
5:
6:      main( )
7:      {       int x;
8:
9:              printf("\nUnesite ceo broj:\t");
10:             scanf("%d", &x);
11:
12:             printf("\nDecimalni oblik:\t%d",x);
13:             printf("\nOktalni oblik:\t\t%o",x);
14:             printf("\nHeksadecimalni oblik:\t%x (%X)\n\n",x,x);
15:     }
```

Анализа програма **primer2_2**

Програм користи једне податке да би одрадио свој задатак, а даје друге резултујуће. За сваки од ових података резервише одговарајући простор у оперативној меморији, променљиву одговарајућег типа.

У овом примеру програм приказује на екрану задати цео број, у разним облицима. Има само један улазни податак, и за овај податак декларише у линији 7 целобројну променљиву типа *int*., а резултате приказује на екрану. Дакле, нема декларације променљивих за резултате. Декларација променљиве садржи тип (који резервише одговарајући број бајтова у оперативној меморији) и име (потребно за приступ променљивој у програму). Декларација променљиве има први ниво провере исправности, јер се одмах зна опсег дозвољених вредности за податак који ће чувати та променљива. Дакле,

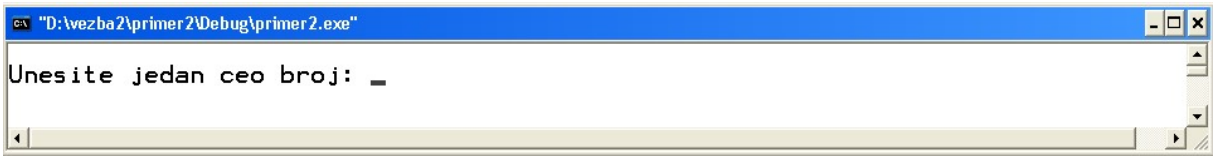
променљива *x* типа *int* спремна је да прихвати један цео број у опсегу 4 бајта (у овом случају).

Даље, из линије 9. приказује се на екрану промпт програма, упутство кориснику какву акцију треба да предузме преко тастатуре (слика 2.2 а)).

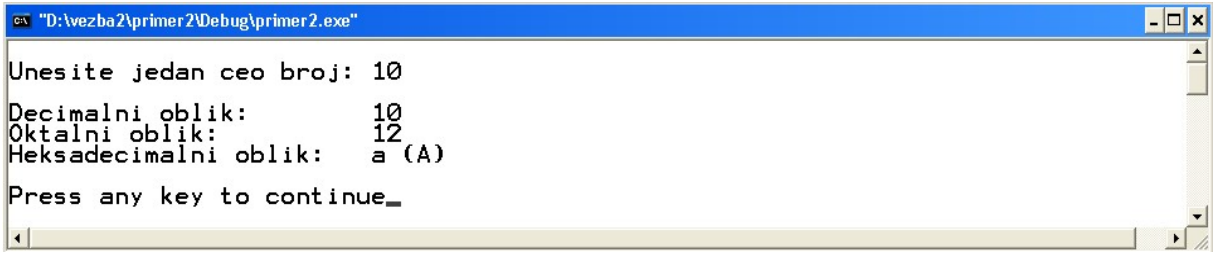
Из линије 10. програм прихвата са тастатуре један цео број у задатом децималном формату *(%d)* и смешта га од адресе резервисане у оперативној меморији, за променљиву (*&x).*

Из линија 12., 13. и 14. програм приказује на екрану садржај исте променљиве (*x)* у децималном *(%d)*, октално *(%o)* и хексадецималном облику, са малим *(%x)* и великим словима *(%X)* хексадецималног система (слика 2.2 б)).

Погледати контроле за форматирање излаза и улаза у програмима на језику *C*, у табели 4. и табели 5. у Прилогу.



Слика 2.2 а) Излаз програма **primer2_2** (први део)



Слика 2.2 б) Излаз програма **primer2_2** (други део)

Изворни кôд програма **primer2_3**

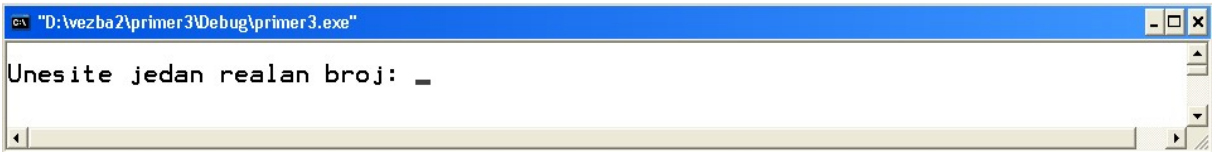
```
1:      /*primer2_3.c*/
2:      /*Prikaz na ekranu zadatog realnog broja u raznim oblicima*/
3:
4:      #include <stdio.h>
5:
6:      main( )
7:      {       double x;
8:
9:              printf("\nUnesite jedan realan broj: ");
10:             scanf("%lf", &x);
11:
12:             printf("\nOblik sa decimalnom tackom:\t%f",x);
13:             printf("\nOblik sa eksponentom:\t\t%e",x);
14:             printf("\nJednostavniji oblik:\t\t%g\n\n",x);
15:     }
```

Анализа програма **primer2_3**

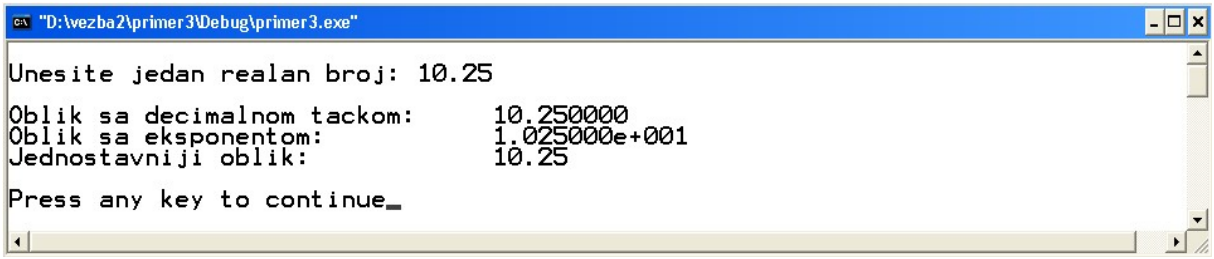
У свим примерима програми ће на почетку главне функције садржати неопходне наредбе декларације свих променљивих потребних програму (за улазне и излазне податке). У овом случају то је променљива *x* типа *double* (декларација написана у линији 7.).

После промпта, из линије 9. (слика 2.3 а)), програм прихвата са тастатуре реалну вредност задату са двоструком тачношћу *("%lf)* и смешта је од адресе резервисане за променљиву *x* (*&x).*

Из линија 12., 13. и 14. програм приказује на екрану садржај исте променљиве (*x)* са децималном тачком*(%f)*, са експонентом *(%e)* и у облику *(%g)* који преводилац прихвати као најједноставнији (слика 2.3 б)).



Слика 2.3 а) Излаз програма **primer2_3** (први део)



Слика 2.3 б) Излаз програма **primer2_3 (**други део)

Изворни кôд програма **primer2_4**

```
1:      /*primer2_4*/
2:      /*Razliciti nacini prikaza na ekranu zadatog broja*/
3:      #include  <stdio.h>
4:      main( )
5:      {       const int x = 999;
6:              const double y = 1.2345;
7:
8:              printf("\nCeo broj:\n\n");
9:              printf("%10d\n", x);
10:             printf("%-10d\n", x);
11:
12:             printf("\nRealan broj:\n\n");
13:             printf("%10f\n", y);
14:             printf("%-10f\n", y);
15:             printf("%10.2f\n", y);
16:             printf("%.2f\n", y);
17      }
```

Анализа програма **primer2_4**

Овај програм приказује на екрану (у одговарајућим облицима) вредности константи дефинисаних кључном речи *const* (линије 5. и 6.). На овај начин је могуће дефинисање променљиве било ког типа (резервисање одговарајућег простора у меморији) чији је садржај закључан (измене садржаја ове променљиве нису дозвољене даље у програму.)

Када се стандардна функција *printf()* користи за приказ нумеричких података и наведе се у формату само знак % и одговарајуће слово, онда поменута функција приказује на екрану:

- целобројну вредност (типа *char*, *int* или *long*), на минималној ширини потребној за њен приказ (ако број има 3 цифре за тај број одваја поље ширине 3),
- реалну вредност (тип *float* или *double*), на минималној ширини потребној за њен приказ и са подразумеваним бројем цифара иза децималне тачке (6 цифара).

У општем случају формат функције *printf()* може имати и опције између знака % и слова. У овом примеру су неке од тих опција које се највише користе.

Из линија 9. и 10. програм приказује (слика 2.4) целобројну вредност 999:

- из линије 9. одваја минималну ширину поља (10) за приказ броја и равна број на подразумевани начин, уз десну ивицу поља (%10d) (празна места испред броја могу бити замењена нулама писањем у формату нуле испред минималне ширине поља (%010d)),
- из линије 10. одваја минималну ширину поља (10) за приказ броја и равна број уз леву ивицу поља (%-10d).

Из линија 13., 14., 15. и 16. програм приказује (слика 2.4) реалну вредност 1.2345 на следеће начине:

- из линије 13. одваја минималну ширину поља (10) за приказ броја, равна број уз десну ивицу (подразумевани начин), приказује подразумевани број цифара (6) иза децималне тачке (%10f),
- из линије 14. одваја минималну ширину поља (10) за приказ броја, равна број уз леву ивицу поља (због знака -), приказује подразумевани број цифара (6) иза децималне тачке (%-10f),
- из линије 15. одваја минималну ширину поља (10) за приказ броја, равна број уз десну ивицу (подразумевани начин), приказује две цифре иза децималне тачке (%10.2f),
- из линије 16. одваја минималну потребну ширину поља за приказ броја, приказује две цифре иза децималне тачке (%.2f).

Слика 2.4 Излаз програма **primer2_4**

Изворни кôд програма **primer2_5**

```
1:      /*primer2_5.c*/
2:      /*Zamena vrednosti dva zadata cela broja*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int a, b, temp;
8:
9:              printf("\nUnesite dva cela broja: ");
10:             scanf("%d%d", &a, &b);
11:
12:             printf("\nPre zamene:\ta=%d, b=%d", a, b);
13:
14:             temp = a;               /*temp - pomocna promenljiva*/
15:             a = b;                  /*koja se koristi pri razmeni*/
16:             b = temp;               /*vrednosti a i b*/
17:
18:             printf("\nPosle zamene:\ta=%d, b=%d\n\n", a, b);
19:     }
```

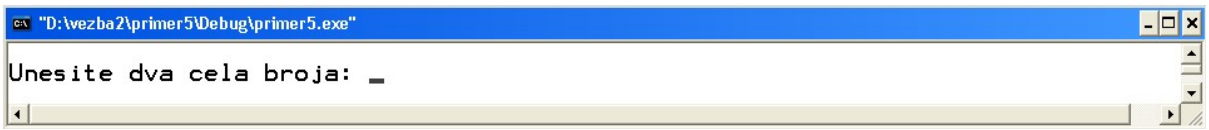
Анализа програма **primer2_5**

Програм у овом примеру замењује вредности два задата цела броја.

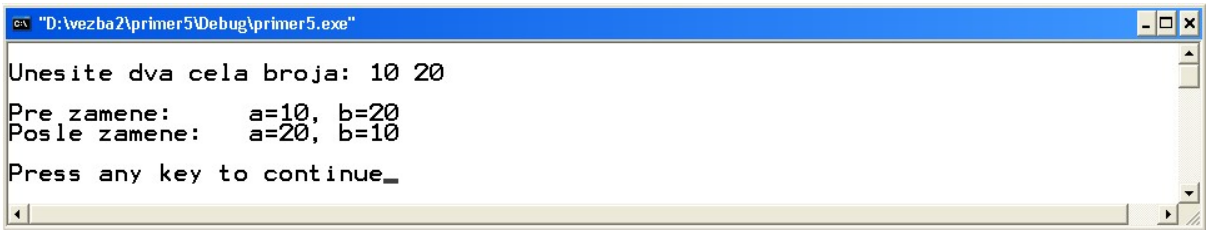
Да би била извршена замена један број мора бити привремено сачуван да на његово место дође други, а онда на место другог сачувана вредност првог. У линији 7. декларисане су променљиве *a* и *b* за два задата цела броја, као и помоћна целобројна променљива *(temp)* за потребе замене. Све променљиве су декларисане у једној линији, јер су истог типа.

После промпта, генерисаног из линије 9. (слика 2.5 а)), програм помоћу једног позива функције *scanf()* прихвата два цела броја, задата у децималном облику са празним знаком између (*Space, Tab* или *Enter*) и чува их од адреса резервисаних за променљиве *a* и *b* (*&a* и *&b).* Подаци се при уносу могу одвајати и неким знаком различитим од празног знака. Ако би програм чекао да корисник уноси два цела броја, одвојена знаком тачка-зарез, онда би позив функције за читање (у линији 10.) изгледао овако: *scanf("%d;%d", &a, &b);*

После приказа на екрану вредности задатих бројева (из линије 12.) и замене (од 14. до 16. линије), програм приказује на екрану (линија 18.) (слика 2.5 б)) нове вредности бројева. Замена садржи следеће операције: у променљиву *temp* смешта се вредност првог задатог броја *a*, променљива *a* добија вредност другог задатог броја *b*, а променљива *b* вредност првог задатог броја *a*, сачувану у променљивој *temp.*



Слика 2.5 а) Излаз програма **primer2_5** (први део)



Слика 2.5 б) Излаз програма **primer2_5** (други део)

Изворни кôд програма **primer2_6**

```
1:      /*primer2_6.c*/
2:      /*Izracunavanje  obima i povrsine*/
3:      /*kruga zadatog poluprecnika*/
4:
5:      #include  <stdio.h>
6:      #define  PI  3.14159
7:
8:      main( )
9:      {       int r, obim, povrs;
10:
11:             printf("\nUnesite  vrednost poluprecnika (ceo broj: 0..10):  ");
12:             scanf("%d", &r);
13:
14:             obim = (int) (2 * r * PI);      /*celobrojni deo vrednosti obima*/
15:             povrs = (int) (r * r * PI);     /*celobrojni deo vrednosti povrsine*/
16:
17:             printf("\nObim kruga je %d", obim);
18:             printf("\nPovrsina kruga je %d\n\n", povrs);
19:     }
```

Анализа програма **primer2_6**

У овом програму који израчунава обим и површину задатог круга, потребна је вредност константе *PI.*.

У линији 6. помоћу претпроцесорске директиве *#define* дефинисан је симбол *PI,* који ће у тексту програма бити коришћен уместо вредности ове константе *3.14159.* Коришћење симболичке константе олакшава писање и измене програма, јер онда у његовом тексту треба писати само симбол, а у случају измене мењати само вредност константе у линији где је написана претпроцесорска директива. Нема резервисања простора у меморији, за разлику од константе дефинисане кључном речи *const*.

У линији 9. програм декларише променљиве потребне за задату вредност полупречника (*r*), и тражених вредности (*obim* и *povrs*).

Када прихвати вредност полупречника, у линијама 11. и 12. (слика 2.6 а)) програм рачуна обим (у линији 14.) и површину (у линији 15.). У линији број 14. по први пут се појављује оператор конверзије типа *(int),* који тип израза испред кога је написан *(2 * r * PI)* конвертује у целобројни тип *int* у овом случају. Вредност овог израза се у истој линији додељује целобројној променљивој

obim, тако да написан је оператор конверзије у исти целобројни тип. Оператор конверзије је написан и код израза за површину, у линији 15. И без овог оператора био би одсечен целобројни део вредности израза, због доделе резултујућој целобројној променљивој, али би било упозорења.

Из линија 17. и 18. програм приказује на екрану резултате (слика 2.6 б)).

Слика 2.6 а) Излаз програма **primer2_6** (први део)

Слика 2.6 б) Излаз програма **primer2_6** (други део)

Изворни кôд програма **primer2_7**

```
1:      /*primer2_7.c*/
2:      /*Izracunavanje  Celsius-ovog ekvivalenta zadatom Fahrenheit-u*/
3:      /* Celsius = (5/9) * ( Fahrenheit-32) */
4:
5:      #include  <stdio.h>
6:
7:      main( )
8:      {       int temp;
9:              double fahr, celsius;
10:
11:             printf("\nUnesite T u stepenima F (ceo broj: 0..300):  ");
12:             scanf("%d",  &temp);
13:
14:             fahr = (double)temp;                    /*konverzija celobrojnog u realni tip*/
15:             celsius = (5.0 / 9.0) * (fahr - 32.0);/*da bi svi operandi bili istog tipa*/
16:             printf("\n%d F    %.2f C\n\n", temp, celsius);
17:     }
```

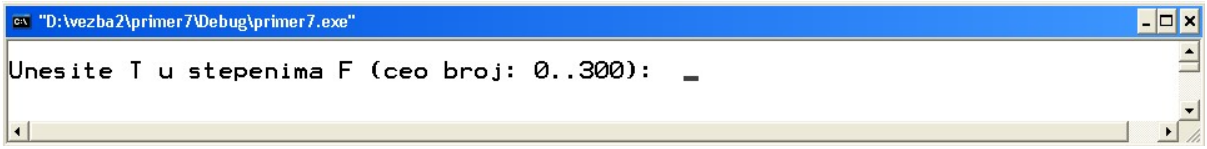
Анализа програма **primer2_7**

Овај програм израчунава Celsiusov еквивалент задатој температури у Fahrenheitu.

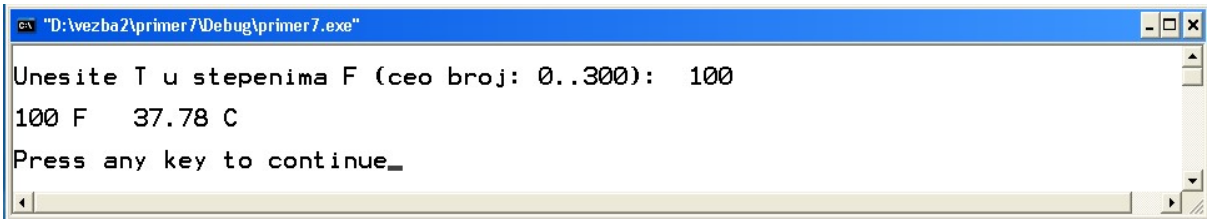
Декларација свих потребних променљивих изведена је помоћу две наредбе: једне за декларацију помоћне целобројне променљиве *temp* (линија 8.) и друге за декларацију реалних променљивих: за задату температуру у *Fahrenheitu (fahr*) и за исту температуру у *Celsiusu* (*celsius*), у линији 9.

Након прихваћене вредности температуре, задате преко тастатуре у линијама 11. и 12. (слика 2.7 а)), програм врши конверзију задате целобројне вредности у реалну вредност температуре (типа *double)* и додељује ту вредност одговарајућој променљивој *fahr* (линија 14.). У линији 15. програм израчунава *Celsiusov* еквивалент задате температуре и додељује је резултујућој променљивој *(celsius).* Конверзија типа (у линији 14.) изведена је да би сви операнди у изразу, написаном у линији 15., били истог типа (што је једно од неписаних правила). Из истог разлога су и вредности константи написане као реалне (5.0, 9.0 и 32.0).

Из линије 16. програм приказује на екрану (слика 2.7 b)) целобројну вредност задате температуре *(temp*) у децималном облику *(%d*) и реалну вредност еквивалента у *Celsiusu* (*celsius*), са две децимале *(%.2f*).



Слика 2.7 а) Излаз програма **primer2_7** (први део)



Слика 2.7 б) Излаз програма **primer2_7** (други део)

Практични савети

- ✓ Променљива у програму треба да има име које добро описује садржај те променљиве. Ово доприноси доброј документованости програма.
- ✓ За имена променљивих користе се мала слова, док су за имена константи резервисана велика слова.
- ✓ Ако се за податак декларише променљива са непотребно великим опсегом, нема грешке, али се неефикасно користи меморијски простор.

Евентуалне грешке

- ➢ *C* преводилац прави разлику мала/велика слова, тако да три идентификатора *temp, Temp* и *TEMP* прихвата као три различита имена.
- ➢ Име променљиве (као и било који други *C* идентификатор) не може бити службена реч језика *C* (списак ових речи видети у табели 1. у Прилогу).
- ➢ Ако се за податак декларише променљива са недовољно великим опсегом, променљива чува вредност -1, уместо предвиђене вредности.
- ➢ Ако се за негативан број декларише *unsigned* тип, онда променљива чува максималну вредност свог опсега, уместо предвиђене вредности.
- ➢ Ако се за реалан број декларише целобројни тип променљиве онда она чува само одсечени целобројни део, уместо предвиђене вредности.
- ➢ Константа дефинисана кључном речи *#define* дефинише симбол за претраживање и замену, без резервисања меморијске локације.

Задаци за припрему вежбе 2. у лабораторији

Задатак 1.

Написати програм на језику *C* који вредност целог броја, унетог преко тастатуре, у било ком од децималног, октално или хексадецималног облика, приказује на екрану у свим поменутим облицима и исписује одговарајући текст:

Decimalni oblik unetog broja je:
Oktalni oblik unetog broja je:
Heksadecimalni oblik unetog broja je:

Задатак 2.

Написати програм на језику *C* који вредност реалног броја, унетог преко тастатуре, са децималном тачком, приказује на екрану у разним облицима:

а) са три децимале (предвиђена минимална ширина поља за број је 10 места),
б) са експонентом,
в) у најједноставнијем облику.

Задатак 3.

Како треба написати позив стандардне *C* функције *scanf()* да би се за унос три броја на следећи начин: 1:23:456 (бројеви одвојени знаком двотачка), променљивама *a, b* и *c* доделиле вредности 1, 23 и 456, респективно.

Задатак 4.

Које вредности исписује на екрану следећи програм на језику *C:*

```
#include <stdio.h>
main( )
{
        int x, y, z;
        scanf("%2d%3d%2d", &x, &y, &z);
        printf("%d %d %d", x, y, z);
}
```

за задата три броја на следећи начин: 123 456 789?

Задатак 5.

За вредност целобројне променљиве *a* = 555 написати позиве стандардне *C* функције *printf()*, који на екрану приказују следеће (предвиђена ширина поља за број је 10):

а) / 555/
б) /555 /

Задаци за вежбу 2. у лабораторији

Задатак 1.

Написати програм на језику *C* по програму *primer2_1.c*, који приказује на екрану величине у бајтовима још неких основних *C* типова променљивих.

Задатак 2.

Написати програм на језику *C* који вредност целог броја, задатог преко тастатуре, приказује на екрану на следећи начин:

а) поравнан уз леву маргину, предвиђена минимална ширина поља за број је 10,
б) поравнан уз десну маргину, предвиђена минимална ширина поља за број је 10, са нулама испред цифара,
в) поравнан уз десну маргину, предвиђена минимална ширина поља за број је 10, без нула испред цифара.

Задатак 3.

Написати програм на језику *C* који вредност реалног броја, задатог преко тастатуре, приказује на екрану на следећи начин:

а) само целобројни део,
б) са 4 децимале и предвиђеном минималном ширином поља 10, поравнан уз леву маргину,
в) са 2 децимале и предвиђеном минималном ширином поља 10, поравнан уз десну маргину.

Задатак 4.

Написати програм на језику *C* који помоћу једног позива стандардне функције *printf()* исписује слово A на следеће начине: сам знак A, као и *ASCII* вредност слова A у децималном, окталном и хексадецималном облику.

Задатак 5.

Написати програм на језику *C* који помоћу једног позива стандардне *C* функције *scanf()* прихвата са тастатуре два цела броја, израчунава и приказује на екрану њихову аритметичку средину (целобројну и реалну вредност).

Вежба 3.

Оператори, изрази и наредба селекције у програмима на језику *C*

Аритметички, релацијски, логички оператори и оператори на нивоу бита

Контрола тока програма

Наредба селекције: *if*

Чланови наредбе селекције *if*: *if..else* и *else*

Примери

Изворни кôд програма **primer3_1**

```
1:      /*primer3_1.c*/
2:      /*Primena unarnog aritmetickog operatora dekrement*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int a = 3, b = 3;
8:
9:              printf("\n a     b \n");
10:             printf("\n %d  %d", a--, --b);          /*a se koristi i dekrementira*/
11:             printf("\n %d  %d", a--, --b);          /*b se dekrementira i koristi*/
12:             printf("\n %d  %d \n\n", a--, --b);     /*kod sva tri poziva funkcije printf()*/
13:     }
```

Анализа програма **primer3_1**

У табели 6. у Прилогу дат је преглед свих оператора у језику *C*. Један од њих је унарни аритметички оператор декремент. Оператор декремент (--) умањује вредност променљиве над којом је примењен за 1.

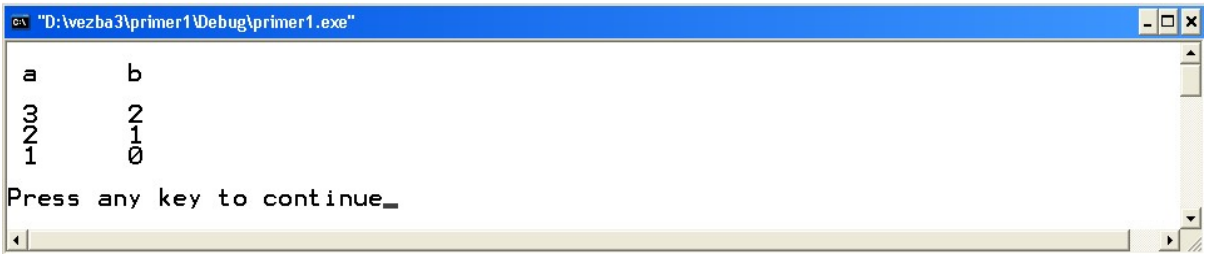
У линији 7. су променљиве *a* и *b* декларисане и иницијализоване (додељене су им почетне вредности). Иницијализација унутар декларација може бити изведена само на приказан начин. Израз *int a = b = 3;* условио би грешку.

Наредбе од линије 10. до линије 12. приказују два начина примене оператора декремент (слика 3.1).

Први начин је примењен код променљиве *a.* Вредност променљиве прво се прикаже на екрану, а онда се декрементира (написан је постдекремент *a*-).

Други начин је примењен код променљиве *b*. Вредност променљиве се прво декрементира, а онда се прикаже на екрану (написан је предекремент *–b*).

Исто важи и за унарни аритметички оператор инкремент, који увећава вредност операнда за 1 (могући су постинкремент (*a*++) и преинкремент (++*b*)).



Слика 3.1 Излаз програма **primer3_1**

Изворни кôд програма **primer3_2**

```
1:    /*primer3_2.c*/
2:    /*Izracunavanje brojeva sati, minuta i sekundi u zadatom broju sekundi*/
3:    /* (primena aritmetickog operatora ostatak deljenja) */
4:
5:    #include  <stdio.h>
6:    #define SEKUNDI_U_MINUTI  60
7:    #define SEKUNDI_U_SATU  3600
8:
9:    main( )
10:   {       unsigned int sekunde, minute, sati, sekunde_ostatak, minute_ostatak;
11:
12:           printf("\nUnesite broj sekundi (ceo broj < 65000):  ");
13:           scanf("%d", &sekunde);
14:
15:           sati = sekunde / SEKUNDI_U_SATU;
16:           minute = sekunde / SEKUNDI_U_MINUTI;
17:           minute_ostatak = minute % SEKUNDI_U_MINUTI;
18:           sekunde_ostatak = sekunde % SEKUNDI_U_MINUTI;
19:
20:           printf("\n%u sekundi je isto sto i %u sati,",sekunde, sati);
21:           printf("%u minuta i %u sekundi\n\n",minute_ostatak, sekunde_ostatak);
22:   }
```

Анализа програма **primer3_2**

Највише од свих оператора користе се бинарни аритметички оператори. Овде ће бити речи само о два оваква оператора, о оператору дељења (*/*) и оператору остатак дељења (*%*).

Оператор дељења даје целобројни део количника када су оба операнда целобројна. Следе примери:

Израз 7/4 има целобројну вредност количника, а то је 1. Сваки од израза 7.0/4, 7/4.0 и 7.0/4.0 има реалну вредност количника, то је 1.750000, због тога што је бар један од операнада реалан.

Код променљивих то би изгледало овако: за декларацију *int a = 7, b = 4;* израз *a/b* има целобројну вредност количника, јер су оба операнда целобројна, док сваки од израза (float)*a/b*, *a/(float)b* и (float)*a/(float)b* има реалну вредност истог количника (извршена је конверзија типа бар код једног од операнада).

Израз *7%4* даје остатак дељења броја 7 бројем 4, а то је 3.

У овом примеру прво су дефинисане симболичке константе, ради лакшег праћења вредности у програму (линије 6. и 7.), а онда су у главној функцији декларисане потребне променљиве.

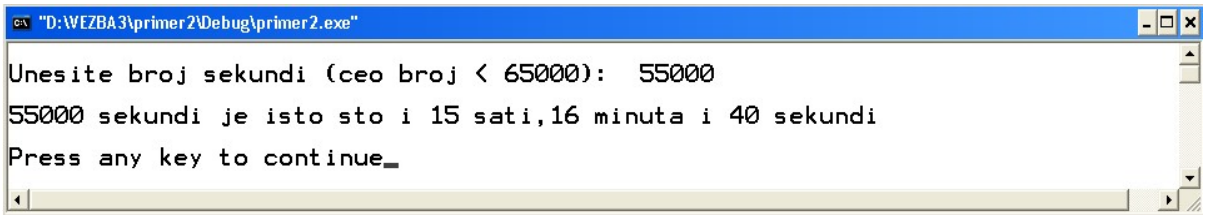
За задати укупан број секунди програм израчунава број сати, преосталих минута и преосталих секунди. После приказаног промпта (слика 3.2 а)) и прихватања укупног броја секунди са тастатуре (*sekunde*) следи рачунање:

- укупан број сати (*sati*) добија се дељењем задатог броја секунди са 3600 (линија 15.),

- укупан број минута (*minute*) добија се дељењем задатог броја секунди са 60 (линија 16.),
- преостали број минута (*minute_ostatak*) добија се као остатак дељења укупног броја минута са 60 (линија 17.),
- а преостали број секунди (*sekunde_ostatak*) добија се као остатак дељења задатог броја секунди са 60 (линија 18.).

и приказ на екрану резултата (слика 3.2 б)).

Слика 3.2 а) Излаз програма **primer3_2** (први део)



Слика 3.2 б) Излаз програма **primer3_2** (други део)

Изворни кôд програма **primer3_3**

```
1:      /*primer3_3.c*/
2:      /*Prikaz na ekranu rezultata izraza sa relacijskim i logickim  operatorim*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int x, a = 5,  b = 6,  c = 5,  d = 1;
8:
9:              x = a<b  ||  a<c  &&  c<d;
10:             printf("\nx = a<b || a<c && c<d");          /*&& ima veci prioritet od ||*/
11:             printf("\nBez zagrada izraz ima vrednost\t%d\n\n",x);
12:
13:             x = (a<b || a<c) && c<d;
14:             printf("\nx = (a<b || a<c) && c<d");         /*zagrade menjaju prioritet*/
15:             printf("\nSa zagradama izraz ima vrednost\t%d\n\n",x);
16:             }
```

Анализа програма **primer3_3**

Приоритет оператора у језику *C* приказан је у табели 6. у Прилогу. За извршавање операција другим редоследом, неопходно је коришћење заграда. На пример, резултати израза *a+b/2* и *(a+b)/2* нису исти.

Примена заграда код измене редоследа извршавања оператора показана је у овом примеру код израза са релацијским (<) и логичким операторима (|| и &&).

Релацијски оператори су следећи: == (еквиваленције), *!=* (негације еквиваленције), < (мање), <= (мање или еквивалентно), > (веће), >= (веће или еквивалентно). Резултат сваког релацијског израза (на пример *a* < *b*) је логичког типа и може имати једну од две вредности: 1 (*true*), ако је израз тачан или 0 (*false*) ако израз није тачан.

Сваки логички оператор примењује се над два логичка израза и резултат је логичког типа. Код оператора логички И (&&) резултат је 1 (*true*) само ако су оба израза тачна, код оператора логички ИЛИ (||) резултат је 1 (*true*) ако је бар један од израза тачан, код оператора логички НЕ (!) резултат је 1 (*true*) само ако је израз нетачан.

Сада следи сама анализа примера. После декларације променљивих програм приказује на екрану један логички израз (линија 10.), као и вредност истог израза (линија 11.), а онда приказује на екрану (слика 3.3) други логички израз, добијен од првог тако што су додате заграде (линије 14. и 15.).

Ево вредности ова два израза, за декларацију из линије 7. (*int x, a=5, b=6, c = 5, d = 1;):*

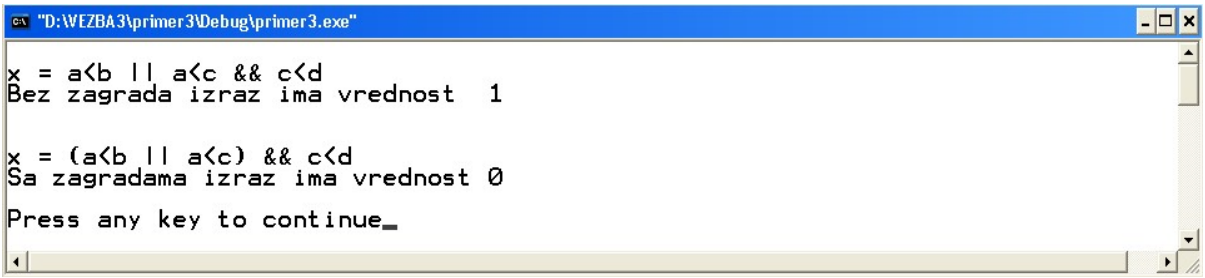
Вредност израза из линије 9.

x	=	a<b	||	a<c	&&	c<d
x	=	5<6	||	5<5	&&	5<1
x	=	1	||	0	&&	0
x	=	1	||	0		
x	=	1				

Вредност израза из линије 13.

x	=	(a<b	||	a<c)	&&	c<d
x	=	(5<6	||	5<5)	&&	5<1
x	=	(1	||	0)	&&	0
x	=	1			&&	0
x	=	0				

У сваком изразу подвучен је оператор који је приоритетнији, заграде су измениле приоритет извршавања оператора у другом изразу и због тога он има различиту вредност од првог.



Слика 3.3 Излаз програма **primer3_3**

Изворни кôд програма **primer3_4**

```
1:    /*primer3_4.c*/
2:    /*Provera i izmena vrednosti odredenog bita u zadatom podatku*/
3:    /* (maskiranje i promena vrednosti na nivou bita) */
4:
5:    #include <stdio.h>
6:
7:    main( )
8:    {     unsigned short x, n;
9:
10:         printf("\nUnesite jedan pozitivan ceo broj u heksadecimalnom obliku");
11:         printf("\n(Opseg broja: do 0xffff):\t");
12:         scanf("%x",&x);
13:
14:         printf("\nUnesite redni broj bita koji se ispituje (od 1 do 16): ");
15:         scanf("%d",&n);
16:
17:         if(x & (1 << (n-1)))    /*ako bit sa rednim brojem n ima vrednost 1*/
18:               printf("\n%d. bit zadatog podatka ima vrednost 1\n",n);
19:         else
20:               printf("\n%d. bit zadatog podatka ima vrednost 0\n",n);
21:
22:         x ^= (1 << (n-1));      /*menja se vrednost bita sa rednim brojem n*/
23:
24:         printf("\nVrednost %d. bita zadatog podatka je promenjena\n",n);
25:         printf("\nNova vrednost zadatog podatka je: %#x\n\n",x);
26:   }
```

Анализа програма **primer3_4**

Програмски језик *C* има и операторе на нивоу бита. Сваки унарни оператор на нивоу бита (комплемент (~), померај улево (<<) и померај удесно (>>)) примењује се над сваким битом податка операнда редом, док се сваки бинарни оператор на нивоу бита (логички И на нивоу бита (*&*), логички ИЛИ на нивоу бита (|) и логички искључиво ИЛИ на нивоу бита (^)) примењује над одговарајућим битовима од операнада (над првим битом првог операнда и првим битом другог операнда, и тако редом).

У овом примеру се по први пут појављује и једна наредба за контролу тока програма, то је наредба *if*. До овог примера наредбе у програмима су се извршавале редом којим су написане. Наредба *if* омогућује да се донесе одлука у програму да ли ће се одређена наредба (или блок наредби) извршавати. Наредба *if* је наредба селекције, ако је испуњен њен услов (написан унутар заграда) извршава се наредба (или блок) која следи, у супротном наредба (или блок) се прескаче. Наредба *if* може имати и *else* члан, којим се дефинише наредба (или блок) која се извршава ако *if* услов није испуњен.

У овом примеру програм тражи (слика 3.4 а)) и прихвата са тастатуре један цео број (линије 10, 11., и 12.), а онда (слика 3.4 б)) и редни број бита који треба испитати у задатом броју (линије 14. и 15.).

Следи опис дела програма од линије 17. до линије 22. У задатку је обележавање позиције бита такво да бит најмање тежине има редни број 1.

Ако бит са редним бројем *n* задатог броја има вредност 1 (линија 17.) програм приказује на екрану једну поруку (линија 18.). За *n=2* израз *(1 << (n-1))* има вредност у бинарном облику: 00000000 00000010. На тај начин је направљена маска, која као резултат примене на задатом броју (у овом случају то је број: 0xaa) уз помоћ оператора логочки И на нивоу бита (&) нулира све остале битове сем *n*-тог (другог у овом случају):

```
        00000000 10101010    (0xaa)
&       00000000 00000010    (0x02)
        00000000 00000010    (0x02)
```

Резултат је број различит од 0, што значи да је услов *if* наредбе у линији 17. испуњен.

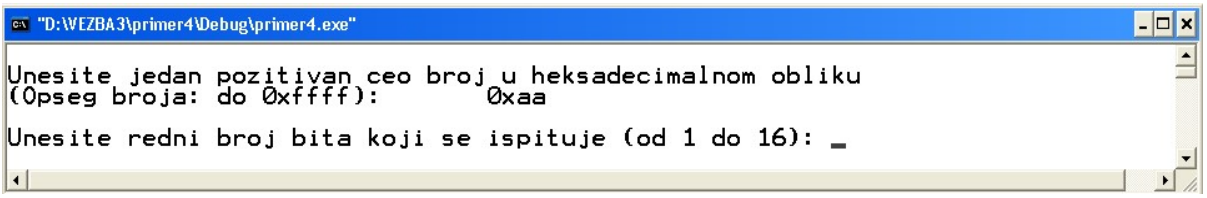
У супротном, ако бит са редним бројем *n* задатог броја нема вредност 1 (линија 19.), програм приказује на екрану другу поруку (линија 20.).

Помоћу наредбе у линији 22. програм мења вредност бита са редним бројем *n* у задатом броју. За *n=2* од израза *(1 << (n-1))* направљена је маска у бинарном облику: 00000000 00000010. Уз помоћ оператора логочки ИСКЉУЧИВО ИЛИ на нивоу бита (^) ова маска мења вредност *n*-тог бита (другог у овом случају):

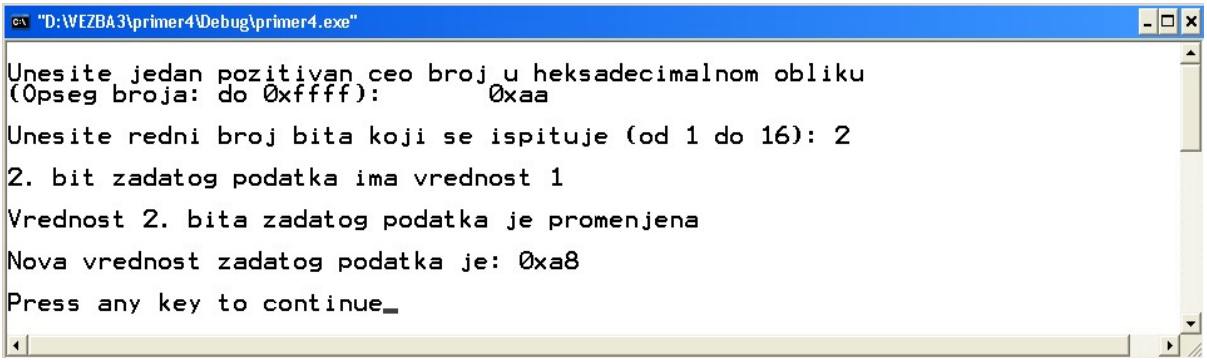
```
        00000000 10101010    (0xaa)
^       00000000 00000010    (0x02)
        00000000 10101000    (0xa8)
```

Измењена је само вредност другог бита задатог податка.
На слици 3.4 в) може се видети приказ резултата програма на екрану.

Слика 3.4 а) Излаз програма **primer3_4** (први део)



Слика 3.4 б) Излаз програма **primer3_4** (други део)



Слика 3.4 в) Излаз програма **primer3_4** (трећи део)

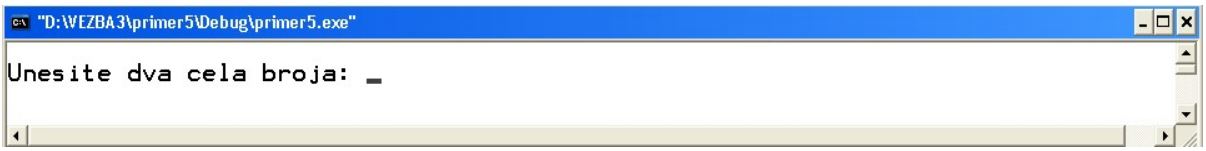
Изворни кôд програма **primer3_5**

```
1:      /*primer3_5.c*/
2:      /*Odredjivanje veceg od dva zadata cela broja (ili prvog od jednakih)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int a, b, max;
8:
9:              printf("\nUnesite dva cela broja: ");
10:             scanf("%d%d", &a, &b);
11:
12:             max = a;                        /*pretpostavka: a ima vecu vrednost*/
13:             if(b>max)                       /*suprotan slucaj od pretpostavljenog*/
14:                     max = b;
15:
16:             printf("\nVeci od dva zadata broja je: %d\n\n", max);
17:     }
```

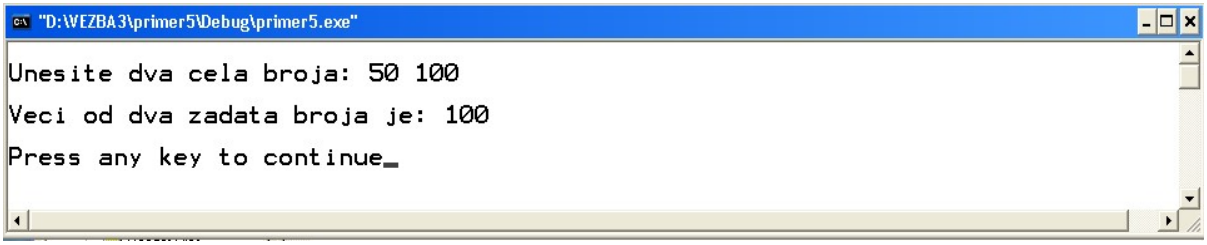
Анализа програма **primer3_5**

Програм у овом примеру помоћу *if* наредбе одређује већи од два задата броја на следећи начин. Прогласи први од задатих бројева за већи (линија 12.) и његову вредност додели променљивој *max.* Онда испитује супротно од претпоставке, да ли је други број већи (линија 13.), и ако је тако онда додељује вредност другог броја променљивој *max* (линија 14.).

Промпт програма и приказ резултата на екрану могу се видети на слици 3.5 а) и слици 3.5 б), респективно.



Слика 3.5 а) Излаз програма **primer3_5** (први део)



Слика 3.5 б) Излаз програма **primer3_5** (други део)

Изворни кôд програма **primer3_6**

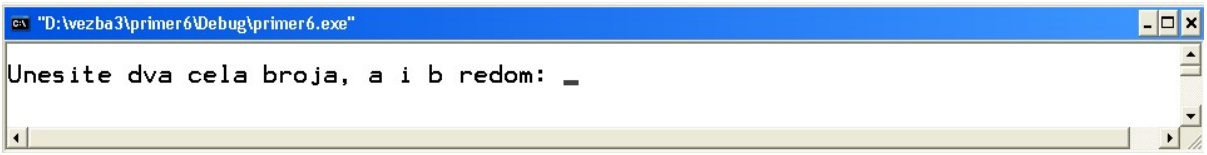
```
1:      /*primer3_6.c*/
2:      /*Prikaz na ekranu poruke o relaciji*/
3:      /*izmedju dva zadata cela broja, primenom naredbe if*/
4:
5:      #include  <stdio.h>
6:
7:      main( )
8:      {       int a, b;
9:
10:             printf("\nUnesite dva cela broja, a i b redom: ");
11:             scanf("%d%d", &a, &b);
12:
13:             if(a == b)
14:                     printf("\na je jednako b\n\n");
15:             else  if(a > b)
16:                     printf("\na je vece  od b\n\n");
17:             else
18:                     printf("\na je manje  od b\n\n");
19:     }
```

Анализа програма **primer3_6**

Програм у овом примеру такође одређује већи од два задата броја, али за разлику од претходног програма даје на екрану поруке у сва три могућа случаја:

- ако су вредности задатих бројева исте (*if* у линији 13.) приказује на екрану једну поруку (линија 14.),
- ако то није био случај, већ је први задати број већи од другог (*else if* у линији 15.), приказује на екрану другу поруку (линија 16.),
- ако ни то није био случај (*else* у линији 17.), онда је први задати број мањи од другог, приказује на екрану трећу поруку (линија 18.).

Промпт програма и приказ резултата на екрану могу се видети на слици 3.6 а) и слици 3.6 б), респективно.



Слика 3.6 а) Излаз програма **primer3_6** (први део)

Слика 3.6 б) Излаз програма **primer3_6** (други део)

Изворни кôд програма **primer3_7**

```
1:      /*primer3_7.c*/
2:      /* Odredjivanje veceg od dva zadata cela broja (ili prikaz poruke*/
3:      /*ako su jednaki), primenom uslovnog operatora*/
4:
5:      #include  <stdio.h>
6:
7:      main( )
8:      {       int a, b;
9:
10:             printf("\nUnesite dva cela broja, a i b redom: ");
11:             scanf("%d%d", &a, &b);
12:
13:             if(a == b)
14:                     printf("\n\n a je jednako b \n");
15:             else
16:                     printf("\n\n%d je veci broj\n\n",(a > b) ? a : b);
17:     }
```

Анализа програма **primer3_7**

И овај програм одређује већи од два задата броја, али помоћу условног оператора (јединог тернарног оператора у језику *C*).

Анализа прати текст програма од линије 13.:

- ако је вредност првог задатог броја иста као и вредност другог задатог броја (*if* у линији 13.) програм приказује одговарајућу поруку о томе (линија 14.),
- ако бројеви нису истих вредности (*else* у линији 15.) програм приказује на екрану поруку о томе који број има већу вредност помоћу условног оператора (линија 16.) Израз *(a > b) ? a : b* има вредност променљиве *a* ако је израз *(a > b)* тачан или вредност променљиве *b* ако израз *(a > b)* није тачан.

Наредба *x = (a > b) ? a : b;* је еквивалент следећем блоку наредби:

```
if(a>b)
        x = a;
else
        x = b;
```

Промпт програма и приказ резултата на екрану могу се видети на слици 3.7 а) и слици 3.7 б), респективно.

Слика 3.7 а) Излаз програма **primer3_7** (први део)

Слика 3.7 б) Излаз програма **primer3_7** (други део)

Практични савети

- ✓ Наредбу или блок наредби које се извршавају по било којој наредби за контролу тока програма треба померити за хоризонтални табулатор, ради боље прегледности кôда.
- ✓ Пазити на приоритет оператора код сложених наредби.
- ✓ Писање заграда се препоручује (онда када није сигурно познат приоритет оператора).
- ✓ Блок наредби је најлакше писати на следећи начин: прво заграде за почетак и крај блока *{ }* а онда попуњавање блока наредбама.
- ✓ При употреби логичких оператора сваки операнд који има вредност ≠0 сматра се логички тачним, а сваки са вредношћу =0 логички нетачним, без обзира на тип операнда.
- ✓ Један од оператора у језику C је и позив функције. Позивима стандардних функција реализује се велики број задатака у језику C. Кратки описи стандардних функција које се најчешће користе могу се видети у табелама 7., 8., 9. и 10. у Прилогу.
- ✓ Мада у овој фази учења језика *If else* наредба делује читљивија од примене условног оператора често се у програмима на језику *C* наредба *if-else* пише помоћу условног оператора *?:*

Евентуалне грешке

- ➢ Не треба поистовећивати оператор = и оператор ==. Први се користи увек код доделе вредности, а други код релацијских израза за испитивање еквиваленције.
- ➢ Писање релацијских оператора са размаком између два симбола оператора није дозвољено. (не писати: < =, > =, = = и ! =
  већ: <=, >=, == и !=)
  као ни замена места симболима код истих оператора (није дозвољено писати: =<, => и =!)
- ➢ Писање знака за крај наредбе (;) после затворене заграде *If* услова значи извршавање празне наредбе ако је задовољен *If* услов.

Задаци за припрему вежбе 3. у лабораторији

Задатак 1.

Дате су вредности целобројних променљивих: *a = 1, b = 1, c = 2, d = 2*. Написати декларације и иницијализације поменутих променљивих у истој линији програма на језику C.

Које ће вредности имати променљиве *a, b, c* и *d* након примене следећих оператора:

```
a = b++;        c *= d;        d *= a + b;
```

Задатак 2.

Дате су вредности променљивих: *a = 0x01ff, b = 0x02fe*. Које ће вредности приказати на екрану следећи позив *C* функције *printf():*

```
printf("%x", a&b);        printf("%x", a|b);        printf("%x", a^b);
```

Задатак 3.

Упростити следећи сегмент програма на језику C:

```
if(a>b)
        c = 1;
if(a>b)
        d = 2;
if(a<=b)
        c = 3;
if(a<=b)
        d = 4;
```

Задатак 4.

Написати наредбе програма на језику C којима се за три улазна податка, целе бројеве *a, b* и *c*, променљивој d додељује вредност 1 (у супротном вредност 0) ако су:

а) вредности све три променљиве *a, b* и *c* једнаке,

б) вредности све три променљиве *a, b* и *c* парни бројеви,

в) вредности све три променљиве *a, b* и *c* позитивни бројеви не већи од 100.

Задатак 5.

Написати наредбу *if* којом се у програму на језику *C* за задату вредност *x* израчунава вредност *y:*

y = -10, *ако је x < 0,*
y = x, *ако је* $0 \leq x < 10$,
y = x - 10, *ако је* $x \geq 10$.

Задаци за вежбу 3. у лабораторији

Задатак 1.

Написати *C* програм који тражи да се преко тастатуре изабере опција *Da/Ne* (унесе слово *D* или *N*) и приказује на екрану одговарајућу поруку. Ако није изабрана ни једна од предвиђених опција, програм треба да пријави грешку.

Задатак 2.

Написати програм на језику *C* који пријављује на екрану одговарајућу поруку, ако цео број, задат преко тастатуре (у опсегу од два бајта) има вредност 1 на местима 1., 2., и 3. бита (на местима бита тежине 2^0, 2^1 и 2^2).

Задатак 3.

Написати програм на језику *C* који одређује највећи од три цела броја, задата преко тастатуре.

Задатак 4.

Написати програм на језику *C* који за два реална броја *x* и *y,* задата преко тастатуре, израчунава *z* по обрасцу:

$$z = (min(x,y) + 0.5) / (1 + max^2(x,y))$$

Задатак 5.

Написати програм на језику *C* који за два реална броја *x* и *y,* задата преко тастатуре, израчунава *z* по обрасцу:

$z = min(x,y),$ *за* $y \geq 0,$
$z = max^2(x,y),$ *за* $y < 0.$

Задатак 6.

Написати програм на језику *C* који задате реалне бројеве *x* , *y* и *z* множи са 2, ако је задовољен услов $x \geq y \geq z$, или им мења знак, ако није задовољен поменути услов.

Вежба 4.

Наредбе циклуса у програмима на језику *C*

Наредбе циклуса: *for*, *while* и *do..while*

Угњеждене наредбе за контролу тока програма

Стандардне функције улаза и излаза, карактер по карактер

Примери

Изворни кôд програма **primer4_1**

```
1:      /*primer4_1.c*/
2:      /*Prikaz na ekranu neparnih brojeva od 1 do 10 (for petlja)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int x;
8:
9:              for(x=1; x<=10; x+=2)
10:                     printf("\n%d ", x);
11:             printf("\n\n");
12:     }
```

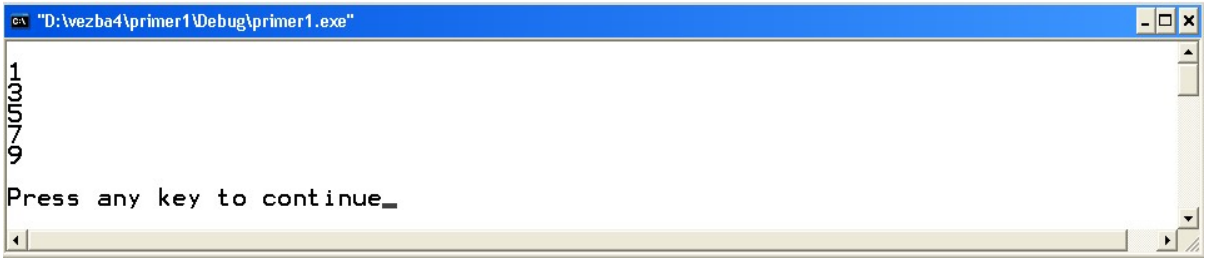
Анализа програма **primer4_1**

Овај програм има *for* наредбу циклуса (*for* петљу) са експлицитним бројачем (линије 9. и 10.). У линији 9. су изрази *for* циклуса, написани унутар малих заграда:

1. почетни израз: променљива *x* се иницијализује на вредност 1,
2. израз који садржи услов: ако је садржај променљиве *x* број мањи или једнак броју 10,
3. израз промене: увећање садржаја променљиве *x* за број 2.

У линији 10. је наредба која се извршава у *for* циклусу, а то је приказ на екрану садржаја променљиве *x.*

Дакле, ево како ради *for* наредба. Само на почетку циклуса, променљива *x* се иницијализује на вредност 1 (први израз унутар заграда *for* наредбе), а онда све док је *x<=10* (други израз унутар заграда *for* наредбе) траје циклус: извршава се наредба (приказ садржаја променљиве *x*), а онда следи корак промене, увећава се садржај променљиве *x* за 2 (трећи израз унутар заграда for наредбе). Тако се на екрану приказују само непарни бројеви од 1 до 9 (слика 4.1). Наредба у линији 11. не припада *for* циклусу (циклус је завршен знаком *;* у линији 10).



Слика 4.1 Излаз програма **primer4_1**

Изворни кôд програма **primer4_2**

```
1:      /*primer4_2.c*/
2:      /*Prikaz na ekranu slova X u 3 reda i 10 kolona (ugnjezdena  for  petlja)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int i, j;
8:
9:              printf("\n");
10:
11:             for(i=0; i<3; i++)              /*3 iteracije spolj. petlje za 3 reda*/
12:             {       for(j=0; j<10; j++)     /*10 iteracija un. petlje u svakom redu*/
13:                             printf("X");
14:             printf("\n");
15:             }
16:             printf("\n");
17:     }
```

Анализа програма **primer4_2**

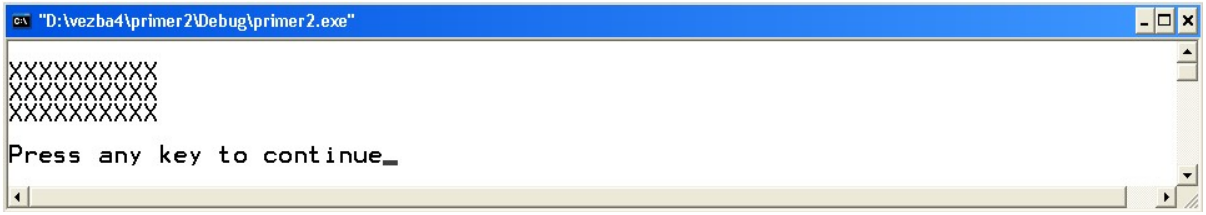
Овај програм има две *for* петље (једну унутар друге) помоћу којих исписује на екрану слово *X,* у три реда и десет колона. Сваки цуклус има своју бројачку променљиву (то су у овом случају *i* и *j*). Ево како раде петље.

Унутар прве итерације (*i*=0) спољашње петље (од линије 11. до линије 15.) програм извршава десет итерација (од *j=0* до *j<10*) унутрашње петље (у линијама 12. и 13.) и приказује на екрану десет слова *X* у првом реду.

Унутар сваке следеће итерације (*i=1* и *i=2*) спољашње петље извршава се по десет итерација унутрашње петље (приказ по десет слова *X* у другом и трећем реду*).*

На крају сваке итерације спољашње петље, програм има позив функције *printf()* за прелаз у нови ред (линија 14.).

По завршеној спољашњој петљи програм још једном позива функцију *printf()* за прелаз у нови ред (линија 16.), да би редови слова *X* били издвојени на екрану (слика 4.2).



Слика 4.2 Излаз програма **primer4_2**

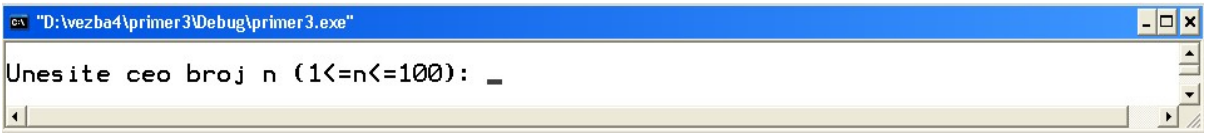
Изворни кôд програма **primer4_3**

```
1:      /*primer4_3.c*/
2:      /*Izracunavanje sume S=1+2+..+n,gde je n zadati ceo broj (for petlja)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int n, s, suma=0;
8:
9:              printf("\nUnesite ceo broj n (1<=n<=100): ");
10:             scanf("%d", &n);
11:
12:             if(n<1 || n>100)                /*vrednost n van dozvoljenog opsega*/
13:                     printf("\nZadati broj je van dozvoljenog opsega\n");
14:             else                            /*vrednost n u dozvoljenom opsegu*/
15:             {for(s=1;s<=n;s++)
16:                     suma += s;              /*skraceni zapis od: suma=suma+s*/
17:
18:             printf("\nSuma prvih %d pozitivnih celih brojeva je %d\n\n",n,suma);
19:             }
20:     }
```

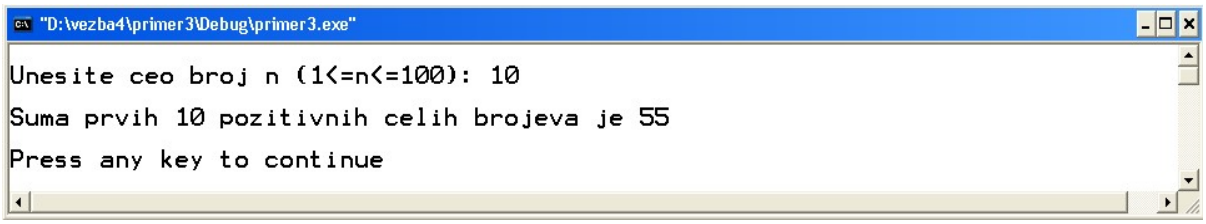
Анализа програма **primer4_3**

Овај програм израчунава и приказује на екрану суму од првих *n* позитивних целих бројева (*n* је цео број задат преко тастатуре).

За почетак сума се иницијализује (*suma=0*) у линији 7. да би касније у програму било могуће повећање вредности ове суме. Након учитане вредности са тастатуре (слика 4.3 а)), програм пита у линији 12. да ли је број ван задатог опсега (од 1 до 100) и у том случају појављује се на екрану порука (линија 13.). У супротном, иде се на члан *else* (од 15. до 19. линије). У *for* петљи, од *s=1* до *s<=n* (у *n* итерација) сума се увећава за број *s*. Дакле: *suma = 0 + 1 + 2 + …+ n* (слика 4.3 б)).



Слика 4.3 а) Излаз програма **primer4_3** (први део)



Слика 4.3 б) Излаз програма **primer4_3** (други део)

Изворни кôд програма **primer4_4**

```
1:      /*primer4_4.c*/
2:      /*Izracunavanje stepena realne osnove celobrojnim eksponentom*/
3:      /*za zadate vrednosti osnove  i eksponenta  (for petlja)*/
4:
5:      #include  <stdio.h>
6:
7:      main( )
8:      {       int i, eksp;
9:              float osnova, stepen = 1.0;
10:
11:             printf("\nUnesite za osnovu realan broj ");
12:             scanf("%f", &osnova);
13:             printf("\nUnesite za eksponent pozitivan ceo broj: ");
14:             scanf("%d", &eksp);
15:
16:             if(eksp<=0)
17:                     printf("\nNije zadat pozitivan ceo broj za eksponent\n\n");
18:
19:             else
20:             {
21:                     for(i=1;i<=eksp;i++)
22:                     stepen *= osnova;                   /*stepen=stepen*osnova*/
23:
24:                     printf("\nBroj %f na stepen %d je %f\n\n",osnova,eksp,stepen);
25:             }
26:     }
```

Анализа програма **primer4_4**

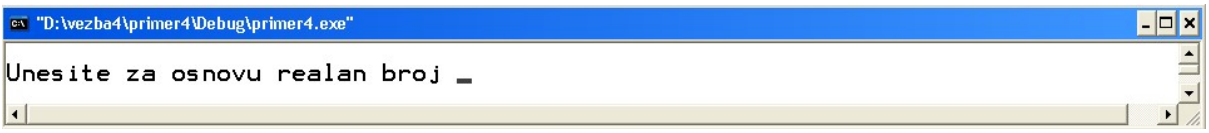
Програм прихвата преко тастатуре (линије 12. и 14.): реалну основу (линија 12.) као и целобројни експонент, да би израчунао степен реалне основе целобројним експонентом (слике 4.4 а) и 4.4 б)).

Садржај резултујуће променљиве *stepen* иницијализује се на почетку (у линији 9.) на вредност 1.0 (због каснијих множења).

У случају непозитивне вредности задатог експонента програм приказује на екрану само одговарајућу поруку (линије 16. и 17.).

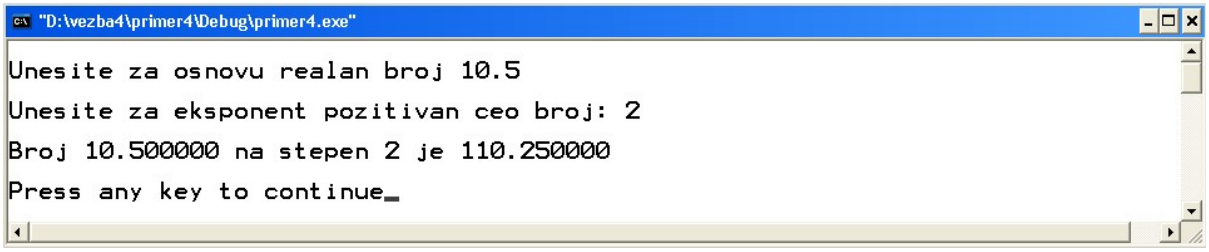
У линијама 21. и 22. је *for* петља која има *eksp* итерација, за израчунавање степена. У свакој итерацији садржај променљиве степен (из претходне итерације) множи се са вредношћу основе.

За задате вредности *osnova=10.5, eksp=2*, на пример, *for* петља има 2 итерације за рачунање степена 10.5^2 : *stepen = 1.0 * 10.5 * 10.5* (слика 4.4 в))



Слика 4.4 а) Излаз програма **primer4_4** (први део)

Слика 4.4 б) Излаз програма **primer4_4** (други део)



Слика 4.4 в) Излаз програма **primer4_4** (трећи део)

Изворни кôд програма **primer4_5**

```
1:      /*primer4_5.c*/
2:      /*Izracunavanje srednje vrednosti zadatog niza realnih brojeva*/
3:      /*koriscenje broja 0 kao STOP koda za kraj niza (while petlja)*/
4:
5:      #include <stdio.h>
6:      #define STOP 0
7:
8:      main( )
9:      {       int n=0;
10:             float x, suma=0;
11:
12:             printf("\nUnesite niz realnih brojeva (0 za kraj):\n\n");
13:             scanf("%f",&x);
14:
15:             while(x!=STOP)              /*ciklus ce biti prekinut kada se zada 0*/
16:             {       suma += x;
17:                     n++;
18:                     scanf("%f", &x);
19:             }
20:
21:             if(n==0)
22:                     printf("\nNije zadat niz realnih brojeva.\n\n");
23:             else
24:                     printf("\nAritmeticka sredina zadatog niza je %f\n\n", suma/n);
25:     }
```

Анализа програма **primer4_5**

У овом програму се, при рачунању средње вредности задатог низа бројева, користи *while* наредба циклуса са излазом на врху (*while* петља). Ова

наредба садржи само један израз у заградама и то је израз услов. Док је услов овог израза испуњен, извршава се тело while петље (једна или блок наредби). Када услов више није испуњен, завршава се петља. Овде нема као код *for* наредбе бројача и праћења њихових вредности.

На почетку (у линији 10.) овај програм садржи све потребне иницијализације *(n=0, suma=0)*, као и прихватање првог задатог броја са тастатуре (слика 4.5 а)).

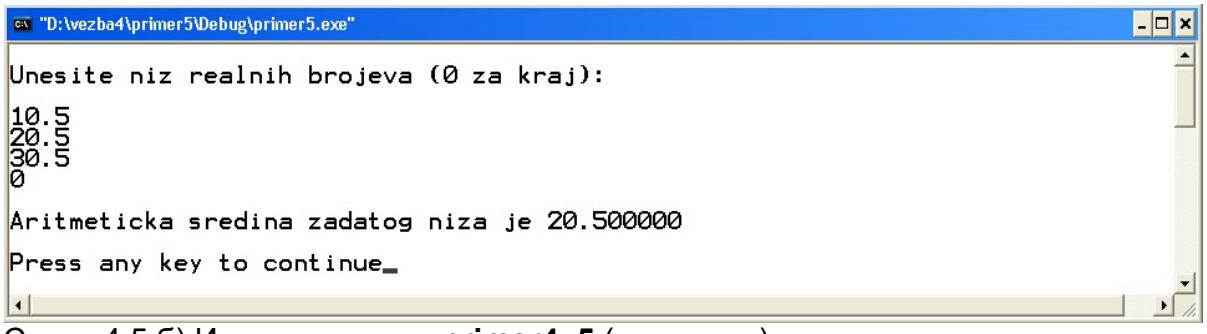
Од линије 15. до линије 19. је једна *while* наредба. У овом случају се под условом да је испуњен *while* услов (у линији 15.) извршава блок наредби (од линије 16. до линије 19.)

Програм у линији 15. проверава да ли задати број нема вредност 0 и све док је задовољен овај *while* услов, извршава блок наредби од линије 16. до линије 19. Променљивој *suma* у свакој итерацији циклуса додаје се вредност задатог броја, променљивој *n* његов редни број и прихвата се са тастатуре нови број. Након задатог броја 0 биће прескочен *while* блок наредби и завршена петља.

У случају да није задат низ (линија 21.), већ само један број, програм не би ни ушао у *while* петљу (испуњеност услова *while* петље проверава се на самом почетку *while* наредбе и због тога се зове петља са излазом на врху) и приказује на екрану поруку (линија 22.), а у супротном, средњу вредност задатог низа (*suma/n*) из линије 24. (слика 4.5 б)).

Наредба *while* је у овом програму омогућила читање са тастатуре низа бројева чија дужина није дефинисана на почетку програма, већ је позната тек по унетом СТОП кôду (некој задатој вредности броја)

Слика 4.5 а) Излаз програма **primer4_5** (први део)



Слика 4.5 б) Излаз програма **primer4_5** (други део)

Изворни кôд програма **primer4_6**

```
1:    /*primer4_6.c*/
2:    /*Odredjivanje brojeva karaktera i linija u ulaznom tekstu (while petlja)*/
3:
4:    #include  <stdio.h>
5:
6:    main( )
7:    {       int c, nc=0, nl=0;                  /*nc - broj karaktera i nl - broj linija*/
8:
9:            while((c = getchar()) != EOF)       /*EOF - znaka za kraj unosa teksta*/
10:           {       nc++;                       /*(detektovan Ctrl+Z sa tastature)*/
11:
12:                   if(c=='\n')
13:                           nl++;
14:           }
15:
16:           printf("\nUnetih karaktera:  %d\nLinija:  %d\n\n", nc, nl);
17:   }
```

Анализа програма **primer4_6**

Следећа три примера ове вежбе немају нове наредбе, у њима је већ позната наредба циклуса *while* примењена код читања, тестирања и копирања текста који се задаје преко тастатуре, и то карактер по карактер.

У овом примеру програм прихвата улазни текст, броји његове карактере, као и линије и приказује ове бројеве на екрану. За број карактера и број линија користе се променљиве *nc* и *nl* и обе су иницијализоване у линији 7.

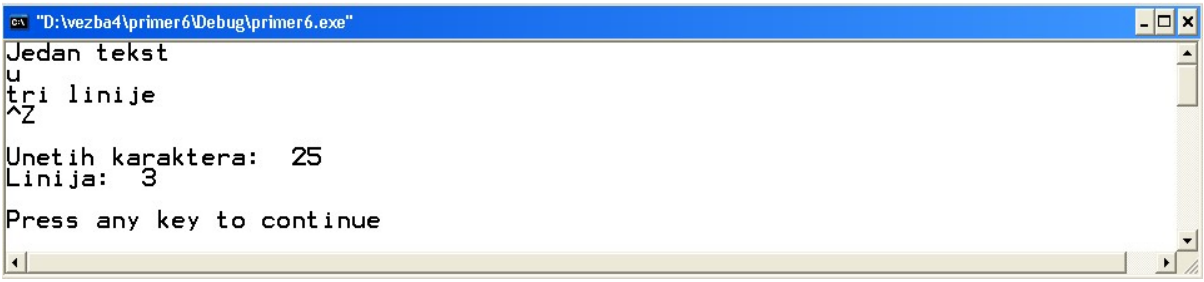
Следи *while* петља, а у услову *while* петље једна нова константа *EOF* (*EndOfFile*) из библиотеке *stdio.h*. То је ознака за крај код текстуалних датотека али се може симулирати преко тастатуре, истовременим притиском на тастере *Ctrl* и *Z*, да означи крај уноса текста.

Наредба *while* (од линије 9. до линије 14.) има услов који испитује резултат функције *getchar()*. То је стандардна функција из библиотеке *stdio.h*. која прихвата један знак са тастатуре, а као резултат даје *ASCII* кôд тог знака (један цео број). Резултат ове функције прво се смести у једну целобројну променљиву (*c*), а тек онда се испитује вредност ове променљиве. Због тога се израз услов може написати само на начин, на који је написан у линији 9. овог програма (све написане заграде су неопходне: *while((c = getchar()) != EOF)*.

Све док са тастатуре не прихвати ознаку за крај уноса текста, програм :

- инкрементира број карактера (*nc*) за сваки учитани знак,
- инкрементира број линија (*nl*) само за учитани знак за прелаз у нови ред (*if* наредба у линијама 12. и 13.).

На слици 4.6 може се видети приказ резултата програма на екрану.



Слика 4.6 Излаз програма **primer4_6**

Изворни кôд програма **primer4_7**

```
1:      /*primer4_7.c*/
2:      /*Kopiranje ulaznog teksta na ekran, uz zamenu svakog niza*/
3:      /*uzastopnih praznih znakova jednim praznim znakom (while petlja)*/
4:
5:      #include  <stdio.h>
6:      #define NIJE_PRAZNO     'a'
7:      #define PRAZNO          ' '
8:      #define HOR_TAB         '\t'
9:
10:     main( )
11:     {       int c, predh_c = NIJE_PRAZNO;
12:
13:             while((c = getchar()) != EOF)
14:             {       if(c == HOR_TAB)
15:                             c = PRAZNO;
16:
17:                     if(c != PRAZNO || predh_c != PRAZNO)
18:                             putchar(c);
19:
20:                     predh_c = c;
21:             }
22:     }
```

Анализа програма **primer4_7**

Програм овог примера копира улазни текст на екран тако што више узастопних празних знакова замењује једним празним знаком.

У линијама 6., 7. и 8. дефинисане су симболичке константе (имена их добро описују),за константу *NIJE_PRAZNO* је случајно изабрани кôд слова *a* (*'a'*).

Променљива *predh_c* користи се за чување статуса: био је или није био претходно празан знак. За почетак је ова променљива иницијализована на статус *NIJE_PRAZNO.* Све док не учита ознаку за крај уноса текста програм:

- ако је текући знак хоризонтални табулатор, он се прихвата као празан знак (линије 14. и 15.),

- ако текући знак није празан или ако претходни знак није празан, програм копира тај знак на екран (линије 17. и 18.). Ово значи да неће бити копиран само празан знак, коме предходи исти такав знак. Функција *putchar(c)* (такође из библиотеке stdio.h) приказује на екрану знак чији *ASCII* кôд чува променљива *c* (написана као аргумент функције),
- на крају сваке итерације циклуса променљивој *predh_c* додели се *ASCII* кôд текућег знака, да би био искоришћен у следећој итерацији, као кôд претходног знака.

На слици 4.7 може се видети приказ резултата програма на екрану.

Слика 4.7 Излаз програма **primer4_7**

Изворни кôд програма **primer4_8**

```
1:      /*primer4_8.c*/
2:      /*Kopiranje ulaznog teksta na ekran, po jednu rec u liniji (while petlja)*/
3:
4:      #include <stdio.h>
5:      #define U_RECI          1
6:      #define VAN_RECI        0
7:
8:      main( )
9:      {       int c, poz = VAN_RECI;
10:
11:             while((c=getchar())!=EOF)
12:             {       if(c==' ' || c=='\n' || c=='\t')   /*ako je prihvacen neki beli znak*/
13:                     {       if(poz==U_RECI)
14:                             {       putchar('\n');
15:                                     poz = VAN_RECI;
16:                             }
17:                     }
18:                     else                            /*ako nije prihvacen beli znak*/
19:                     {       poz = U_RECI;
20:                             putchar(c);
21:                     }
22:             }
23:     }
```

Анализа програма **primer4_8**

Овај програм копира по једну реч улазног текста на екран. После дефиниција симболичких константи *U_RECI* и *VAN_RECI* (линије 5. и 6.) и

иницијализације променљиве *poz* (за чување статуса: знак је унутар речи или знак није унутар речи), до ознаке за крај уноса текста, програм:

- ако је задат било који бели знак (' ', \n', или '\t') и само ако претходно није био бели знак већ је статус позиције био *U_RECI,* програм има прелаз у нови ред и постављање статуса позиције *VAN_RECI*,
- ако није прихваћен бели знак, програм поставља статус позиције *U_RECI* (за било коју предходну вредност) и копира знак на екран.

На слици 4.8 може се видети приказ резултата програма на екрану.

Слика 4.8 Излаз програма **primer4_8**

Изворни кôд програма **primer4_9**

```
1:      /*primer4_9.c*/
2:      /*Prikaz na ekranu stepenova broja 2 (2, 4, 8, 16, …)*/
3:      /* u zadatom opsegu od 0 do max (do..while)*/
4:
5:      #include  <stdio.h>
6:      #define MAX 100
7:
8:      main( )
9:      {       int stepen = 2, max;
10:
11:             do                              /*ponavljanje poruke i prihvatanje broja*/
12:             {       printf("\nUnesite jedan ceo broj (2<n<100): ");
13:                     scanf("%d", &max);
14:             }
15:             while(max<=2 || max>=MAX);      /*dok je max van trazenog opsega*/
16:
17:             do                              /*stepen se izracunava u petlji*/
18:             {       printf("\n%d", stepen);
19:                     stepen *= 2;            /*skraceno od: stepen=stepen*2*/
20:             }
21:             while (stepen <= max);          /*dok ima vrednost ne vecu od max*/
22:
23:             printf("\n\n");
24:     }
```

Анализа програма **primer4_9**

У примерима до краја ове вежбе програми имају и наредбу циклуса *do..while* (петљу *do.. while)*. За разлику од наредбе *while*, код ове наредбе се услов пише на крају (петља са излазом на дну). Једном се изврши наредба (блок наредби) написана између речи *do* и *while*, а онда се испитује *do.. while* услов написан на крају наредбе. Све док је овај услов испуњен, наредба (блок наредби) се понавља. Прва итерација петље је обавезна, а остале условне.

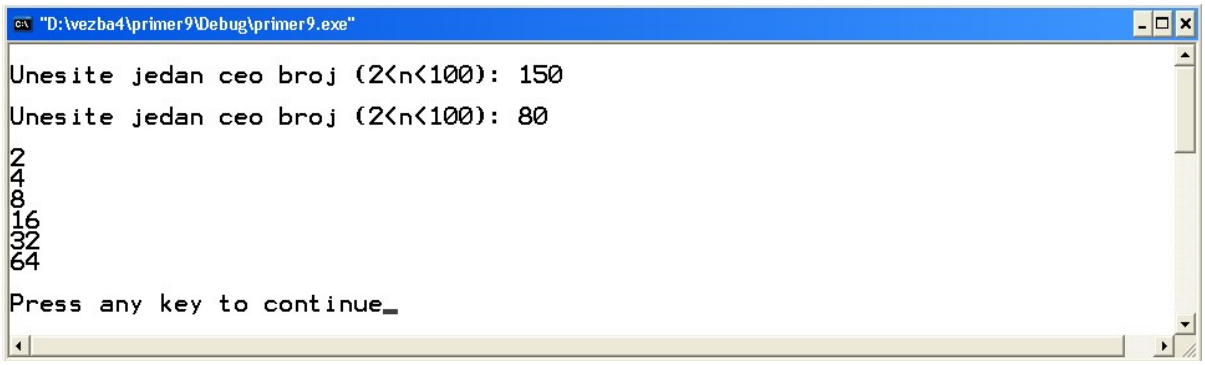
Програм овог примера приказује на екрану све степенове броја 2, у задатом опсегу целих бројева.

Помоћу прве *do.. while* наредбе (линије од 11. до 15.) прихвата максималну вредност опсега (вредност већа од 2 и мања од 100). Наредба ради на следећи начин: приказује на екрану промпт (слика 4.9 а)) и прихвата са тастатуре број *max* (линије 12. и 13.), све док број није у дозвољеном опсегу , док је задовољен *do.. while* услов у линији 15. (слика 4.9 б)). Када се учита број у траженом опсегу, програм напушта прву *do.. while* петљу и иде даље.

Следи још једна *do.. while* наредба (линије од 17. до 21.). У првој, обавезној, итерацији ове наредбе садржај променљиве *stepen* (иницијализован претходно на вредност 2) се прикаже на екрану и помножи бројем 2. Све док је вредност степена мања или једнака максималној вредности опсега (док је испуњен *do..while* услов у линији 21.), извршавају се даље итерације петље (приказ на екрану и множење бројем 2). Када вредност степена постане већа од максималне вредности опсега, програм напушта и другу *do.. while* петљу, тако да су на екрану приказане вредности само тражених степенова броја 2 (слика 4.9 в)).

Слика 4.9 а) Излаз програма **primer4_9** (први део)

Слика 4.9 б) Излаз програма **primer4_9** (други део)



Слика 4.9 в) Излаз програма **primer4_9** (трећи део)

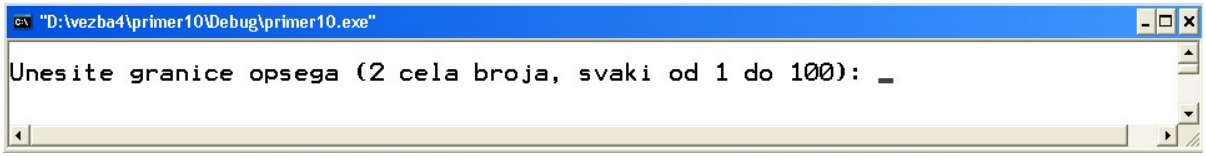
Изворни кôд програма **primer4_10**

```
1:      /*primer4_10.c*/
2:      /*Prikaz na ekranu brojeva zadatog opsega, deljivih brojem 3(do..while)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int x, a, b, brojac = 0;
8:
9:              do                              /*ponavljanje poruke i prihvatanje brojeva*/
10:             {       printf("\nUnesite granice opsega (cele brojeve, od 1 do 100): ");
11:                     scanf("%d%d", &a, &b);
12:             }                               /*sve dok nisu zadovoljeni zahtevi*/
13:             while(a<1 || a>100 || b<=a || b>100);
14:
15:             printf("\nBrojevi u opsegu od %d do %d deljivi sa 3:\n", a, b);
16:
17:             for(x=a; x<=b; x++)
18:             {       if(x%3==0)
19:                     {       printf("%d ", x);
20:                             brojac++;
21:                     }
22:
23:                     if(brojac%10==0)
24:                             printf("\n");
25:             }
26:     }
```

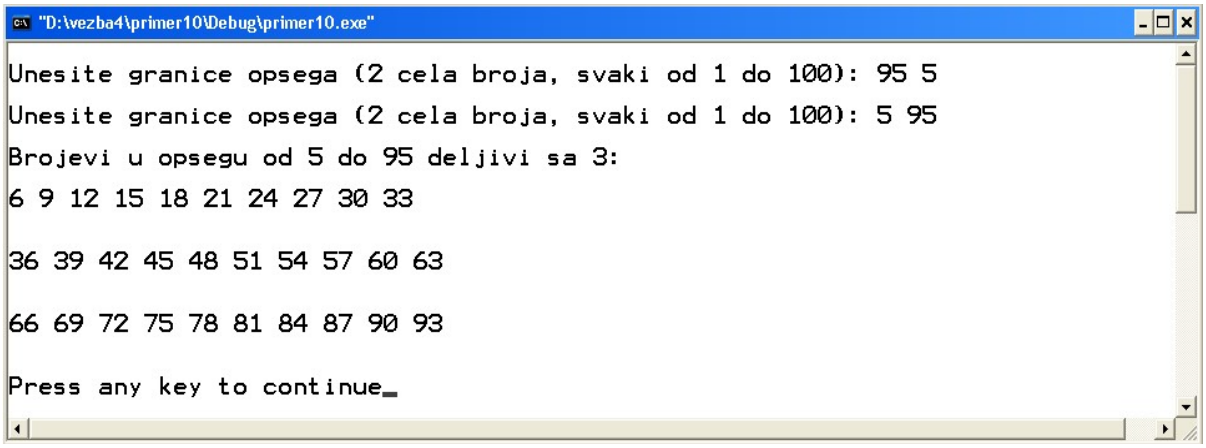
Анализа програма **primer4_10**

Програм овог примера извршава једну *do..while* наредбу, од линије 9. до линије 13. У првој, обавезној итерацији петље, програм тражи (слика 4.10 а)) и прихвата са тастатуре два цела броја *a* и *b*. Све док су задати бројеви ван дозвољеног опсега од 1 до 100, или други задати број није већи од првог задатог броја (док је задовољен *do..while* услов) програм поново тражи два цела броја и чита их са тастатуре (линије 10. и 11.). Када прихвати два цела броја *a* и *b*, оба у опсегу од 1 до 100 и други број већи од првог, програм напушта *do..while* петљу и иде даље.

У *for* наредби (од линије 17. до линије 25.) која пролази кроз све целе бројеве, од броја *a* до броја *b,* угњеждена је *if* наредба (од линије 18. до линије 21.), која проверава да ли је задати број дељив бројем 3 (да ли је остатак дељења бројем 3 број 0). За сваки број у опсегу од *a* до *b*, који задовољава поменути услов, програм приказује на екрану вредност броја и инкрементира вредност бројача (иницијализованог на почетку програма). Програм користи бројач да прикаже на екрану највише по 10 бројева у једном реду (после 10 бројева има прелаз у нови ред) у линијама 23. и 24. (слика 4.10 б)).



Слика 4.10 а) Излаз програма **primer4_10** (први део)



Слика 4.10 б) Излаз програма **primer4_10** (други део)

Практични савети

- ✓ Убацивање празних редова изнад и испод наредби за контролу тока програма чине програм читљивијим.
- ✓ *for* наредба бројачког циклуса треба да има сва своја три израза (најпрегледнија је у том случају).
- ✓ Почетни израз и израз промене у *for* наредби не треба да садрже иницијализацију и промену већег броја променљивих (пожељно је да имају само бројачку променљиву).
- ✓ Наредбу *while* користити код блокова наредби који се опционо извршавају (блок наредби се извршава само ако је задовољен постављени услов), а наредбу *do..while* код блокова наредби који се опционо понављају (блок наредби се изврши једном, онда се понавља док је испуњен услов наредбе).

Евентуалне грешке

- ➢ Коришћење знака , уместо знака ; између израза *for* наредбе.
- ➢ Знак ; написан после затворене заграде за изразе *for* наредбе значи извршавање празне наредбе у циклусу.
- ➢ Изостављена промена бројачке променљиве значи бесконачну петљу.

Задаци за припрему вежбе 4. у лабораторији

Задатак 1.

Написати програм на језику *C* који израчунава суму квадрата свих парних и кубова свих непарних бројева, у опсегу задатих целих бројева *a* и *b, a<b (for* петља).

Задатак 2.

Написати програм на језику *C* који одређује број реченица (број појављивања знака .) у улазном тексту и исписује овај број на екрану (*while* петља).

Задаци за вежбу 4. у лабораторији

Задатак 1.

Написати програм на језику *C* који исписује на екрану:
а) табелу децималних и хексадецималних бројева од 0 до 16 (*for* петља),
б) велика слова од *A* до *Z* и њихове *ASCII* кôдове (*for* петља).

Задатак 2.

Написати програм на језику *C* који исписује на екрану слова *X* и *Y* на следећи начин:

1: *XXXXX YYYYY XXXXX*
2: *XXXXX YYYYY XXXXX*
3: *XXXXX YYYYY XXXXX*

помоћу наредбе за штампање једног слова *X* и наредбе за штампање једног слова *Y* (*for* петља).

Задатак 3.

Написати програм на језику *C* који приказује на екрану следеће (*for* петља):

а)
```
*
**
***
****
*****
```

б)
```
*****
****
***
**
*
```

Задатак 4.

Написати програм на језику *C* који одређује највећу и најмању вредност у низу задатих целих бројева. При томе, као знак за крај уноса низа, користи број 0. (*while* петља).

Задатак 5.

Написати програм на језику *C* који копира улазни текст на екран на следећи начин:

‘ ‘	у слово	*B*
‘\t”	у слово	*T*
“\n”	у слово	*N*

Вежба 5.

Наредбе скокова и вишеструке селекције у програмима на језику *C*

Наредба за излаз из петље: *break*

Наредба за излаз из итерације петље: *continue*

Наредба вишеструке селекције: *switch*

Наредба скока: *goto*

Примери

Изворни кôд програма **primer5_1**

```
1:      /*primer5_1.c*/
2:      /*Kopiranje sa tastature na ekran niza celih brojeva,*/
3:      /*sve dok se ne unese 10 brojeva ili broj 0 kao STOP kod (break)*/
4:
5:      #include  <stdio.h>
6:      #define STOP 0
7:
8:      main( )
9:      {       int i, x;
10:             printf("\nUnesite cele brojeve (10 njih ili do stop koda 0):\n\n");
11:
12:             for(i=0; i<10; i++)
13:             {       scanf("%d", &x);
14:                     if(x==STOP)                     /*za zadati broj 0*/
15:                             break;                  /*izlaz iz petlje*/
16:                     printf("\n%d", x);
17:             }
18:
19:             printf("\n\n");
20:     }
```

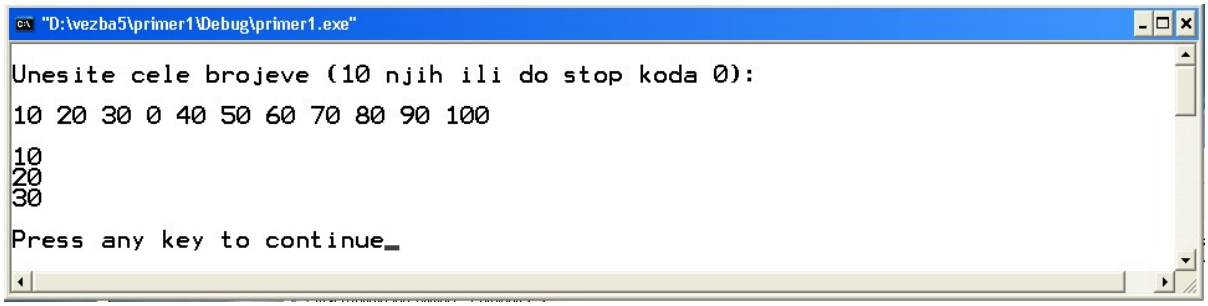
Анализа програма **primer5_1**

Програм из овог примера тражи да се преко тастатуре унесу цели бројеви (слика 5.1 а)), њих 10, или до броја 0 (СТОП кôда дефинисаног симболом у линији 6.)

Без *if* наредбе, у линијама 14. и 15, програм би прихватио и приказао на екрану све задате бројеве (у *for* петљи). Овако, ако задати број има вредност 0 (линија 14.) извршава се наредба *break* (линија 15.) која значи излаз из текуће петље (прескакање наредбе из линије 16. и крај *for* петље) и извршава наредба (из линије 19.) која је прва на реду после петље. Програм приказује на екрану све задате бројеве до броја 0 (слика 5.1 б)).

Наредба *break* може се користити само за излаз из петље (*for, while, do..while)* или за излаз из *switch* наредбе, која ће бити објашњена касније, у овој вежби.

Слика 5.1 а) Излаз програма **primer5_1** (први део)



Слика 5.1 б) Излаз програма **primer5_1** (други део)

Изворни кôд програма **primer5_2**

```
1:      /*primer5_2.c*/
2:      /*Prikaz na ekranu brojeva od 1 do 30, deljivih brojem 3 (continue)*/
3:
4:      #include  <stdio.h>
5:
6:      main( )
7:      {       int x;
8:
9:              printf("\nBrojevi od 1 do 30 koji su deljivi brojem 3:\n");
10:
11:             for(x=1; x<=30; x++)
12:             {       if(x%3  !=  0)                  /*za broj nedeljiv sa 3*/
13:                             continue;               /*izlaz iz iteracije petlje*/
14:                     printf("\n%d", x);
15:             }
16:
17:             printf("\n\n");
18:     }
```

Анализа програма **primer5_2**

Да нема наредби у линијама 12. и 13. овај програм би приказао на екрану све целе бројеве, од 1 до 30. Наредба *if* у линији 12. испитује да ли број *x* није дељив бројем 3 (*if(x%3 != 0)*). У том случају извршава наредбу *continue*, која значи излаз из текуће итерације петље, али не и излаз из петље (под поменутим условом, програм прескаче све остале наредбе петље за текућу итерацију и иде на почетак следеће итерације исте петље). Дакле, наредбу за приказ броја на екрану (линија 14.) програм прескаче сваки пут када задати број није дељив бројем 3, али не прекида *for* петљу све док није испуњен услов *for* петље док је задати број мањи или једнак броју 30 (слика 5.2).

Наредба *continue* може се користити само за излаз из итерације петље (*for, while, do..while)*.

Слика 5.2 Излаз програма **primer5_2**

Изворни кôд програма **primer5_3**

```
1:      /*primer 5_3.c*/
2:      /*Ponavljanje prompta, dok uneti broj nije u zadatom opsegu */
3:      /*i crtanje histograma vrednosti ako je u zadatom opsegu (go..to)*/
4:
5:      #include <stdio.h>
6:
7:      main( )
8:      {       int i, n;
9:
10:     start:  printf("\nUnesite ceo broj od 0 do 10: ");
11:             scanf("%d", &n);
12:
13:             if (n < 0 || n > 10 )
14:                     goto start;             /*za broj n van opsega - skok na pocetak*/
15:             else if (n == 0)
16:                     goto loc0;              /*za broj 0 - skok na loc0*/
17:             else
18:                     goto loc1;              /*za broj u opsegu - skok na loc1*/
19:
20:     loc0:   printf("\nIzabran je broj 0 za kraj\n\n");
21:             goto end;
22:
23:     loc1:   for(i=0; i<n; i++)
24:                             printf("*");
25:             goto start;
26:
27:     end:    ;
28:     }
```

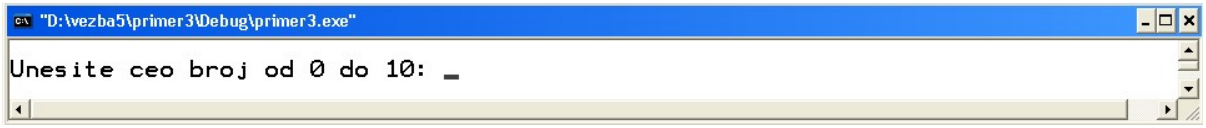
Анализа програма **primer5_3**

Програм у овом примеру показује структуру *goto* наредбе.

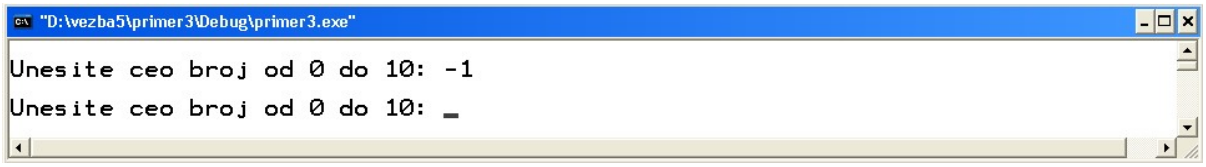
Лабела се у тексту програма означава помоћу идентификатора и знака двотачка (на почетку линије 10. је лабела *start*:) Након прихватања (линије 10. и 11.) броја са тастатуре (слика 5.3 а)) програм ради следеће: ако је број ван траженог опсега (линија 13.) скаче на наредбу, прву написану после лабеле *start* (повратак на промпт програма у линији 10.) (слика 5.3 б)). Ако то није био случај, већ је прихваћен сам број 0, следи скок на наредбу, прву која је написана после лабеле *loc0* (на линију 20.), а онда и скок на празну наредбу на крају програма (из линије 21. на линију 27.) Ако ни то није био случај, следи скок на наредбу написану прву после лабеле *loc1*(*for* наредба, у линијама 23. и 24., за штампање на екрану задатог броја знака ∗, а онда повратак на почетак програма (на промпт програма у линији 10.).

Наредба *goto* се пре може наћи у књигама о програмском језику *C*, него у самим програмима. Са употребом ове наредбе у угњежденим петљама треба пазити. Наредба *goto* је преузета из старијих програмских језика и у програмима на језику *C* увек је могућа замена ове наредбе неком другом, читљивијом наредбом.

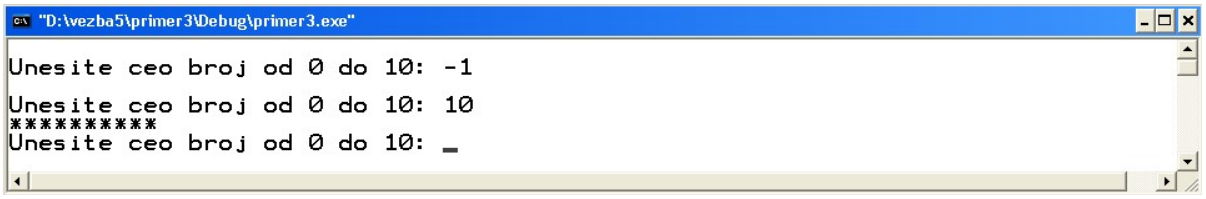
На сликама 5.3 в) и 5.3 г) може се видети да је резултат програма, за задата два различита цела броја, у траженом опсегу, а на слици 5.3 д) за број 0 који значи крај извршавања програма.



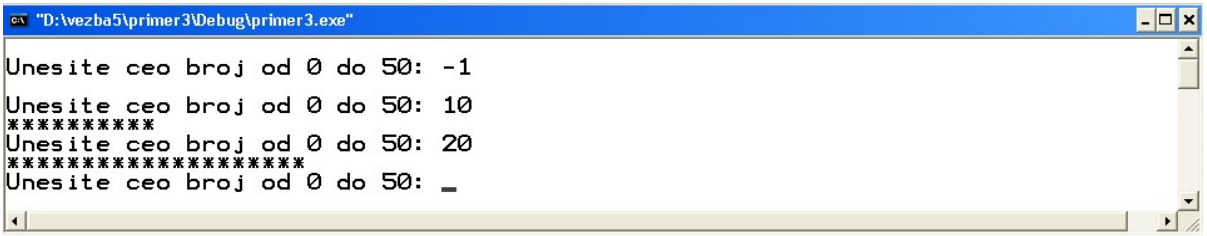
Слика 5.3 а) Излаз програма **primer5_3** (први део)



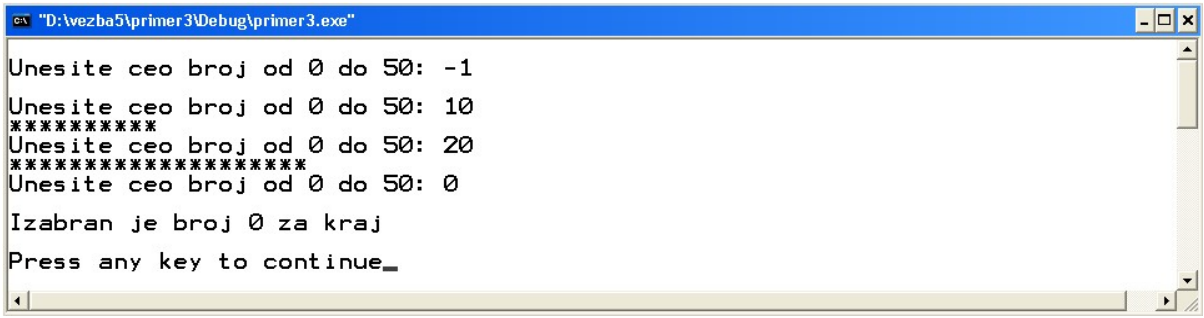
Слика 5.3 б) Излаз програма **primer5_3** (други део)



Слика 5.3 в) Излаз програма **primer5_3** (трећи део)



Слика 5.3 г) Излаз програма **primer5_3** (четврти део)



Слика 5.3 д) Излаз програма **primer5_3** (пети део)

Изворни кôд програма **primer5_4**

```
1:      /*primer5_4.c*/
2:      /* Izracunavanje srednje vrednosti s = (x1 + x2 +…+ xn)/n  */
3:      /* i standardne devijacije d = ((x1^2 + x2^2 +…+ xn^2)/n – s^2)^1/2  */
4:      /*za zadati niz realnih brojeva x, duzine 1<=n<=10 (while(1) i break)*/
5:
6:      #include <stdio.h>
7:      #include <math.h>
8:
9:      main( )
10:     {       int i, n;
11:             double x, s=0, d=0;
12:
13:             while(1)
14:             {       printf("\nUnesite broj elemenata niza (1<=n<=10): ");
15:                     scanf("%d", &n);
16:
17:                     if(n<1 || n>10)         /*za broj n van zadatog opsega*/
18:                     {       printf("\nZadati broj elemenata niza nije dozvoljen\n”);
19:                             break;          /*izlaz iz petlje*/
20:                     }
21:                     s=d=0;
22:                     printf("Elementi niza realnih brojeva? ");
23:                     for(i=1; i<=n; i++)
24:                     {       scanf("%lf", &x);
25:                             s += x; d += x*x;
26:                     }
27:
28:                     s /= n;
29:                     d = sqrt(d/n - s*s);
30:
31:                     printf("\nSrednja vrednost: %f\n", s);
32:                     printf("\nStandardna devijacija: %f\n", d);
33:             }
34:     }
```

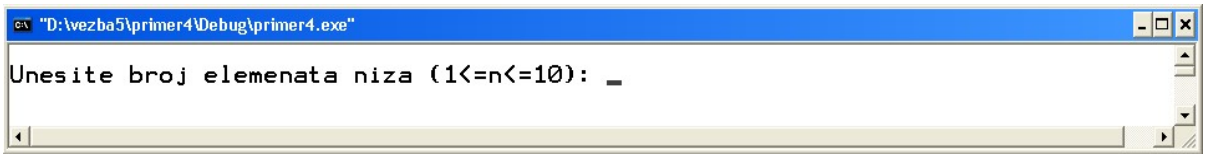
Анализа програма **primer5_4**

Програм овог примера, након свих потребних иницијализација (линија 11.), ради у *while(1)* петљи (од линије 13. до линије 33.). Петља *while(1)* би се извршавала бесконачно (на месту израза услова је број 1), да нема наредбе за излаз из петље (*break* у линији 19. под условом *if* наредбе, у линији 17).

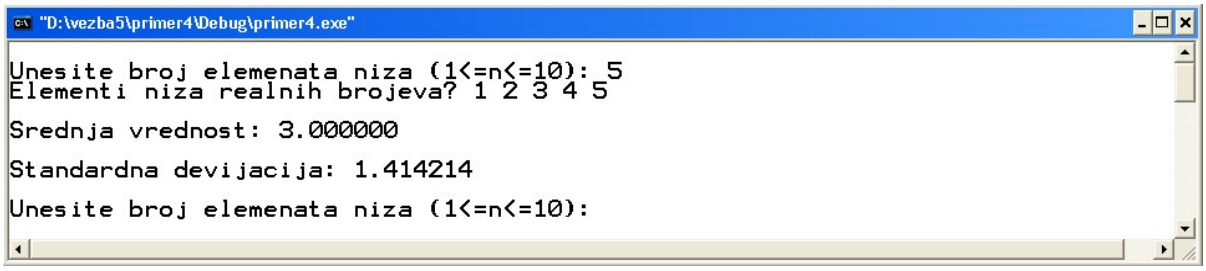
Наредбе петље се извршавају следећим редом: ако вредност задате дужине низа (*n*) (слика 5.4 а)) није у траженом опсегу (од 0 до 10), петља се прекида (слика 5.4 в)). Ако се није десио прекид програм иде даље, тражи да се унесе *n* елемената низа и прихвата их са тастатуре (слика 5.4 б)). Вредност сваког елемента додаје првој суми (променљива *s*, иницијализована на почетку програма), а квадрат вредности сваког елемента додаје другој суми (променљива *d*, такође иницијализованој на почетку програма).

По прекиду петље (унетој вредности броја ван задатог опсега) у програму се креира средња вредност низа тако што се прва сума подели бројем елемената низа (за средњу вредност је искоришћена променљива *s*) (линија 28.), а стандардна девијација низа по одговарајућем обрасцу (за стандардну девијацију је искоришћена променљива *d*) (линија 29.). Стандардна функција *sqrt()*, из библиотеке *math.h* (укључена у линији 7.) даје квадратни корен израза аргумента (написаног унутар заграда).

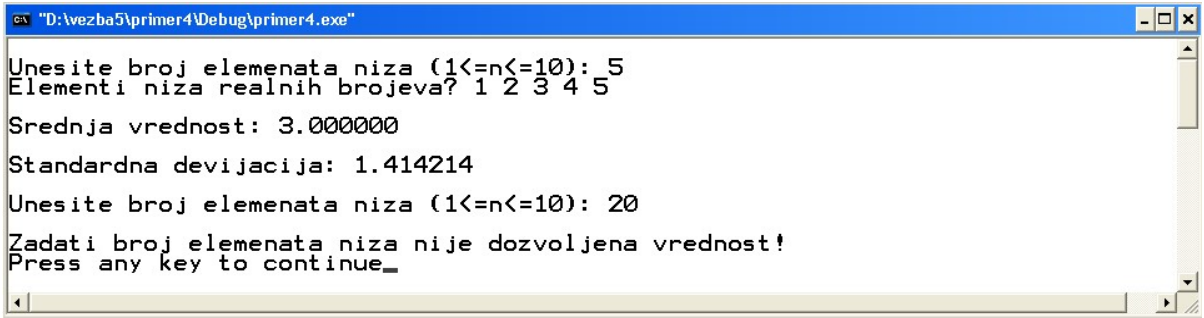
Програм у петљи израчунава и приказује на екрану средњу вредност и стандардну девијацију, за сваки задати низ, чија је дужина у опсегу од 0 до 10. Када прихвати са тастатуре, за дужину низа, број ван поменутог опсега, прекида се петља и завршава извршавање програма.



Слика 5.4 а) Излаз програма **primer5_4** (први део)



Слика 5.4 б) Излаз програма **primer5_4** (други део)



Слика 5.4 в) Излаз програма **primer5_4** (трећи део)

Изворни кôд програма **primer5_5**

```
1       /*primer5_5.c*/
2:      /*Prikaz bitova svake zadate celobrojne vrednosti (u opsegu 2 bajta)*/
3:      /*broj 0 je stop kod za unos brojeva (naredbe while i break)*/
4:
5:      #include <stdio.h>
6:      #define STOP            0
7:      #define VEL_BAJTA       8
8:      #define VEL_RECI        16
9:
10:     main( )
11:     {       int broj, poz_bita;
12:
13:             while(1)
14:             {       printf("\n\nUneti ceo broj (od 0 do 255) u decimalnom obliku: ");
15:                     scanf("%d", &broj);
16:
17:                     if(broj==STOP)                  /*za zadati broj 0*/
18:                             break;                  /*zlaz iz petlje*/
19:
20:                     printf("\nBinarni oblik zadatog broja: ");
21:
22:                     for(poz_bita=1; poz_bita <= VEL_RECI; poz_bita++)
23:                     {       if((broj >> (VEL_RECI - poz_bita)) & 1)
24:                                     printf("1");
25:                             else
26:                                     printf("0");
27:
28:                             if(poz_bita == VEL_BAJTA)
29:                                     printf(" ");            /*prikaz: po 8 bitova u grupi*/
30:                     }
31:             }
32:     }
```

Анализа програма **primer5_5**

У програму овог примера је опет *while(1)* петља (од 13. до 31. линије), која би била бесконачна без наредбе за прекид под одређеним условом (наредба *break* у линији 18., под условом из линије 17.).

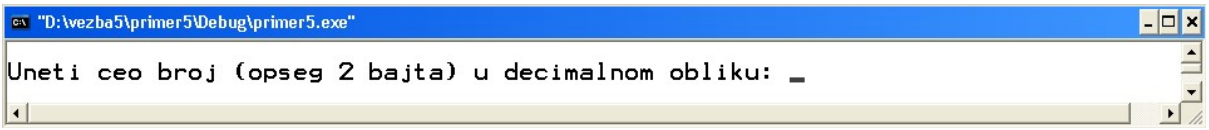
После прихваћеног целог броја (у децималном облику), наредбе поменуте петље се извршавају следећим редом: ако задати број има вредност 0, петља се прекида. Ако се није десио прекид програм иде даље и приказује задати број у бинарном облику тако што користи for петљу за контролу броја помераја битова задатог броја.

Програм почиње дефинисањем константи *VEL_BAJTA* и *VEL_RECI*, која може бити измењена. *VEL_RECI* се користи у услову *for* петље (у линији 22.). Битови броја (који чува променљива *bro*j) померају се унутар петље за

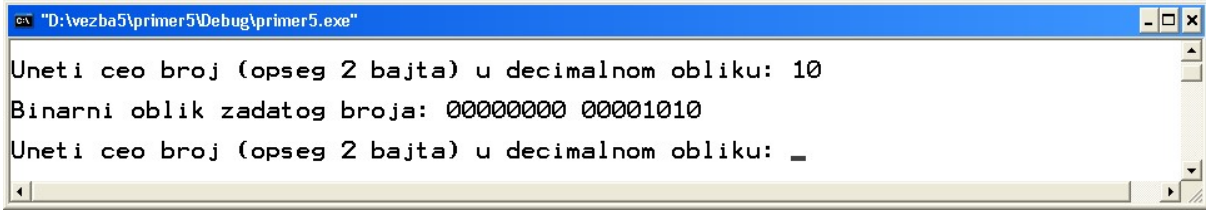
VEL_RECI - poz_bita удесно (у овом случају за 15, 14,...,0 места) тако да се сваки бит (16., 15., ..., 1.) доведе на место првог бита (најмање тежине) и примени се операција логички И на нивоу бита (&) над тим битом и бинарном цифром 1 (линија 23.). Ако је резултат примене поменутог оператора над одређеним битом и бинарном цифром 1 различит од 0 приказује се на екрану бинарна цифра 1 (линије 23. и 24.), а ако то није случај приказује се на екрану бинарна цифра 0 (линије 25. и 26.).

Резултат програма је приказ на екрану задатог броја у бинарном облику, од бита највеће, до бита најмање тежине (од 16. до 1. бита). У бинарној презентацији броја празним знаковима су издвојени бајтови. Након приказа 8 битова (*VEL_BAJTA*) приказује се на екрану празан знак (линије 28. и 29.).

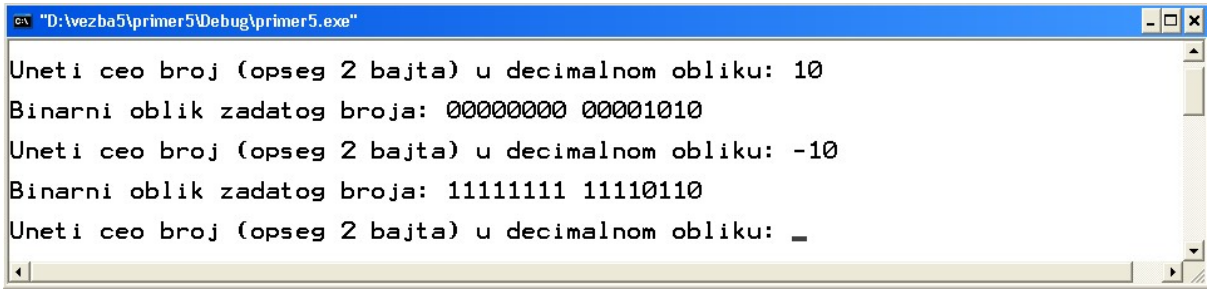
На сликама које следе, 5.5 а), 5.5 б), 5.5 в) и 5.5 г), могу се видети резултати програма, за задата два различита броја у траженом опсегу и за један број ван овог опсега.



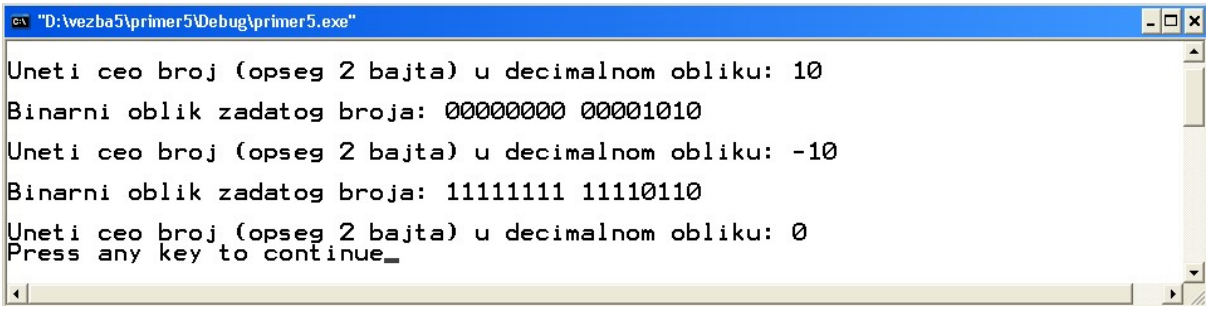
Слика 5.5 а) Излаз програма **primer5_5** (први део)



Слика 5.5 б) Излаз програма **primer5_5** (други део)



Слика 5.5 в) Излаз програма **primer5_5** (трећи део)



Слика 5.5 г) Излаз програма **primer5_5** (четврти део)

Изворни кôд програма **primer5_6**

```
1:      /*primer5_6.c*/
2:      /*Odredjivanje u ulaznom tekstu broja praznih karaktera, kao i broja*/
3:      /*znakova: tacka, zarez, dvotacka i tacka-zarez (naredbe switch i break)*/
4:
5:      #include <stdio.h>
6:
7:      main( )
8:      {   int c, prazno=0, interp=0;
9:
10:         while((c=getchar())!=EOF)
11:             switch(c)
12:             {       case ' ':
13:                                     prazno++;
14:                                     break;
15:                     case '.':
16:                     case ',':
17:                     case ':':
18:                     case ';':
19:                                     interp++;
20:                                     break;
21:                     default:
22:                                     break;
23:             }
24:
25:       printf("\nBroj praznina: %d", prazno);
26:       printf("\nBroj znakova tacka, zarez, dvotacka i tacka-zarez: %d\n", interp);
27:     }
```

Анализа програма **primer5_6**

У овом програму се први пут појављује наредба *switch.* То је наредба вишеструке селекције, помоћу које се предвиђају различите акције у програму, у зависности од вредности целобројне променљиве (или целобројног израза). Може се користити уместо великог броја чланова *else* и *else if* наредбе *if*.

У овом случају, *switch* наредба (од линије 11.) испитује вредност целобројне променљиве *c*, којој је претходно додељен *ASCII* кôд знака задатог преко тастатуре (резултат функције *getchar() у* линији 10.). Све док не прихвати знак за крај уноса текста (вредност константе EOF), програм испитује сваки задати знак.

Функција *getchar()* је већ коришћена функција библиотеке *stdio.h* за улаз са тастатуре, знак по знак. Ова функција има ехо, што значи да се на екрану показује сваки знак, унет преко тастатуре (то није излаз на екрану, он се може добити помоћу функције *putchar(),* из библиотеке *stdio.h*). Поред еха, функција има и бафероване улаз, сви унети знакови се уписују у улазни бафер, из ког се прослеђују даље подаци, по задатом прелазу у нови ред (притиснут тастер *Enter*).

Поред ове функције има функција које немају echo (када је потребно да се не види на екрану сваки укуцани знак), а и функција без баферованог улаза, када је потребно предвидети неку акцију у програму без чекања на тастер *Enter.* Једну од таквих функција, из библиотеке *conio.h,* користи програм следећег примера.

У програму овог примера:

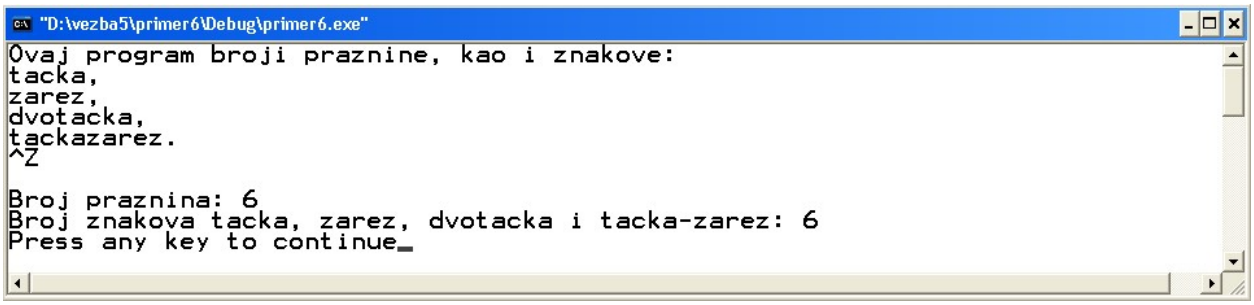
- ако је задат празан знак (case ' ':) увећава се претходно иницијализован садржај променљиве која чува број празних знакова (у линији 13.: *prazno*++) и излази се из *switch* наредбе (наредба *break* у линији 14.),
- ако је задат неки од тражених интерпункцијских знакова (тачка *('.'),* *зарез ((',')),* двотачка *(':')* или тачка-зарез *(';')*) увећава се претходно иницијализован садржај променљиве која чува њихов број (у линији 19.: *interp*++) и излази се из *switch* наредбе (наредба *break* у линији 20.).

Овај пример показује једну значајну особину *switch* наредбе. То је извршавање истог кôда (линије 19. и 20.) за више набројаних вредности променљиве (израза) која се испитује. Део кôда од линије 15. до линије 20. је скраћени запис блока наредби:

```
case '.':
        interp++;
        break;
case ',':
        interp++;
        break;
case ':':
        interp++;
        break;
case ';':
        interp++;
        break;
```

Дакле, *switch* наредба има посебан случај (*case*) за сваки од предвиђених знакова. Ако није задат ни један од предвиђених знакова, извршава се *default*: случај, само се излази из *switch* наредбе (наредба *break* у линији 22.)

Након излаза из *switch* наредбе, на екрану се приказују тражени бројеви (слика 5.6).



Слика 5.6 Излаз програма **primer5_6**

Изворни кôд програма **primer5_7**

```
1:      /*primer5_7.c*/
2:      /*Prikaz na ekranu poruke o strelici zadatoj preko tastature*/
3:
4:      #include  <stdio.h>
5:      #include <conio.h>
6:
7:      /*definisanje kodova za stampanjei citanje  znakova pojedinih strelica*/
8:      enum {          STAMPA_STREL_GORE = 24,STAMPA_STREL_DOLE,
9:                      STAMPA_STREL_DESNO,STAMPA_STREL_LEVO       };
10:
11:     #define CITA_STREL_GORE      72
12:     #define CITA_STREL_DOLE      80
13:     #define CITA_STREL_DESNO     77
14:     #define CITA_STREL_LEVO      75
15:
16:     main( )
17:     {       int c, kraj = 0;
18:
19:             while(1)
20:             {       printf("\nBirati strelicu %c %c %c %c\n",
21:                             STAMPA_STREL_GORE,STAMPA_STREL_DOLE,
22:                             STAMPA_STREL_DESNO,STAMPA_STREL_LEVO);
23:
24:                     c = getch();    /*citanje 1. dela koda*/
25:                     printf("\nPrvi bajt koda: %d", c);            /*prikaz 1. dela koda*/
26:                     c = getch();    /*citanje 2. dela koda*/
27:                     printf("\nDrugi bajt koda: %d", c);           /*prikaz 2. dela koda*/
28:
29:                     switch(c)
30:                     {       case CITA_STREL_GORE:
31:                                     printf("\n\nIzabrali ste strelicu gore.\n");   break;
32:                             case CITA_STREL_DOLE:
33:                                     printf("\n\nIzabrali ste strelicu dole.\n");   break;
34:                             case CITA_STREL_LEVO:
35:                                     printf("\n\nIzabrali ste strelicu levo.\n");   break;
36:                             case CITA_STREL_DESNO:
37:                                     printf("\n\nIzabrali ste strelicu desno.\n");break;
38:                             default:
39:                                     printf("\n\nNiste izabrali strelicu.\n"); kraj =1; break;
40:                     }
41:                     if(kraj)
42:                             break;
43:             }
44:     }
```

Анализа програма **primer5_7**

Поред стандардних *C* библиотека постоји и читав сет библиотека које олакшавају рад са улазно/излазним уређајима и специфичне су за различите врсте рачунара.

Програм овог примера користи једну нестандардну библиотеку *conio.h*. Функција *getch()* библиотеке *conio.h* искоришћена је за читање са тастатуре кодова стрелица, без бафероваља и без еха на екрану.

На почетку програма (у линијама 8. и 9.) дефинисани су кôдови за штампање на екрану појединих стрелица. Употребљено је набрајање симболичких константи, које имају узастопне целобројне вредности, помоћу службене *C* речи *enum*. То је скраћен запис за четири директиве *#define*. Дефиниција набрајања у линијама 8. и 9. је еквивалент за четири директиве:

```
#define STAMPA_STREL_GORE    24
#define STAMPA_STREL_DOLE    25
#define STAMPA_STREL_DESNO   26
#define STAMPA_STREL_LEVO    27
```

У набрајању константи (линије 8. и 9.) вредност (24) је дефинисана само за прву константу, за остале константе подразумевају се узастопне целобројне вредности редом, почев од прве следеће (25, 26 и 27). Дефинисање вредности за прву константу у низу није обавезно, ако изостане подразумевана вредност ове константе је 0).

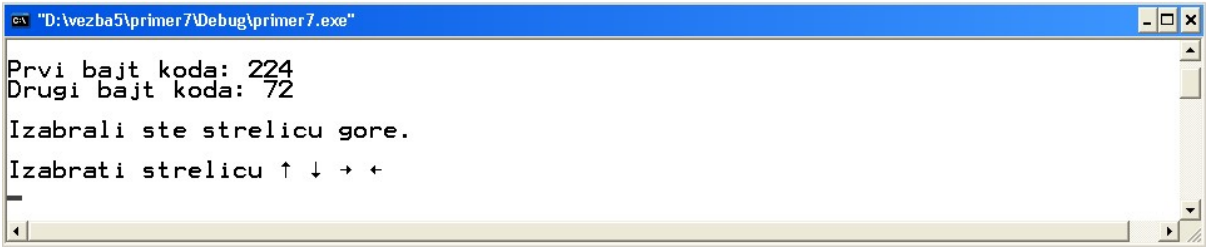
Кôдови за штампање симбола за стрелице, као и других симбола, могу се наћи у проширеној табели ASCII кôдова,у табели 12. Прилога.

Даље су дефинисани кôдови који се читају са тастатуре за изабране стрелице. За притиснути било који од тастера ↑, ↓, →, ←, *PageUp, PageDown, Home, End, Insert,* или *Delete*, програм чита два бајта кôда. За све поменуте знакове први бајт је исти (*0xE0*), а други се може добити ако се из програма одштампа на екрану кôд изабраног знака (помоћу функције *printf()).* На овај начин су добијени и дефинисани кôдови од линије 11. до линије 14. (За притиснути било који функцијски тастер први бајт кода је 0, а други бајт зависи од изабраног тастера).

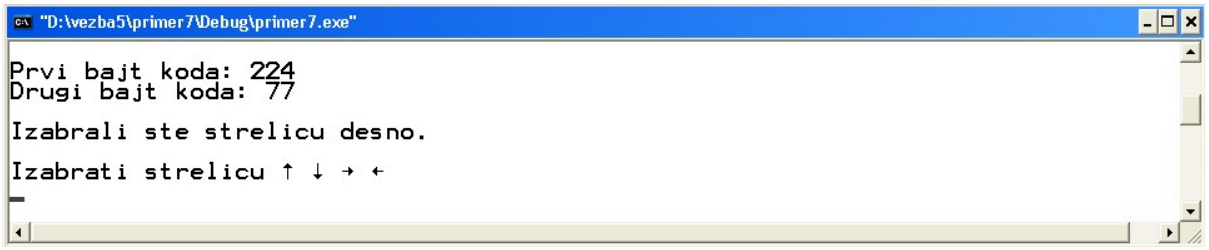
Програм прво у *while(1)* петљи штампа (од линије 20. до линије 22.) промпт на екрану (слика 5.7 а)). Следи читање кôда притиснутог тастера (линије од 24. до 26.) помоћу стандардне функције *getch()*, из библиотеке *conio.h* (укључене у линији 5.). Ова функција се користи, као и функција *getchar(),* за читање текста са тастатуре, знак по знак. За разлику од функције *getchar()* функција *getch()* нема ехо (не приказује на екрану сваки учитани знак) и нема бафероване улаз (није потребно притиснути тастер Enter да би програм прихватио знак са тастатуре). У конкретном примеру потребна су два позива функције *getch()* (линије 24. и 26.), за први и други бајт кôда, који се даље, из линија 25. и 27. приказују на екрану. Вредност другог бајта остаје у променљивој *c*, и испитује се даље у *switch* наредби. Садржај променљиве *c*, узастопно се пореди са сваком предвиђеном вредношћу (разни *case* случајеви). Када се пронађе једнака, на екрану се штампа одговарајућа порука, извршава се наредба *break* и прескаче се остатак наредбе *switch*. Ако нема једнаке вредности, извршава се *default* случај (излаз из наредбе *switch*).

На сликама 5.7 б) и 5.7 в) могу се видети резултати програма, за изабране две различите стрелице, а на слици 5.7 г) за знак који није стрелица.

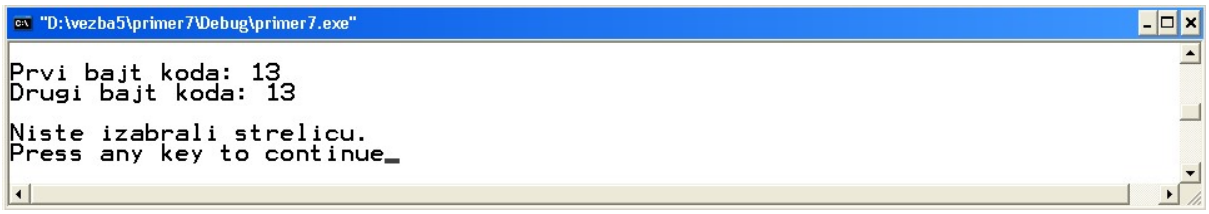
Слика 5.7 а) Излаз програма **primer5_7** (први део)



Слика 5.7 б) Излаз програма **primer5_7** (други део)



Слика 5.7 в) Излаз програма **primer5_7** (трећи део)



Слика 5.7 г) Излаз програма **primer5_7** (четврти део)

Практични савети

- ✓ Предвидети излаз из петље или из итерације петље, под одређеним условима, које нуде наредбе *break* и continue.
- ✓ Написати и *default*, за све непредвиђене случајеве *switch* наредбе.
- ✓ Много угњеждених наредби за контролу тока програма је тешко за читање. (Не претеривати са великим бројем наредби које се извршавају једна унутар друге.)

Евентуалне грешке

- ➢ Услед изостављања *break* наредбе код појединих случајева *switch* наредбе долази до пропадања из једног случаја у други који је написан следећи по реду.
- ➢ Изостављање наредбе *break* из петље *while(1)* значи бесконачну петљу у програму, која се може прекинути само насилним прекидом програма.

Задаци за припрему вежбе 5. у лабораторији

Задатак 1.

Написати *switch* наредбу еквивалентну *if* наредбама у програму на језику *C*.

```
if(a == 0)
        b++;
else if(a==1)
        c++;
else if ((a == 2) || (a == 3) || (a == 4))
        d++;
```

Задатак 2.

Написати програм на језику *C* који за задати редни број месеца у години,приказује на екрану његов број дана (За задати број 2, програм треба да пита да ли је година преступна (*d/n*) (*switch* наредба).

Задаци за вежбу 5. у лабораторији

Задатак 1.

Написати програм на језику *C* који за сваки задати низ целих бројева (преко тастатуре прихвата прво дужину низа *n, 1<=n<=10*, а онда и вредности самих елемената) одређује индекс и вредност елемента низа чија је вредност најближа некој задатој вредности. Задата дужина низа 0 значи крај извршавања програма (*break*).

Задатак 2.

Написати програм на језику *C* који одређује број белих знакова (*' ', '\t', '\v'* и *'\n'*), као и број свих осталих знакова у улазном тексту (*switch*).

Задатак 3.

Написати програм на језику *C* који прихвата са тастатуре симбол за један од бинарних аритметичких оператора (+, -, ∗, / или %), као и два реална операнда, а онда приказује на екрану резултат (при томе онемогућује дељење нулом) (*switch*).

Задатак 4.

Написати програм на језику *C* који решава квадратну једначину $ax^2 + bx + c = 0$ (при томе разматра различите корене једначине) *(switch).*
за $a \neq 0$ и $d = b^2 - 4ac$:

$d > 0$: $x_1 = \frac{-b + \sqrt{d}}{2a}$ $x_2 = \frac{-b - \sqrt{d}}{2a}$

$d = 0$: $x_1 = \frac{-b}{2a}$

$d < 0$: $x_1 = \frac{-b}{2a}$ $y_1 = \frac{\sqrt{-d}}{2a}$ $x_2 = x_1$ $y_2 = -y_1$

за $a = 0$ и $b \neq 0$:

$$x_1 = \frac{-c}{b}$$

за $a = 0$ и $b = 0$: штампање одговарајуће поруке на екрану.

Вежба 6.

Нумерички низови у програмима на језику *C*

Једнодимензионални низови

Декларација низова

Врсте иницијализације низова

Употреба низова

Дводимензионални низови

Примери

Изворни кôд програма **primer6_1**

```
1:      /*primer6_1.c*/
2:      /*Stampanje na ekranu tabele sa rednim brojevima, indeksima,*/
3:      /*vrednostima i histogramom vrednosti elemenata inicijalizovanog niza*/
4:
5:      #include <stdio.h>
6:      #define MAX 5
7:
8:      main( )
9:      {       int i, j, niz[] = {15, 3, 9, 7, 11};
10:
11:             printf("\nRBr\tIndeks\tVrednost\tHistogram\n");
12:             for(i=0; i<MAX; i++)
13:             {       printf("\n%d\t%d\t%d\t\t",i+1,i,niz[i]);
14:
15:                     for(j=0; j<niz[i]; j++)
16:                             printf("*");
17:             }
18:
19:             printf("\n\n");
20:     }
```

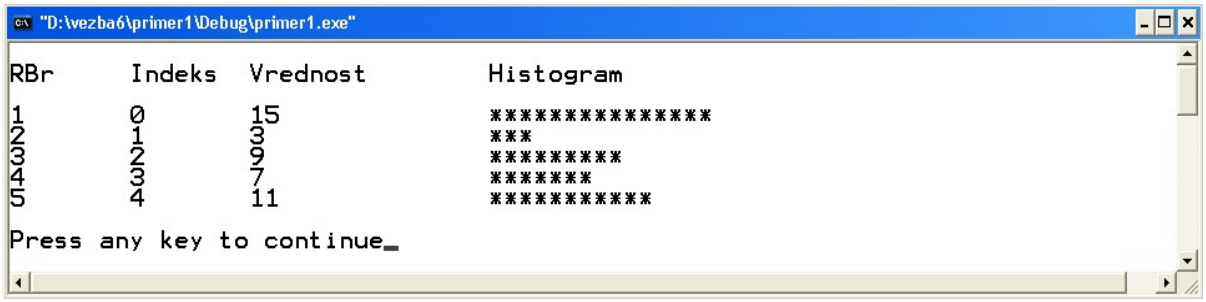
Анализа програма **primer6_1**

Низ је колекција објеката истог типа. У програмима који следе декларисани су и користе се низови променљивих основног типа (*char, int, float*,...). За приступ одређеном елементу у низу користи се име низа (које је заједничко за све елементе) и индекс.

У програму овог примера декларисан је низ целих бројева (линија 9.) који се тако и зове (*niz*). Средње заграде говоре да се ради о низу, а дужина низа је у овом случају одређена бројем иницијалних вредности елемената (унутар великих заграда).

Из линије 11. програм приказује заглавље, а помоћу спољашње *for* петље (линије од 12. до 17.) приказује на екрану за сваки елемент низа: редни број, индекс, вредност и хистограм вредности од знака * (унутрашња *for* петља, у линијама 15. и 16.). Редни број елемента у програму на језику *C* разликује се од индекса (за 1. је већи). У конкретном примеру елемент низа са редним бројем 1. има индекс 0, а елемент са редним бројем *MAX* има индекс *MAX -1* (слика 6.1).

У свим примерима ове вежбе програм приступа елементима низа помоћу индекса. Овај начин се реализује писањем имена низа и индекса унутар средњих заграда (*niz[i]*). Индекс може имати вредност од 0 (први елемент низа) до *MAX-1* (последњи елемент низа), за дефинисану дужину низа MAX. У сваком случају индекс може бити израз који има позитивну целобројну вредност.



Слика 6.1 Излаз програма **primer6_1**

Изворни кôд програма **primer6_2**

```
1:      /*primer6_2.c*/
2:      /*prikaz na ekranu neinicijalizovanog i inicijalizovanog*/
3:      /*niza celih brojeva*/
4:
5:      #include  <stdio.h>
6:      #define MAX 10
7:
8:      main( )
9:      {       int i, opcija, niz[MAX];
10:
11:             printf("Neinicijalizovan sadrzaj deklarisanog niza:");
12:             for(i=0; i<MAX; i++)
13:                     printf("\n%d ",niz[i]);
14:
15:             printf("\nDa li zelitie da inicijalizujete niz (D/N): ");
16:             opcija = getchar();
17:
18:             if(opcija =='D' || opcija =='d')
19:             {       printf("\nUnesite cele brojeve: \n");
20:
21:                     for(i=0; i<MAX; i++)
22:                     {       printf("niz[%d] = ", i);
23:                             scanf("%d",&niz[i]);
24:                     }
25:
26:                     printf("\nInicijalizovan sadrzaj niza:");
27:                     for(i=0; i<MAX; i++)
28:                             printf("\n%d ",niz[i]);
29:             }
30:     }
```

Анализа програма **primer6_2**

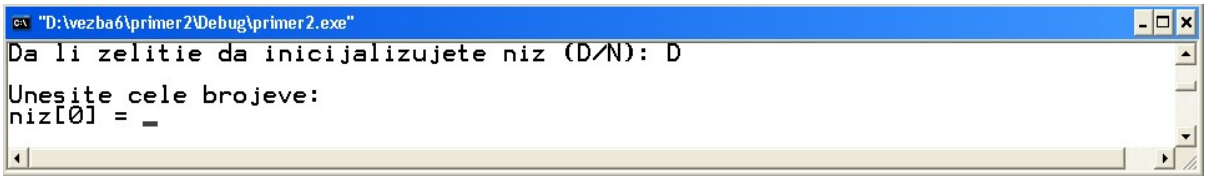
Програм овог примера декларише низ целих бројева, дужине *MAX* (вредност 10). Када нема иницијализације унутар декларације низа обавезно је дефинисање дужине низа, унутар средњих заграда (линија 9.).

У *MAX* итерација (од 0 до *MAX-1*) *for* петље (линије 12. и 13.) програм приказује вредности елемената декларисаног, али неиницијализованог низа. На слици 6.2 а) може се видети да су у том случају садржаји елемената низа неке случајне (затечене) вредности у меморијским локацијама, које се користе за елементе низа.

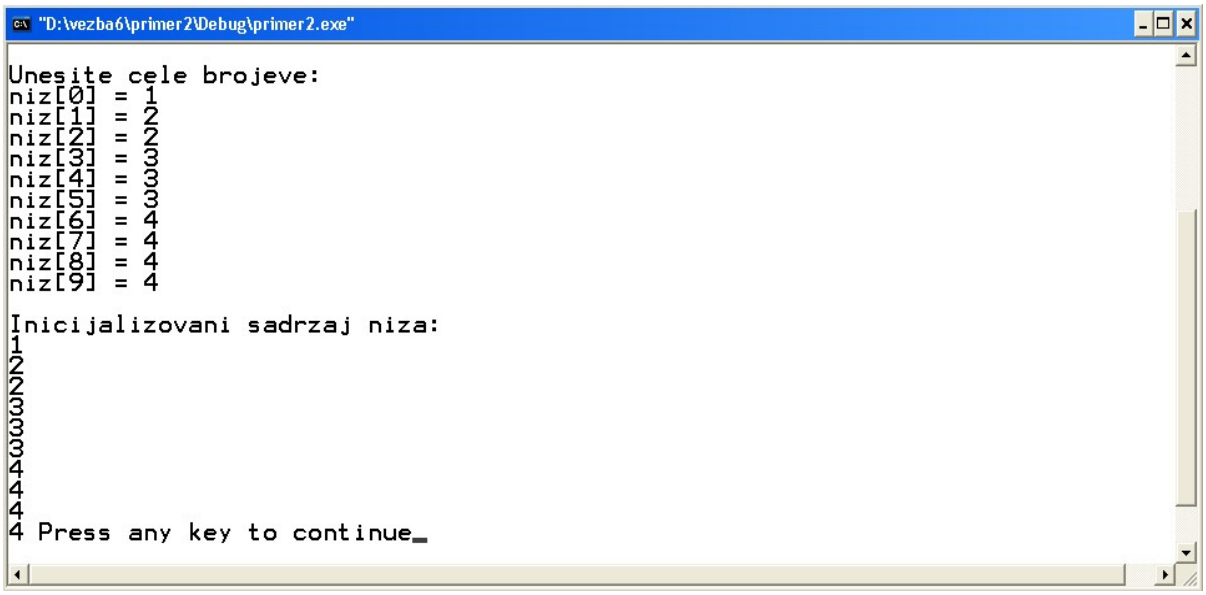
Програм даље тражи да се потврди иницијализација (линије 15. и 16.). Ако добије потврду са тастатуре у *for* петљи (линије од 21. до 24.) иницијализује елементе низа на вредности задате са тастатуре (слика 6.2 б)). Исписује на екрану, у посебној линији, име низа и индекс унутар заграда за сваки елемент декларисаног низа, а по уносу целобројне вредности смешта задату вредност у одговарајући елемент низа.

На крају, програм приказује на екрану иницијализовани низ (слика 6.2 в)).

Слика 6.2 а) Излаз програма **primer6_2** (први део)



Слика 6.2 б) Излаз програма **primer6_2** (други део)



Слика 6.2 в) Излаз програма **primer6_2** (трећи део)

Изворни кôд програма **primer6_3**

```
1:      /*primer6_3.c*/
2:      /*Izracunavanje srednje vrednosti zadatog niza celih brojeva*/
3:
4:      #include  <stdio.h>
5:      #define MAX 10
6:
7:      main( )
8:      {       int i, n, niz[MAX], suma = 0;
9:
10:             do
11:             {       printf("\nUnesite duzinu niza (n<=%d): ", MAX);
12:                     scanf("%d", &n);
13:             }
14:             while(n<=0 || n>MAX);
15:
16:             printf("\nUnesite vrednosti elemenata niza: ");
17:             for(i=0; i<n; i++)
18:             {       scanf("%d", &niz[i]);
19:                     suma += niz[i];
20:             }
21:             printf("\nSrednja vrednost zadatog niza je %f\n", (float)suma/n);
22:     }
```

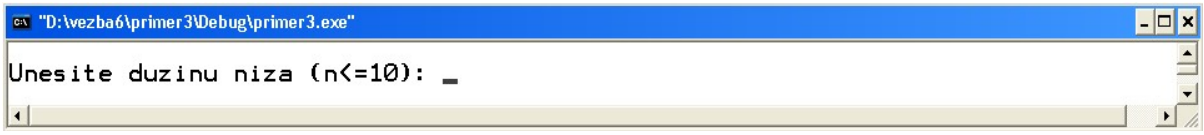
Анализа програма **primer6_3**

На почетку, у линији 8., програм декларише низ, дефинише максималну могућу дужину низа целих бројева (*MAX)* и иницијализује суму низа.

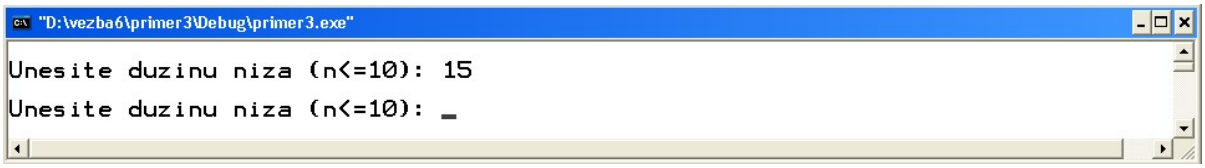
Програм тражи да се преко тастатуре зада дужина низа, прихвата са тастатуре дужину (*n*), а онда се врти у *do.. while* петљи (линије 11. и 12.) док се задаје дужина низа, ван траженог опсега, од 0 до *MAX* (слике 6.3 а) и 6.3 б)).

Када добије дужину низа у траженом опсегу (слика 6.3 в)), програм улази у for петљу (линије од 17. до 20.) и у свакој итерацији петље (од *i=0* до *i<n)* чита са тастатуре вредност елемента са индексом *i* и додаје ту вредност променљивој *suma* (линија 19.), која је на почетку иницијализована.

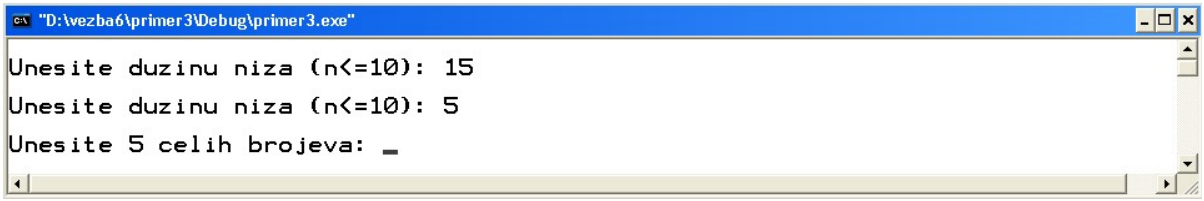
На крају, програм (из линије 21.) приказује на екрану средњу вредност задатог низа, добијену дељењем суме бројем елемената (слика 6.3 г)).



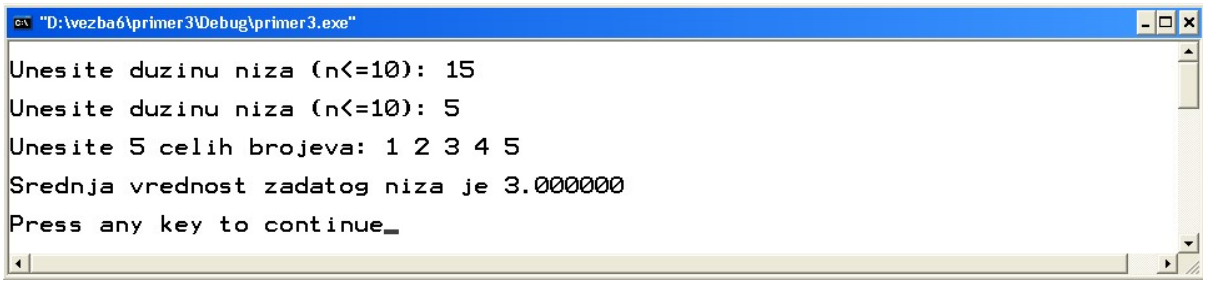
Слика 6.3 а) Излаз програма **primer6_3** (први део)



Слика 6.3 б) Излаз програма **primer6_3** (други део)



Слика 6.3 в) Излаз програма **primer6_3** (трећи део)



Слика 6.3 г) Излаз програма **primer6_3** (четврти део)

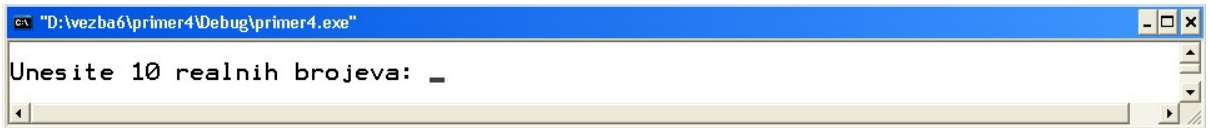
Изворни кôд програма **primer6_4**

```
1:      /*primer6_4.c*/
2:      /*Odredjivanje indeksa i vrednosti najveceg elementa*/
3:      /*zadatog niza realnih brojeva, duzine n (n<=50)*/
4:
5:      #include  <stdio.h>
6:      #define MAX 10
7:
8:      main( )
9:      {       int i, imax = 0;                /*pretpostavka: 1. element niza je najveci*/
10:             float niz[MAX];
11:
12:             printf("\nUnesite %d realnih brojeva: ", MAX);
13:             for(i=0; i< MAX; i++)
14:                     scanf("%f", &niz[i]);
15:
16:             for(i=1; i< MAX; i++)
17:                     if(niz[i] > niz[imax])          /*ako je niz[i] veci od prethodno*/
18:                             imax = i;               /* usvojenog najveceg elementa*/
19:
20:             printf("\nNajvecu vrednost ima %d. element niza “, imax+1);
21:             printf(“i to je: %f\n”, niz[imax]);
22:     }
```

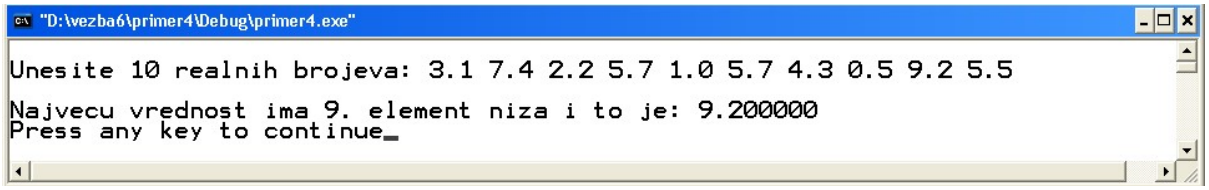
Анализа програма **primer6_4**

После декларације низа реалних бројева максималне дужине у линији 10. и иницијализације низа са тастатуре (слика 6.4 а)), програм одређује и приказује на екрану (слика 6.4 б)) редни број највећег елемента у задатом низу.

На почетку (линија 9.) је било претпостављено да је редни број највећег елемента 1. Ако се помоћу наредбе *if* (од линије 17.) нађе већи елемент, од првог у низу, програм узима његов редни број као тражени.



Слика 6.4 a) Излаз програма **primer6_4** (први део)



Слика 6.4 b) Излаз програма **primer6_4** (други део)

Изворни кôд програма **primer6_5**

```
1:      /*primer6_5.c*/
2:      /*Razvrstavanje slucajno generisanih celih brojeva (opseg od 0 do 10)*/
3:      /*u dva niza: parnih u jedan, a neparnih u drugi niz*/
4:
5:      #include  <stdio.h>
6:      #include <stdlib.h>
7:      #define MAX 10
8:
9:      main( )
10:     {       int i, j=0, k=0, x, a[MAX], b[MAX];
11:
12:             printf("\nSlucajno generisani niz brojeva:\n");
13:             for(i=0; i<MAX; i++)
14:             {       printf("%d ", x = rand( )/((double)RAND_MAX+1)*10);
15:                     if(x%2==0)
16:                             a[j++] = x;             /*a[j++] = x; ⇔ a[j] = x; j++;*/
17:                     else
18:                             b[k++] = x;             /*b[k++] = x; ⇔ b[k] = x; k++;*/
19:             }
20:
21:             printf("\n\nNiz parnih brojeva:\n");
22:             for(i=0; i<j; i++)
23:                     printf("%d ", a[i]);
24:
25:             printf("\n\nNiz neparnih brojeva:\n");
26:             for(i=0; i<k; i++)
27:                     printf("%d ", b[i]);
28:
29:             printf("\n\n");
30:     }
```

Анализа програма **primer6_5**

У програму овог примера користи се функција *rand()*, стандардне библиотеке *stdlib.h*. Резултат ове функције је случајно генерисани цео број, у опсегу од 0 до *RAND_MAX* (вредност ове стандардне константе је 32767 (*0x7fff*)). Број у траженом опсегу (од 0 до 10) може се добити од стандардног опсега (од 0 до *RAND_MAX)* помоћу обрасца у наредби, из линије 14:

x = rand( )/((double)RAND_MAX+1)∗10

Ево примера:

- ako je резултат функције rand( ) broj 1, број *x* добија целобројни део резултата израза *1/(double)32768∗10 = 0.000305*, а то је *0*,
- ako je резултат функције rand( ) broj 1000, број *x* добија целобројни део резултата израза *10000/(double)32768∗10 = 3.051758*, а то је *3*,
- ako je резултат функције rand( ) broj 32767, број *x* добија целобројни део резултата израза *30000/(double)32768∗10 = 9.155273*, а то је *9*.

У *for* петљи (линије од 13. до 19.) програм за 10 случајно генерисаних бројева ради следеће:

- ако је број паран, смешта га у први елемент низа *a* (*a[j]), j* је на почетку 0), а након тога инкрементира вредност индекса (*j*). Коришћен је скраћени запис за наредбе у линији 16., наредба

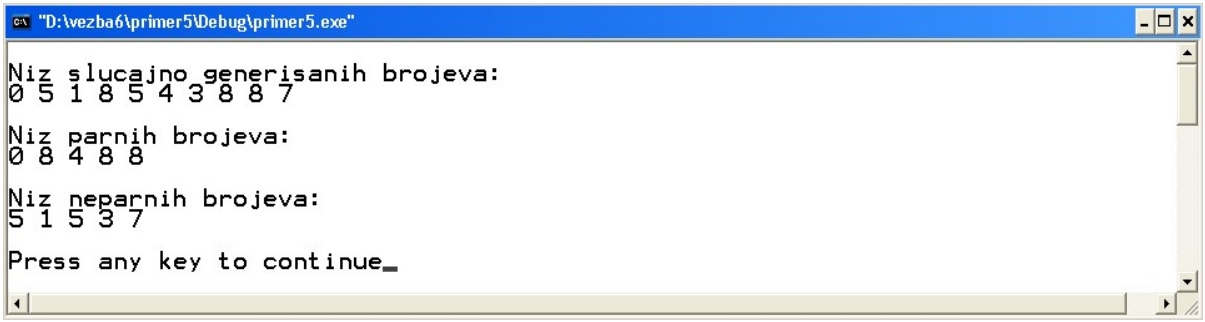
  a[j++] = x;

  је замена за две наредбе:

  a[j] = x;
  j++;

  Прво се елементу низа a[j] додели вредност броја x, а онда следи инкремент индекса *j,*
- ако број није паран смешта га у први елемент низа *b* (*b[k]), k* је на почетку 0), а након тога инкрементира вредност индекса (*k*). (И овде је коришћен скраћени запис за наредбе, који се уобичајно користи код оваквих случајева.)

У свим итерацијама поменуте петље редом се формирају елементи низова *a* и *b* (елемената низа *a* има *j*, а елемената низа *b* има *k).* У одговарајућим петљама (прва у линијама 22. и 23., а друга у линијама 26. и 27.) програм приказује формиране низове парних и непарних бројева (слика 6.5).



Слика 6.5 Излаз програма **primer6_5**

Изворни кôд програма **primer6_6**

```
1:      /*primer6_6.c*/
2:      /*Formiranje niza c od dva zadata niza celih brojeva a i b*/
3:      /*(svaki duzine 5) na sledeci nacin: a[0]+b[4],..a[4]+b[0]*/
4:
5:      #include  <stdio.h>
6:      #define MAX 5
7:
8:      main( )
9:      {       int i, a[MAX], b[MAX], c[MAX];
10:
11:             printf("\nUnesite prvih %d celih brojeva (elemente niza a): ", MAX);
12:             for(i=0; i<MAX; i++)
13:                     scanf("%d", &a[i]);
14:
15:             printf("\nUnesite drugih %d celih brojeva (elemente niza b): ", MAX);
16:             for(i=0; i<MAX; i++)
17:                     scanf("%d", &b[i]);
18:
19:             printf("\nVrednosti elemenata niza c: ");
20:             for(i=0; i<MAX; i++)
21:             {       c[i] = a[i] + b[MAX-1-i];
22:                     printf("%d ", c[i]);
23:             }
24:             printf("\n\n");
25:     }
```

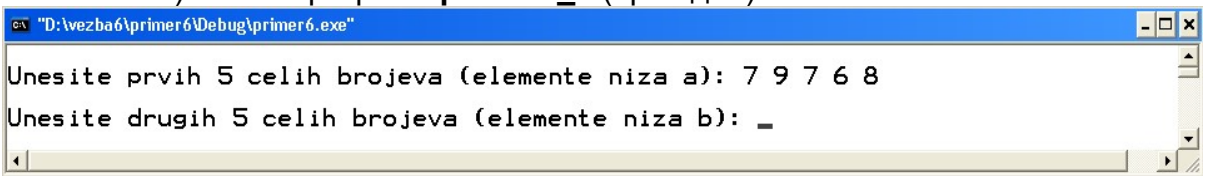
Анализа програма **primer6_6**

После декларације низова *a* и *b* (задати низови) и низа *c* (резултујући низ), у линији 9., програм има три петље.

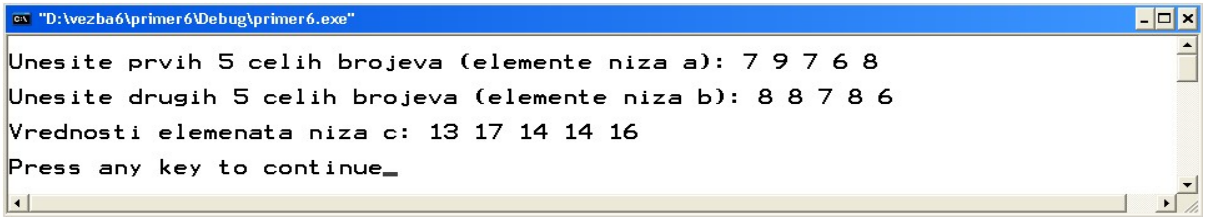
У првој петљи (линије 12. и 13.) прихвата са тастатуре вредности за елементе низа *a*, а у другој (линије 16. и 17.) за елементе низа *b*. Низови *a* и *b* су истих дужина *MAX*, у овом случају то је вредност 5 (слике 6.6 а) и 6.6 б)).

У трећој петљи (линије од 20. до 23.), програм у *MAX* итерација формира низ *c*. Вредност првог елемента низа *c* је збир вредности првог елемента низа *a* и последњег елемента низа *b*, други елемент низа *c* је збир вредности другог елемента низа *a* и претпоследњег елемента низа *b, ...* (слика 6.6 в)).

Слика 6.6 а) Излаз програма **primer6_6** (први део)



Слика 6.6 б) Излаз програма **primer6_6** (други део)



Слика 6.6 в) Излаз програма **primer6_6** (трећи део)

Изворни кôд програма **primer6_7**

```
1:      /*primer6_7.c*/
2:      /*Ciklicno pomeranje niza slucajno generisanih celih brojeva*/
3:      /*(opseg od 0 do 100), duzine 10, za jedno mesto u levo*/
4:
5:      #include  <stdio.h>
6:      #include <stdlib.h>
7:      #define MAX 10
8:
9:      main( )
10:     {       int i, pom, niz[MAX];
11:
12:             printf("\nNiz slucajno generisanih brojeva:\n");
13:             for(i=0; i<MAX; i++)
14:             {       printf("%d ",niz[i] = rand()/((double)RAND_MAX + 1)*100);
15:                     if(i%10 == 9)           /*stampa najvise 10 elemenata u liniji*/
16:                             printf("\n");
17:             }
18:
19:              /* Ciklicno pomeranje */
20:             pom = niz[0];
21:             for(i=0; i<MAX-1; i++)
22:                     niz[i] = niz[i+1];
23:             niz[MAX-1] = pom;
24:
25:             printf("\nCiklicno pomereni  niz za jedno mesto ulevo:\n");
26:             for(i=0; i<MAX; i++)
27:             {       printf("%d ", niz[i]);
28:                     if(i%10 == 9)
29:                             printf("\n");
30:             }
31:     }
```

Анализа програма **primer6_7**

После декларације низа целих бројева (дужине MAX, у овом случају то је вредност 10), програм иницијализује низ на случајно генерисане бројеве, резултате стандардне функције *rand()*. Бројеви треба да буду у опсегу од 0 до 100 и због тога је образац из примера 6.5 модификован. Помоћу наредби у линијама 15. и 16., програм омогућује приказ по 10 бројева у једној линији. Ако је индекс елемента 9, 19, 29, ... , после приказа броја прелази се у нови ред.

Следи циклично померање иницијализованог низа бројева за једно место улево (блок наредби од линије 20. до линије 23.). Променљива *pom* чува вредност првог елемента. У for петљи (линије 21. и 22.) програм реализује циклично померање помоћу наредбе *niz[i] = niz[i+1]*, где i има вредност од *0* до *MAX-1* (да се не би отишло ван граница низа). После петље, последњи елемент добија вредност *pom* и нови низ се приказује на екрану (слика 6.7).

Слика 6.7 Излаз програма **primer6_7**

Изворни кôд програма **primer6_8**

```
1:    /*primer6_8.c*/
2:    /*Nalazenje zadate kombinacije bitova u zadatom nizu elemenata*/
3:
4:    #include <stdio.h>
5:    #include <stdlib.h>
6:    #define MAX                10
7:    #define VEL_RECI           16
8:    #define MASKA              0x7
9:    #define OPSEG              0xff
10:
11:   main( )
12:   {       int i, niz[MAX], broj, poz_bita, nadjen = 0;
13:
14:           printf("\nSlucajno generisani brojevi(opseg od 0 do %d):\n\n", OPSEG);
15:           for(i=0; i<MAX; i++)
16:                   printf("%d ",niz[i] = rand()/((double)RAND_MAX + 1)*OPSEG);
17:
18:           do
19:           {       printf("\n\nUnesite kombinaciju od 3 bita (decimalni oblik): ");
20:                   scanf("%d",&broj);
21:           }while(broj<0 || broj > MASKA);
22:
23:           for(i=0; i<MAX; i++)
24:             for(poz_bita=1; poz_bita <= VEL_RECI; poz_bita++)
25:               if(((niz[i] >> (poz_bita-1)) & MASKA) == broj)
26:               {     nadjen = 1;
27:                     printf("\nZadata kombinacija je ");
28:                     printf("u %d. elementu, od %d. bita\n",i+1,poz_bita);
29:                     break;
30:               }
31:           if(!nadjen)
32:                   printf("\nZadata kombinacija nije nadjena.\n");
33:   }
```

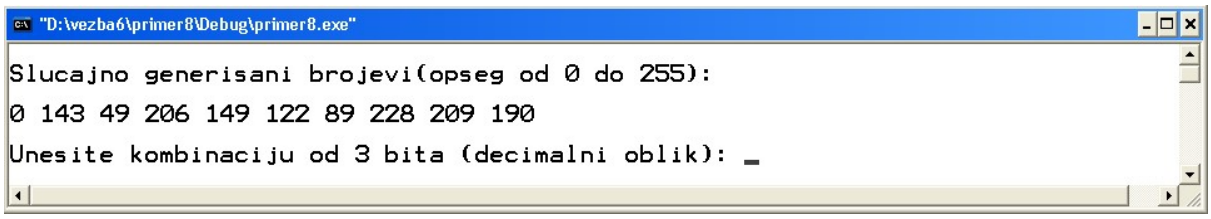
Анализа програма **primer6_8**

После декларације низа целих бројева дужине MAX, у овом случају то је вредност 10 (линија 12.), програм (као и претходни) иницијализује низ на случајно генерисане бројеве (линије од 14. до 16.), резултате стандардне функције *rand()*. Бројеви су у овом случају у опсегу два бајта, од 0 до 255 (0xff) (слика 6.8 а)).

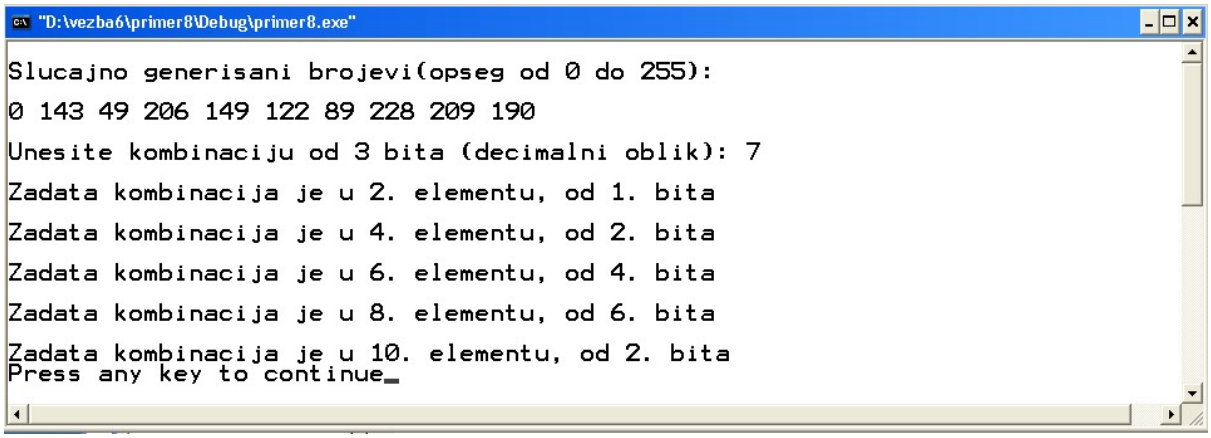
Програм даље тражи да се зада преко тастатуре децимални облик броја у опсегу 3 бита, од 0 до 7, (линије од 18. до 21.)

За сваки елемент задатог низа (*for* петља од линије 23.) програм пролази позиције свих 16 битова (за 2 бајта) редом (*for* петља од линије 24.) и помоћу *if* наредбе (од линије 25.) проверава да ли од позиције сваког бита постоји задата тробитна комбинација. Ово ради тако што помери сваки бит елемента низа *niz[i],* редом до позиције првог бита, а онда постави маску (0x7, то су све јединице на позицијама прва три бита) и пита да ли је добијени број исти као задати тробитни (линија 25.).

Програм региструје прво појављивање задате тробитне комбинације у сваком елементу низа, или пријављује одговарајућу поруку, ако комбинација није нађена у низу (слика 6.8 б)).



Слика 6.8 а) Излаз програма **primer6_8** (први део)



Слика 6.8 б) Излаз програма **primer6_8** (други део)

Изворни кôд програма **primer6_9**

```
1:      /*primer6_9.c*/
2:      /*Prikaz na ekranu elemenata zadate matrice (dvodimenzionog niza)*/
3:
4:      #include  <stdio.h>
5:      #define MAX1 2
6:      #define MAX2 5
7:
8:      main( )
9:      {       int i, j, niz[MAX1][MAX2] = {       10, 20, 30, 40, 50,
10:                                                 60, 70, 80, 90, 100         };
11:             for(j=0; j<MAX2; j++)
12:                     printf("\t[%d]", j);                  /*indeks kolone*/
13:             for(i=0; i<MAX1; i++)
14:             {       printf("\n\nniz[%d] ", i);            /*indeks reda*/
15:                     for(j=0; j<MAX2; j++)
16:                             printf("\t%d", niz[i][j]);
17:             }
18:
19:             printf("\n\n");
20:     }
```

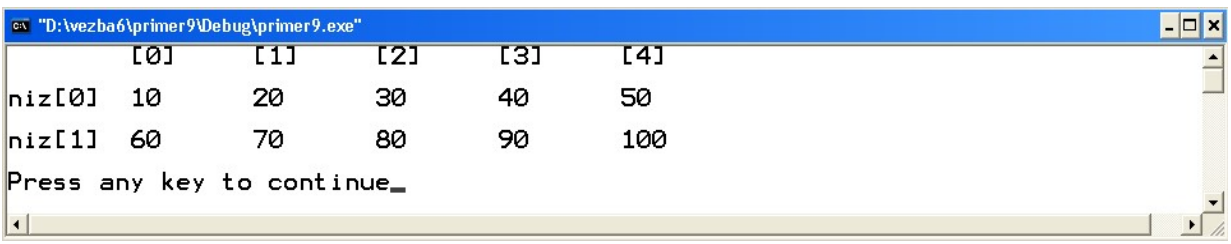
Анализа програма **primer6_9**

У програму овог примера је декларисан и унутар декларације иницијализован дводимензионални низ, или матрица, целих бројева, димензија *MAX1xMAX2* (2x5). То је низ од *MAX1* елемената, а сваки елемент је и сам низ од *MAX2* елемената.

За иницијализацију дводимензионалног низа важи све што важи и код једнодимензионалног низа, могућа је набрајањем вредности елемената између великих заграда (линије 9. и 10.).

У првој *for* петљи (линије 11. и 12.) програм исписује на екрану заглавље (индексе колона матрице (елемената у сваком од низова дводимензионалног низа)).

Помоћу друге *for* петље (линије од 13. до 17.) програм приказује у сваком од редова, прво индекс реда матрице (низа у дводимензионалом низу), а онда и сваки елемент тог низа *(niz[i][j],* од *i=0* до *i<MAX1* и од *j=0* до *j<MAX2)*. Ово реализује помоћу унутрашње *for* петље, у линијама 15. и 16. (слика 6.9).



Слика 6.9 Излаз програма **primer6_9**

Изворни кôд програма **primer6_10**

```
1:      /*primer6_10.c*/
2:      /*Zamena mesta dvema kolonama zadate matrice*/
3:
4:      #include  <stdio.h>
5:      #define MAX1  10
6:      #define MAX2  10
7:
8:      main( )
9:      {       int i, j, m, n, n1, n2, pom, niz[MAX1][MAX2];
10:
11:             do
12:             {       printf("\nDimenzije matrice (max %dx%d): ", MAX1, MAX2);
13:                     scanf("%dx%d", &m, &n);
14:             }while(m<1 || m>MAX1 || n<1 || n>MAX2);
15:
16:             for(i=0; i<m i++)
17:             {       printf("\nUnesite %d elemenata %d. reda matrice: ", n, i+1);
18:                     for(j=0; j<n; j++)
19:                             scanf("%d", &niz[i][j]);
20:             }
21:
22:             printf("\nZadata matrica:\n");
23:             for(i=0; i<m; i++)
24:             {       for(j=0; j<n; j++)
25:                             printf("%d ",niz[i][j]);
26:                     printf("\n");
27:             }
28:
29:             do
30:             {       printf("\nRedni brojevi kolona za zamenu: ");
31:                     scanf("%d%d", &n1, &n2);
32:             }while(n1<1 || n1>n || n2<1 || n2>n);
33:
34:             for(i=0; i<m; i++)
35:             {       pom = niz[i][n1-1]; niz[i][n1-1] = niz[i][n2-1]; niz[i][n2-1] = pom;
36:             }
37:
38:             printf("\nMatrica sa zamenjenim kolonama:\n");
39:             for(i=0; i<m; i++)
40:             {       for(j=0; j<n; j++)
41:                             printf("%d ",niz[i][j]);
42:                     printf("\n");
43:             }
44:     }
```

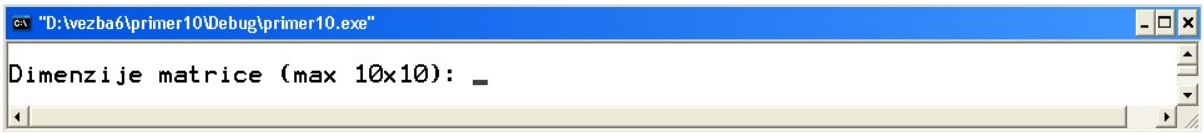
Анализа програма **primer6_10**

У програму овог примера, који мења места двема колонама матрице, није обављена иницијализација унутар декларације матрице(линија 9.), већ ван ње.

Програм прво тражи да се задају преко тастатуре димензије матрице у задатим опсезима бројева и на тражени начин. За ово користи *do.. while* петљу (линије од 11. до 14.), као и у претходним примерима (слика 6.10 а)). Број *m* треба да буде у опсегу од 1 до *MAX1*, а број *n* треба да буде у опсегу од 1 до *MAX2*. Све док је *m<1* или *m>MAX1* (*m* ван траженог опсега) или *n<1* или *n>MAX2* (n ван траженог опсега), програм се врти у петљи (линија 14.)

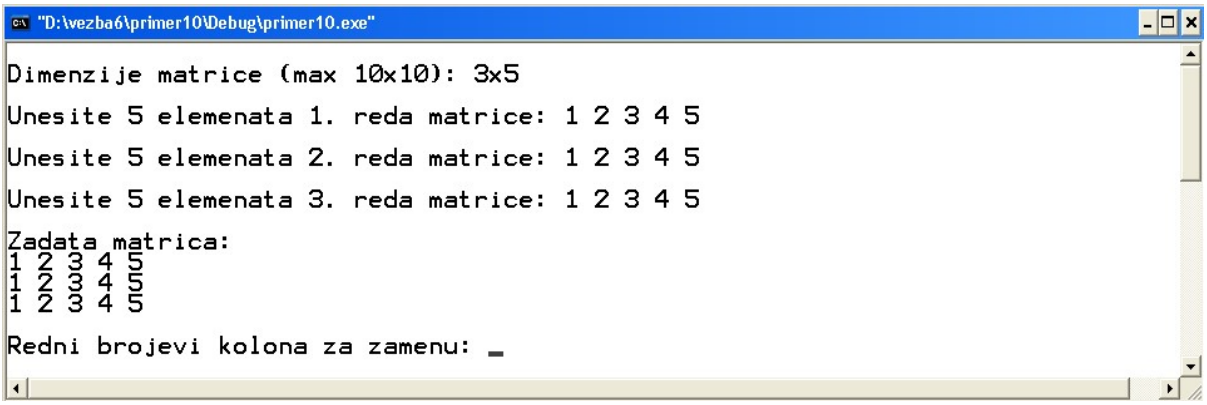
Када се задају димензије матрице (*m* и *n*) у траженим опсезима, програм у *for* петљама (по променљивама *i* и *j*) прихвата вредност за сваки елемент матрице: *&niz[i][j], од i=0 до i<m* и *од j=0 до j<n* (слика 6.10 б)). Помоћу једне *for* петље (линије од 23. до 27.) програм приказује на екрану задату матрицу бројева (слика 6.10 в)), а када прихвати са тастатуре редне бројеве колона за замену (блок наредби од 29. до 32. линије) и када заврши замену колона матрице, помоћу друге *for* петље (линије од 34. до 36.) приказује на екрану резултујућу матрицу (слика 6.10 г)).

У линији 32. постоји провера да ли су редни бројеви тражених колона за замену у дозвољеном опсегу димензија матрице. Све док је *n1<1* или *n1>n* или *n2<1* или *n2>n*, програм се врти у петљи. Замена колона матрице, са редним бројевима *n1* и *n2*, реализована је, опет, помоћу помоћне променљиве (*pom*). У сваком реду матрице (од *i=0* до *i<m*) замењује места елементима из колона са редним бројевима *n1* и *n2* (индекси елемената су *n1-1* и *n2-1*).

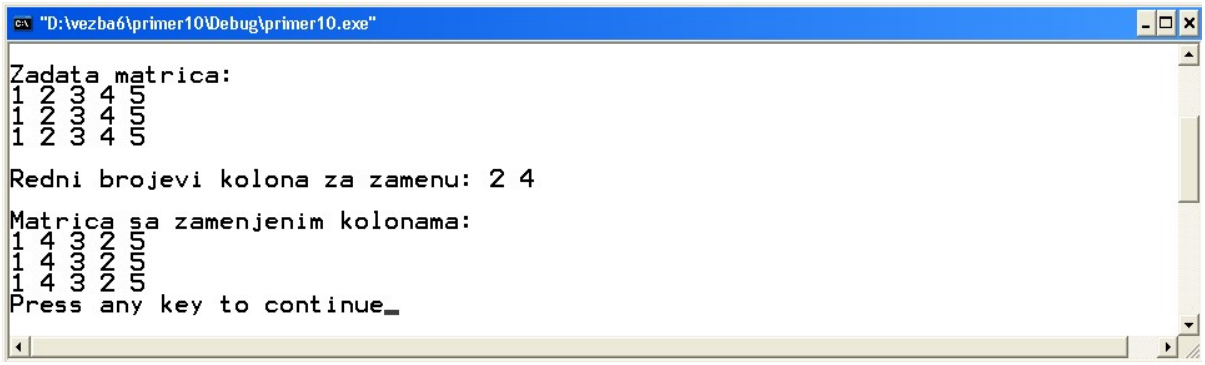


Слика 6.10 а) Излаз програма **primer6_10** (први део)

Слика 6.10 б) Излаз програма **primer6_10** (други део)



Слика 6.10 в) Излаз програма **primer6_10** (трећи део)



Слика 6.10 г) Излаз програма **primer6_10** (четврти део)

Практични савети

- ✓ Користити код низова увек индекс низа типа *int*, а не типа *long*.
- ✓ При коришћењу низова не нарушавати њихове дефинисане границе.

Евентуалне грешке

- ➢ Индекс елемента низа није исти као његов редни број, индекс почиње од 0 а не од 1.
- ➢ За дефинисану дужину низа *MAX*, индекс последњег елемента је *MAX-1* (*niz[MAX]* за дефинисани niz није елемент низа).
- ➢ Изостављање дужине низа при декларацији није дозвољено ако унутар декларације није иницијализован низ.

Задаци за припрему вежбе 6. у лабораторији

Задатак 1.

Написати програм на језику *C* који од 20 целих бројева, учитаних са тастатуре, формира низ од бројева који су у опсегу од 0 до 100, а онда циклично премешта елементе задатог низа за *m* места удесно *(m* је број задат преко тастатуре).

Задаци за вежбу 6. у лабораторији

Задатак 1.

Написати програм на језику *C* који у задатом низу целих бројева дужине *n (n<=20):*

а) одређује индексе највећег и најмањег елемента,
б) формира низ од задатог низа избацивањем највећег и свих њему једнаких елемената по апсолутној вредности.

Задатак 2.

Написати програм на језику *C* који у задатом низу целих бројева, дужине *n (n<=20):*

а) мења знак свим парним елементима са непарним индексима,
б) одређује број елемената којима је под а) измењен знак.

Задатак 3.

Написати програм на језику *C* који прихвата са тастатуре низ целих бројева дужине *n (n<=20)* и у зависности од задате опције са тастатуре *, L (l) ili D (d),* циклично премешта за *m* (број задат преко тастатуре) места улево или удесно елементе задатог низа. Предвидети приказ по 10 вредности у једном реду, као и одговарајуће поруке на екрану, за изабране непредвиђене опције.

Задатак 4.

Написати програм на језику *C* који на основу унетог броја поена на испиту за сваког студента (највише 50 њих) приказује на екрану извештај о просечној оцени и броју студената са оценама у задатим опсезима:

поени	оцена	поени	оцена
95..100	10	65..74	7
85..94	9	55..64	6
75..84	8	0....54	5

Задатак 5.

Написати програм на језику *C* који за задати низ целих бројева (сваки број у опсегу два бајта), дужине *n (n<=10)* одређује*:*

а) број битова у сваком елементу који имају вредност 1,
б) број елемената у којима су битови тестерасто уређени (1010...).

Задатак 6.

Написати програм на језику *C* који:

а) тражи прво појављивање задате комбинације од 4 бита у задатом низу целих бројева, дужине *n<=10* (сваки број у опсегу два бајта),
б) за нађену задату комбинацију помера битове тако да се тражена комбинација нађе на местима битова најмање тежине.

Вежба 7.

Знаковни низови у програмима на језику *C*

Декларација *stringova*

Врсте иницијализације *stringova*

***String* функције**

Употреба *stringova*

Низови *stringova*

Примери

Изворни кôд програма **primer7_1**

```
1:      /*primer7_1.c*/
2:      /*Prikaz na ekranu poruka sa predefinisanim sadrzajem*/
3:
4:      #include <stdio.h>
5:      #define MAX 30
6:
7:      main( )
8:      {       char ime[MAX+1];
9:              const char vezba7[] = "Znakovni nizovi u programima na jeziku C";
10:
11:             puts("Unesite svoje ime i prezime:");
12:             gets(ime);
13:
14:             printf("\nZdravo %s\n", ime);
15:
16:             printf("\nAnalizirate program vezbe 7.: \"%s\"\n\n", vezba7);
17:     }
```

Анализа програма **primer7_1**

Сваки програм обично садржи и прихватање са тастатуре низова знакова, резервисање простора у оперативној меморији за ове знакове, као и разне операције са овим знаковним низовима.

За сваки знак (било да је то слово, цифра или неки специјални знак) дефинисана је следећа синтакса у *C* језику: 'A', '5', '!'. Дакле сам знак се у програму пише између апострофа, а за чување једног знака у меморији користи се целобројни тип променљиве, карактер (*char*), који заузима један бајт. Иницијализација за поменуте променљиве изгледала би овако: char slovo = 'A'; char cifra = '5'; char znak = '!';

За знаковни низ (било да је то неки назив, као *"Visa elektrotehnicka skola u Beogradu"*, или адреса, као "*Vojvode Stepe 283*", или број телефона, као "*011-471-365*", користе се знаци навода и између њих се пише одговарајући знаковни низ. За чување било ког знаковног низа користи се низ карактера, који се зове *string. String* није основни *C* тип али се у програму на језику *C* третира као једна посебна јединица. Сваки *string* је неограничене дужине и из тог разлога се за чување знаковног низа дужине *n* резервише у меморији *string* дужине *n+1* (последњи карактер је *null* ('\0') и означава крај *stringa*). Развијен је велики број *string* функција (функција које раде са *stringovima*, читају их са улазних уређаја, шаљу на излазне уређаје, копирају, спајају, конвертују, ...).

У конкретном примеру за декларисани *string*, који се зове *ime* дужине *MAX+1* (линија 8.), *MAX* карактера је спремно да прихвати *MAX* знакова, док је последњи карактер резервисан за *null* карактер, који означава крај *string*а.

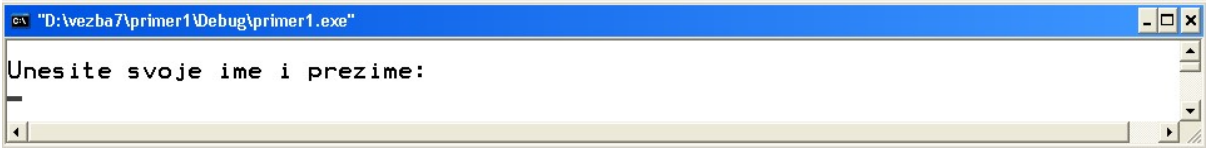
Дужина *string*а није неопходна ако декларација садржи и иницијализацију, као што је случај у линији 9. (константни *string vezba7* има свој иницијализовани текст (укупно 40 знакова у називу вежбе), што значи да се за овај *string* резервише *(40+1)*sizeof(char)* бајтова у оперативној меморији.

У линијама 11. и 12. написани су позиви стандардних функција *puts()* и *gets()* из библиотеке *stdio.h*, које се користе за приказ *string*а на екрану (слика 7.1 а)) и за читање *stringa* са тастатуре, респективно. Функцији *puts()* треба предати текст написан између наводника или име *string*а, чији садржај треба да прикаже на екрану, док функцији *gets()* треба предати име *string*а, који треба да прихвати и чува знаковни низ са тастатуре.

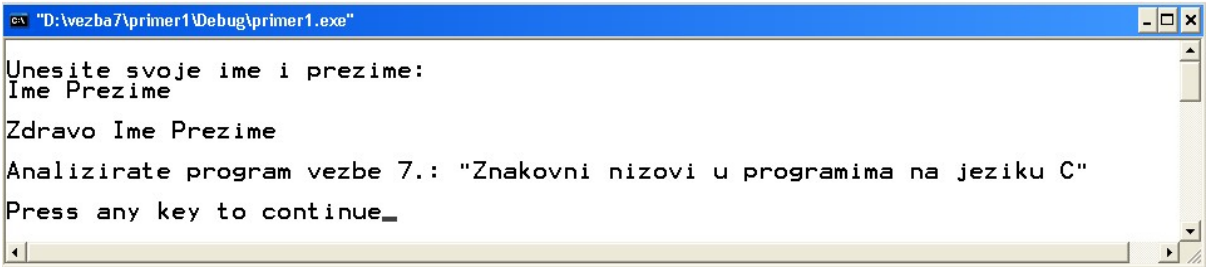
У овом случају функцији *puts()* се предаје као аргумент текст између наводника (линија 11.), који функција даље приказује на екрану (промпт програма) и одради прелаз у нови ред (који нема функција *printf*()). Функцији *gets()* предаје се као аргумент име *stringa* (*ime)* у линији 12., за MAX знакова са тастатуре.

Други начин приказа на екрану садржаја *stringa* је помоћу функције форматираног излаза *printf*(), која је овде искоришћена за приказ комбинације фиксног текста (*Zdravo*) и садржаја *stringa* (*ime)* из линије 14., а онда и за приказ комбинације фиксног текста и садржаја константног *stringa* из линије 16. (слика 7.1 б). Формат за *string* је у сваком случају %s.

Дакле, *string ime* је декларисан као низ променљивих типа карактер, спреман да прихвати један знак мање од своје дефинисане дужине, док је *string vezba7* константни низ карактера, чији је садржај при декларацији дефинисан и непроменљив даље у програму, а дужина број дефинисаних знакова, при иницијализацији, увећан за један.



Слика 7.1 а) Излаз програма **primer7_1** (први део)



Слика 7.1 б) Излаз програма **primer7_1** (други део)

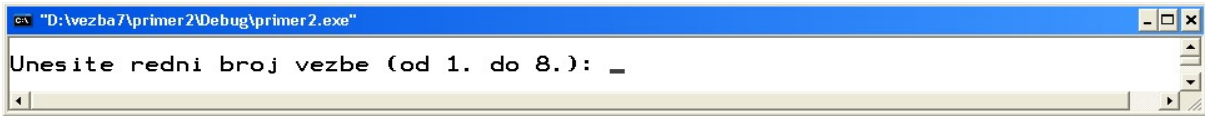
Изворни кôд програма **primer7_2**

```
1:      /*primer7_2.c*/
2:      /*Prikaz na ekranu naziva vezbe sa zadatim rednim brojem*/
3:
4:      #include <stdio.h>
5:      #define BROJ_VEZBI      8
6:      #define IME_VEZBE       80
7:      main( )
8:      {       int rbr;
9:              const char vezbe[BROJ_VEZBI][IME_VEZBE+1] =
10:               {"Osnovne komponente programa na jeziku C",
11:                "Osnovni tipovi podataka u jeziku C",
12:                "Operatori, izrazi i naredba selekcije u programima na jeziku C",
13:                "Naredbe ciklusa u programima na jeziku C",
14:                "Naredbe skokova i visestruke selekcije u programima na jeziku C",
15:                "Numericki nizovi u programima na jeziku C",
16:                "Znakovni nizovi u programima na jeziku C",
17:                "Sortiranje i pretrazivanje nizova u programima na jeziku C"};
18:             do
19:             {       printf("Unesite redni broj vezbe (od 1. do %d.):", BROJ_VEZBI);
20:                     scanf("%d", &rbr);
21:             }while(rbr<1 || rbr>BROJ_VEZBI);
22:
23:             printf("\nVezba broj %d. je:\n\n\"%s\"\n\n", rbr, vezbe[rbr-1]);
24:     }
```

Анализа програма **primer7_2**

У програму овог примера је декларисан (од линије 9.) константни дводимензиони низ типа *char*. То је низ од *BROJ_VEZBI stringova*, где је сваки *string* низ од *IME_VEZBE+1* карактера (*IME_VEZBE* карактера за само име и један *null* карактер). У линијама од 10. до 17. је и иницијализован сваки од константних *stringova* низа (дефинисана су имена вежби).

Програм приказује на екрану (из линије 23.) име вежбе са задатим редним бројем са тастатуре (слика 7.2 а)) у *do..while* петљи (линије од 18. до 21.). Резултат програма може се видети на слици 7.2 б).



Слика 7.2 а) Излаз програма **primer7_2** (први део)

Слика 7.2 б) Излаз програма **primer7_2** (други део)

Изворни кôд програма **primer7_3**

```
1:    /*primer7_3.c*/
2:    /*Formiranje izvestaja o brojevima zadatih pojedinih ocena*/
3:    #include <stdio.h>
4:    #include <stdlib.h>
5:    #include <ctype.h>
6:    #define MIN 0
7:    #define MAX 10
8:    #define BROJ_ZNAKOVA 20
9:    main( )
10:   {       int i, ocena, rezultati[MAX+1];
11:           char bafer[BROJ_ZNAKOVA +1];
12:
13:           /*Inicijalizacija*/
14:           for(i=MIN; i<=MAX;i++)
15:                   rezultati[i] = 0;
16:
17:           /*Prikupljanje podataka*/
18:           printf("\nUnesite ocenu u opsegu od %d do %d", MIN, MAX);
19:           printf("\nPotvrdite svaki podatak tasterom Enter");
20:           printf("\nUnosenje se zavrsava izborom znaka koji nije cifra\n\n");
21:
22:           while(1)
23:           {       /*Prihvatanje ulaznog podatka*/
24:                   printf("Ocena: ");
25:                   gets(bafer);
26:
27:                   /*Provera da li je komanda za kraj*/
28:                   /*(znak koji nije cifra)*/
29:                   if(!(isdigit(bafer[0])))
30:                   {       putchar('\n');
31:                           break;          }
32:
33:                   /*Konverzija ulaznog znakovnog niza u ceo broj*/
34:                   ocena = atoi(bafer);
35:
36:                   /*Provera da li je podatak u dozvoljenom opsegu*/
37:                   if(ocena >= MIN && ocena <=MAX)
38:                           rezultati[ocena]++;
39:                   else
40:                   {       printf("GRESKA: podatak je van dozvoljenog opsega\n");
41:                           continue;       }
42:           }
43:           /*Izvestaj o rezultatima*/
44:           printf("Rezultati:\n\nOcena\tBroj pojavljivanja\n");
45:           for(i=MIN;i<=MAX;i++)
46:                           printf("%d\t%d\n", i, rezultati[i]);
47:   }
```

Анализа програма **primer7_3**

На почетку програма овог примера иницијализован је (у линији 10.) целобројни низ *rezultati*, који ће бити искоришћен за чување бројева појединих оцена у одређеном опсегу (у овом случају од 0 до 10): први елемент низа ће чувати број задатих оцена 0, други елемент низа број задатих оцена 1,.... У линији 11. декларисан је *string bafer* дужине *BROJ_ZNAKOVA +1*.

У *for* петљи (линије 14. и 15.) следи неопходна иницијализација низа *rezultati*.

После промпта програма (линије 18., 19. и 20.) следи *while(1)* петља:

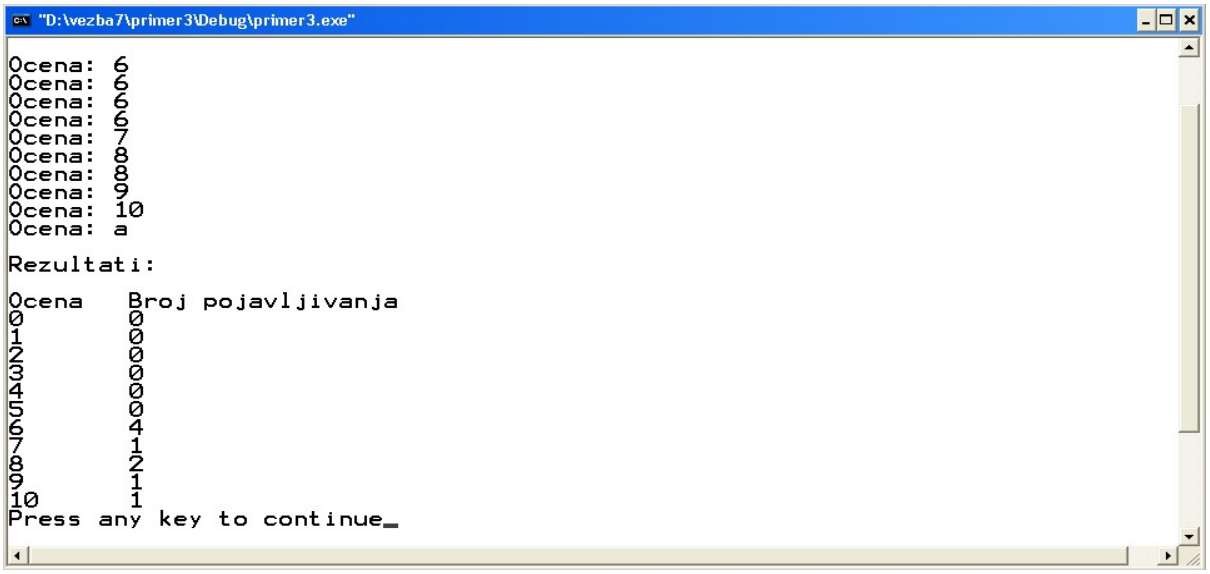
– помоћу функције *gets()* програм прихвата са тастатуре знаковни низ и смешта га у *string bafer* (линија 25.),

– ако први знак *stringa* није цифра *(!(isdigit(bafer[0])* где је *isdigit()* функција стандардне библиотеке *ctype.h*) програм то прихвата као знак за крај, извршава се наредба break; (линија 31.) за излаз из петље,

– ако то није био случај, програм иде даље и помоћу функције *atoi(),* из стандардне библиотеке *stdlib.h*, цифре из *stringa bafer* конвертује у цео број и тај број смешта у променљиву *ocena*. Ако је оцена у дозвољеном опсегу (линија 37.), програм региструје одређену оцену и инкрементира садржај елемента низа *rezultati*, који се користи да чува број ове оцене (линија 38.). У супротном, следи прелаз на следећу итерацију *while(1)* петље (конвертовање следећег знаковног низа у број).

На слици 7.3 а) приказан је промпт програма, а на слици 7.3 б), после одговарајућег броја итерација *while(1)* петље, до првог податка који није цео број, приказ резултата (задате оцене и бројеви њиховог појављивања).

Слика 7.3 а) Излаз програма **primer7_3** (први део)



Слика 7.3 б) Излаз програма **primer7_3** (други део)

Изворни кôд програма **primer7_4**

```
1:    /*primer7_4.c*/
2:    /*Kalkulator sa 4 osnovne aritmeticke operacije*/
3:    #include <stdio.h>
4:    #include <stdlib.h>
5:    #include <ctype.h>
6:    #define MAX_BAF 20        /*maksimalna duzina ulaznog podatka je 20*/
7:    main( )
8:    {     int znak, nepoznat = 0;
9:          double broj1, broj2;                  /*operandi su realni brojevi*/
10:         char bafer[MAX_BAF+1];
11:
12:         while(1)
13:         {     /*Prihvatanje brojeva i operatora*/
14:               printf("\nPrvi broj: ");  gets(bafer);
15:               if(!isdigit(bafer[0]) && bafer[0]!='-')
16:               {     putchar('\n');
17:                     break;          }
18:               broj1 = atof(bafer);
19:
20:               printf("\nDrugi broj: ");  gets(bafer);
21:               if(!isdigit(bafer[0]) && bafer[0]!='-')
22:               {     putchar('\n');
23:                     break;                  }
24:               broj2 = atof(bafer);
25:
26:               printf("\nOperator: ");
27:               znak = getchar(); getchar();        /*znak za operator i novi red*/
28:
29:               /*Izracunavanje i prikaz rezultata*/
30:               switch(znak)
31:               {case '+':     printf("\nRezultat: %f\n", broj1+broj2);
32:                                    break;
33:               case '-':      printf("\nRezultat: %f\n\n", broj1-broj2);
34:                                    break;
35:               case '*':      printf("\nRezultat: %f\n", broj1*broj2);
36:                                    break;
37:               case '/':      if(broj2!=0)
38:                                          printf("\nRezultat: %f\n", broj1/broj2);
39:                                    else
40:                                          printf("\nGRESKA:deljenje nulom\n");
41:                                    break;
42:               default:       printf("\nNepoznat operator\n"); nepoznat = 1;
43:                                    break;                                    }
44:               if(nepoznat)
45:                     break;
46:         }
47:   }
```

Анализа програма **primer7_4**

И у овом примеру програм прихвата са тастатуре податке, али сада проверава за сваки податак да ли је број или оператор.

Програм очекује унос редом два операнда и оператора (једног од основних аритметичких (за сабирање, одузимање, множење или дељење) и даје резултат. Ово изводи у *while(1)* петљи (од линије 12.).

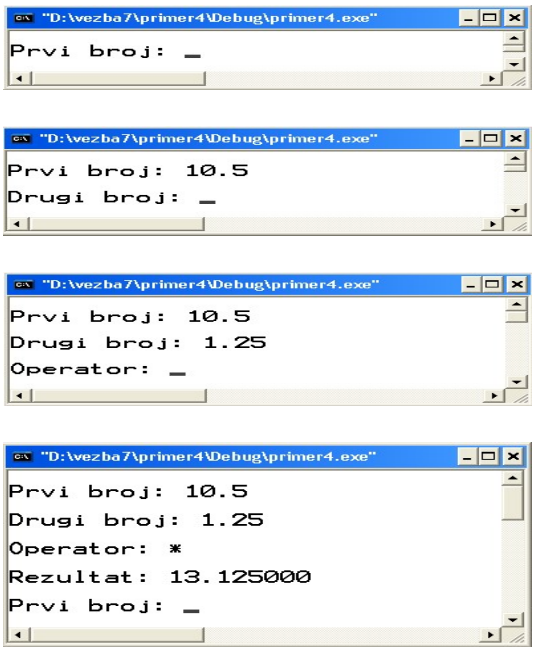
Из линије 14. програм прихвата први податак, смешта га у *string bafer* и ако први карактер садржаја овог *stringa* није цифра (за позитиван број) или предзнак - (за негативан број) следи прекид петље *while(1)*. У супротном извлачи први реалан број из податка помоћу функције *atof()*.

Из линије 20. програм прихвата други податак и понавља поступак за други реалан број.

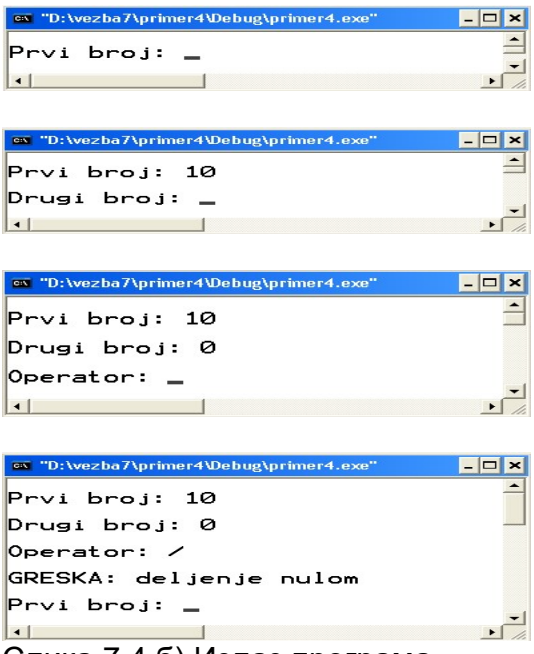
Из линије 27. програм прихвата трећи податак и очекује да је то један од тражених оператора. При томе, први позив функције *getchar()* прихвата изабрани оператор и смешта у променљиву *znak*. Други позив функције *getchar()* је ту да би прихватио знак за прелаз у нови ред од тастера *Enter*, притиснутом по изабраном оператору (да овај знак не би заостао и правио проблеме даље).

У зависности од изабраног оператора (*switch(znak)* од линије 30.) програм приказује на екрану резултате примене задатог оператора над задатим операндима (слика 7.4 а)). У случају изабраног непознатог оператора (*default* :случај у линији 42.) програм сетује статус *nepoznat* и прекида извршавање *while(1)* петље.

Код дељења је избегнуто дељење нулом и предвиђена пријава грешке у том случају (слика 7.4 б)).



Слика 7.4 а) Излаз програма **primer7_4** (први део)



Слика 7.4 б) Излаз програма **primer7_4** (други део)

Изворни кôд програма **primer7_5**

```
1:      /*primer7_5.c*/
2:      /*Osnovne operacije nad stringovima*/
3:      #include  <stdio.h>
4:      #include <string.h>
5:      #define MAX 80
6:
7:      main( )
8:      {       char string1[MAX+1], string2[MAX/2+1];
9:              int l1,l2, x;
10:
11:             do
12:             {       printf("\nPrvi znakovni niz(duzine n<=%d):\n",MAX/2);
13:                     gets(string1);
14:                     l1 = strlen(string1);
15:             }while(l1>MAX/2);
16:
17:             do
18:             {       printf("\nDrugi znakovni niz(duzine n<=%d):\n",MAX/2);
19:                     gets(string2);
20:                     l2 = strlen(string2);
21:             }while(l2>MAX/2);
22:
23:             /*Poredjenje duzina stringova*/
24:             if (l1 == l2)
25:                     puts("\nStringovi su iste duzine.");
26:             else if (l1>l2)
27:                     puts("\nPrvi string je duzi.");
28:             else
29:                     puts("\nDrugi string je duzi.");
30:
31:             /*Poredjenje sadrzaja stringova*/
32:                     x = strcmp(string1,string2);
33:             if (x==0)
34:                     puts("Sadrzaji stringova su isti.");
35:             else if (x>0)
36:                     puts("Prvi string je \"veci\".");
37:             else
38:                     puts("Drugi string je \"veci\".");
39:
40:             /*Spajanje sadrzaja stringova*/
41:             strcat(string1, string2);
42:             printf("\nRezultujuci string: %s",string1);
43:
44:             /*Kopiranje sadrzaja jednog stringa u drugi*/
45:             strcpy(string1,"Nova recenica.");
46:             printf("\nNovi sadrzaj rezultujuceg stringa: %s\n\n",string1);
47:     }
```

Анализа програма **primer7_5**

У овом програму су примери основних операција са *stringovima*.

У линији 8. декларисани су *stringovi*. *String1* може да прими *MAX* знакова, а *string2 MAX/2* знакова.

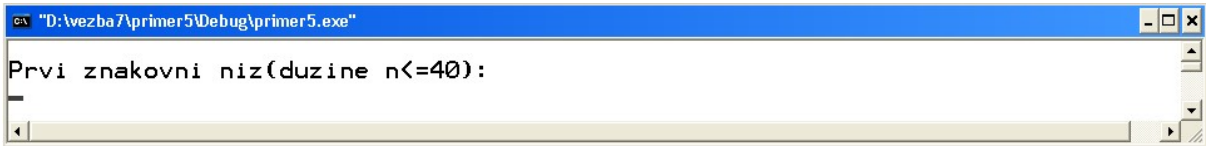
У првој do..while петљи (линије од 11. до 15.) програм се врти, док се не зада први *string* тражене дужине (*MAX/2), а у* другој do..while петљи (линије од 17. до 21.), док се не зада други *string* исте дужине (*MAX/2)* слике7.5 а) и 7.5 б)

У линијама од 24. до 29. програм пореди дужине задатих *stringova* (садржаје променљивих *l1* и *l2*, којима су претходно додељене дужине *stringova*, одређене помоћу функције *strlen()* из стандардне библиотеке *string.h*) и приказује на екрану одговарајуће поруке.

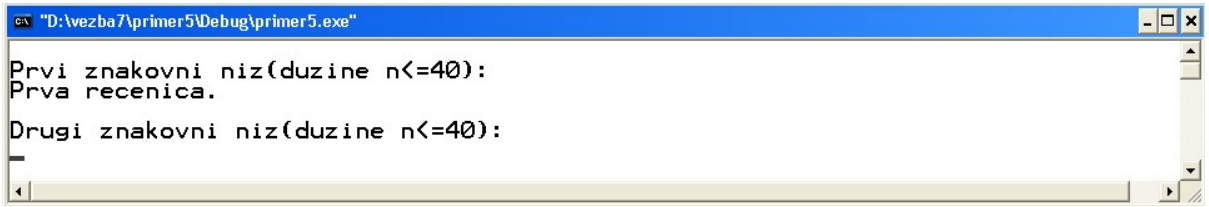
У линијама од 32. до 38. програм пореди садржаје од два задата *stringa*, помоћу функције *strcmp()* из стандардне библиотеке *string.h.* Ако је резултат ове функције 0, садржаји *stringova* су исти, ако је мањи од 0, први *string* је ”мањи” (по табели *ASCII* кôдова *“Astring”* је ”мањи” од *”Bstring”*), а ако је резултат већи од 0, први *string* је ”већи” (по табели ASCII кôдова *“stringC”*, је ”већи” од *“stringB”).* Свако слово, цифра и знак има свој кôд у ASCII табели, *A* има кôд 65, *B* кôд 66, *a* има кôд 98,... (табела 11. у Прилогу). Функција *strcmp()* пореди један по један знак (*ASCII* кôд) од *stringova*.

Помоћу функције *strcat(string1, string2)* програм дописује садржај *string2* у продужетку *string1* и због тога је дужина резултујућег *stringa* (*string1*) требала да буде довољна да прихвати ове садржаје (слика 7.5 в)).

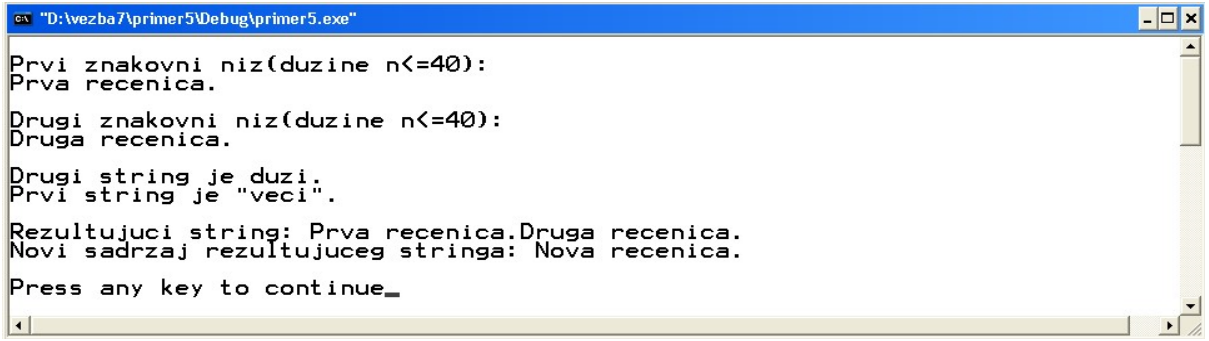
Помоћу позива функције *strcpy(string1, “Novi string”)* или *strcpy(string1, string2)*, програм копира *“Novi string”* или садржај *string2* од почетка резултујућег *stringa string1* (преко евентуалног постојећег садржаја).



Слика 7.5 а) Излаз програма **primer7_5** (први део)



Слика 7.5 б) Излаз програма **primer7_5** (други део)



Слика 7.5 в) Излаз програма **primer7_5** (трећи део)

Изворни кôд програма **primer7_6**

```
1:      /*primer7_6.c*/
2:      /*Odredjivanje najduzeg i najkraceg reda od 20 zadatih redova teksta*/
3:      /*STOP kod pri zadavanju redova teksta moze biti prazan red*/
4:
5:      #include  <stdio.h>
6:      #include <string.h>
7:      #define BROJ 20
8:      #define DUZINA 80
9:
10:     main()
11:     {       char stringovi[BROJ][DUZINA+1];
12:             char string_max[DUZINA+1], string_min[DUZINA+1];
13:             int i,imax,imin,l, lmin,lmax;
14:
15:             printf("\nUnesite 1. red teksta:\n");
16:             gets(stringovi[0]);
17:             imax = imin = 0;
18:             lmax = lmin = strlen(stringovi[0]);
19:             strcpy(string_max, stringovi[0]);
20:             strcpy(string_min, stringovi[0]);
21:
22:             /*Prihvatanje niza stringova*/
23:             for(i=1; i<BROJ; i++)
24:             {       printf("\nUnesite %d. red teksta:\n",i+1);
25:                     gets(stringovi[i]);
26:
27:                     if((l = strlen(stringovi[i])) == 0)
28:                             break;
29:
30:                     /*Trazenje najduzeg i najkraceg stringa u nizu*/
31:                     if(l > lmax)
32:                     {       imax = i;
33:                             lmax = l;
34:                             strcpy(string_max, stringovi[i]);
35:                     }
36:                     if(l < lmin)
37:                     {       imin = i;
38:                             lmin = l;
39:                             strcpy(string_min, stringovi[i]);
40:                     }
41:             }
42:
43:             printf("\nNajduzi od zadatih redova (%d karaktera)", lmax);
44:             printf("je red broj %d. :\n%s\n", imax+1, string_max);
45:             printf("\nNajkraci od zadatih redova (%d karaktera)", lmin);
46:             printf("je red broj %d. :\n%s\n\n", imin+1, string_min);
47:     }
```

Анализа програма **primer7_6**

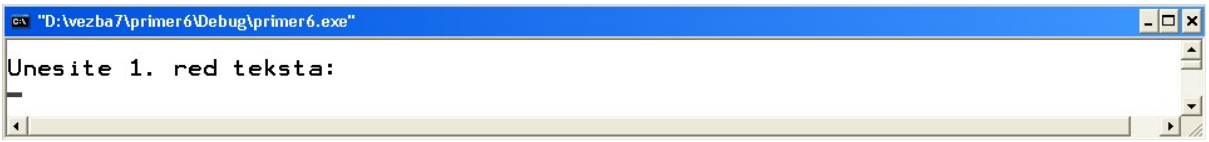
Програм у овом примеру прво декларише низ *stringova* (*stringovi*) да би могао да прихвати са тастатуре низ редова текста (низ низова знакова), као и потребне *stringove*, *string_max* и *string_min*, који треба да чувају најдужи и најкраћи од задатих *stringova*, респективно.

После учитаног првог низа знакова са тастатуре (линије 15. и 16.), програм претпоставља да је овај први ред истовремено најдужи и најкраћи задати ред, тако да је индекс *stringa* најдужег реда исти као и индекс *stringa* најкраћег реда и то је 0 (линија 17.), да је дужина најдужег *stringa* иста као и дужина најкраћег *stringa* и то је дужина првог *stringa* у низу (линија 18.) и да је садржај најдужег *stringa* исти као и садржај најкраћег *stringa* и то је садржај првог *stringa* у низу (линије 19. и 20.).

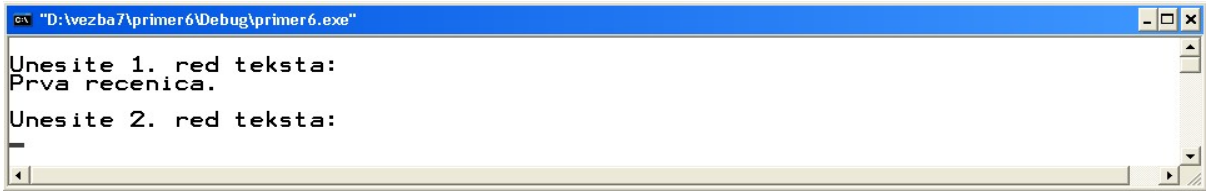
Даље, у for петљи (у линијама од 23. до 41.) програм:

- прихвата један *string* у низу (*stringovi[i])*,
- ако је задат празан *string*,прекида *for* петљу (линије 27. ,28.).
- ако је дужина *stringa* (*stringovi[i])* већа од претходно одређене највеће дужине *stringa* (*lmax*), ажурира све податке везане за *string* највеће дужине: *imax, lmax, string_max* (линије од 31. до 35.),
- ако је дужина *stringa* (*stringovi[i])* мања од претходно одређене најмање дужине *stringa* (*lmin*), ажурира све податке везане за *string* најмање дужине: *imin, lmin, string_min* (линије од 36. до 40.).

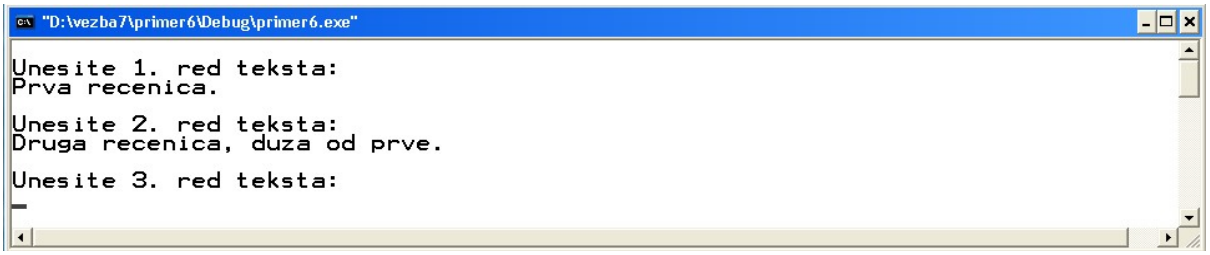
На сликама 7.6 а), 7.6 б), 7.6 в) и 7.6 г) се може видети да програм пружа промпт за унос редова текста, све док се не зада празан ред (ако нема празног реда, онда прихвата, на почетку програма дефинисани број редова). На слици 7.6 д) могу се видети резулатати које приказује програм на екрану..



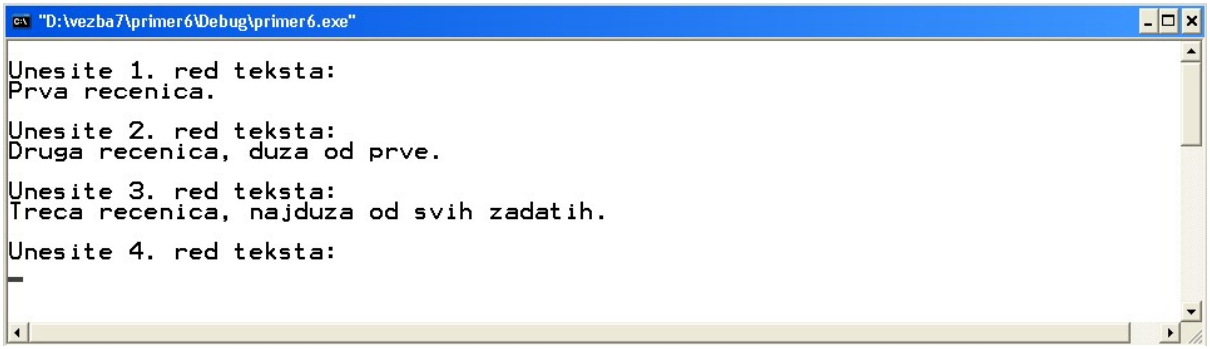
Слика 7.6 а) Излаз програма **primer7_6** (први део)



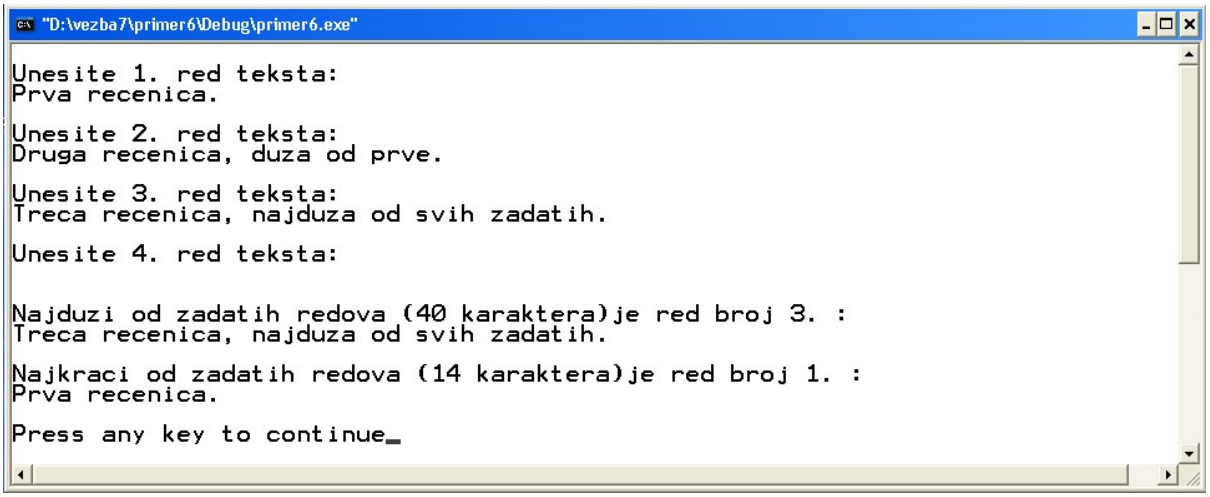
Слика 7.6 б) Излаз програма **primer7_6** (други део)



Слика 7.6 в) Излаз програма **primer7_6** (трећи део)



Слика 7.6 г) Излаз програма **primer7_6** (четврти део)



Слика 7.6 д) Излаз програма **primer7_6** (пети део)

Практични савети

- ✓ При формалној декларацији *stringa* дефинисати његову дужину за један већу од дужине знаковног низа који треба да чува (због *null* завршетка).
- ✓ При употреби двоструких знакова навода за иницијализацију *stringa,* аутоматски се додаје завршетак *null* (иницијализација *stringa* знак по знак захтева експлицитно додавање *null* завршетка).

Евентуалне грешке

- ➢ Коришћење оператора доделе, као и релационих оператора код *stringova* не пролази, јер је *string* низ. За декларисане *stringove*:

  char string1[20], string2[20];

Нису дозвољени доле написани изрази	Дозвољени су доле написани изрази
string1 = string2	*strcpy(string1, string2)*
if(string1==string2)	*if(strcmp(string1, string2)==0)*
if(string1<string2)	*if(strcmp(string1, string2)<0)*

Задаци за припрему вежбе 7. у лабораторији

Задатак 1.

Написати програм на језику *C* који прихвата са тастатуре један ред текста и приказује на екрану поруку за унос другог реда текста, све док се не унесе текст различит од првог задатог или укупно 5 редова. Онда програм приказује на екрану текст од спојених *stringova* (који садржи све задате текстове редом).

Задаци за вежбу 7. у лабораторији

Задатак 1.

Написати програм на језику *C* који:

а) прихвата са тастатуре ред текста и ако је дужина задатог реда већа од 10, копира га на екран,

б) понавља поступак, описан под а), све док се не унесе ред који има само једну реч *“Kraj”*.

Задатак 2.

Написати програм на језику *C* који:

а) прихвата са тастатуре ред текста и сваки празан знак задатог текста копира у знак црта подвучена (_),

б) понавља поступак, описан под а), све док се не зада празан ред.

Задатак 3.

Написати програм на језику *C* који прихвата са тастатуре 10 реченица (реченица је низ знакова који се завршава знаком тачка) и копира их на екран на следећи начин:

а) прво слово сваке речи у реченици копира у исто велико слово (реч је низ знакова између два бела знака),

б) сваку реченицу приказује на екрану у посебном реду.

Задатак 4.

Написати програм на језику *C* који

а) прихвата са тастатуре списак имена (име и презиме за сваку особу у списку) и приказује на екрану следеће:

1. *Unos*
2. *Izmene*
3. *Prikaz*
4. *Izlaz*

све док се не изабере опција 4,

б) за изабрану опцију 1. прихвата са тастатуре име и, ако је различито од свих до тада унетих, додаје га низу имена,

в) за изабрану опцију 2. и редни број имена у списку приказује име на екрану и прихвата са тастатуре ново име,

г) за изабрану опцију 3. приказује на екрану постојећи списак имена .

Вежба 8.

Сортирање и претраживање низова у програмима на језику *C*

Сортирање нумеричких низова

Претраживање нумеричких низова

Сортирање низова *stringova*

Примери

Изворни кôд програма **primer8_1**

```
1:      /*primer8_1.c*/
2:      /*Sortiranje svakog zadatog niza celih brojeva*/
3:      /*u neopadajuci poredak, METODOM IZBORA*/
4:
5:      #include <stdio.h>
6:      #define MAX 10
7:
8:      main()
9:      {       int i, j, n, pom, niz[MAX];
10:
11:             while(1)
12:             {       /*Zadavanje duzine niza*/
13:                     printf("\n\nUnesite broj elemenata niza n (n<=%d): ",MAX);
14:                     scanf("%d", &n);
15:
16:                     if(n<1 || n>MAX)
17:                             break;
18:
19:                     /*Zadavanje elemenata niza*/
20:                     printf("Unesite celobrojne vrednosti elemenata niza: ");
21:                     for(i=0; i<n; i++)
22:                             scanf("%d", &niz[i]);
23:
24:                     /*Prikaz na ekranu originalnog niza*/
25:                     printf("\nOriginalni niz:\t");
26:                     for(i=0; i<n; i++)
27:                             printf("%d ",niz[i]);
28:
29:                     /*Sortiranje niza u neopadajuci poredak*/
30:                     for(i=0; i<n-1; i++)
31:                             for(j=i+1; j<n; j++)
32:                                     if(niz[i] > niz[j])
33:                                     {       pom = niz[i];
34:                                             niz[i] = niz[j];
35:                                             niz[j] = pom;
36:                                     }
37:
38:                     /*Prikaz na ekranu sortiranog niza*/
39:                     printf("\nSortirani niz:\t");
40:                     for(i=0; i<n; i++)
41:                             printf("%d ",niz[i]);
42:             }
43:     }
```

Анализа програма **primer8_1**

Сортирање низова има велики значај у програмима, било која врста података обавезно се сортира и онда једноставније користи. У примерима који следе објашњени су неки најједноставнији алгоритми за сортирање низова као и њихове реализације на језику *C*.

Програм у овом примеру сортира низ целих бројева, задат преко тастатуре (линије од 12. до 22.), у неопадајући поредак. Ово значи да програм поставља елементе низа у поредак: $niz[0] \leq niz[1] \leq niz[2] \leq \ldots \leq niz[MAX-1]$, који дозвољава да два суседна елемента имају исте вредности. за разлику од растућег поретка елемената низа: $niz[0] < niz[1] < niz[2] < \ldots < niz[MAX-1]$. Нерастући поредак елемената је: $niz[0] \geq niz[1] \geq niz[2] \geq \ldots \geq niz[MAX-1]$, где су опет дозвољене исте вредности суседних елемената, док је опадајући поредак елемената истог низа: $niz[0] > niz[1] > niz[2] > \ldots > niz[MAX-1]$.

У овом програму коришћена је метода избора. Програм бира сваки пар елемената низа, пореди им вредности и евентуално (ако нису у траженом поретку) мења места.

Ако вредности пара елемената нису у траженом, неопадајућем поретку, већ у опадајућем (линија 32.) програм ради замену (линије 33. до 36.). У супротном оставља елементе како јесу.

Део кôда од 30. до 36. линије реализује само сортирање:

- у спољашњој петљи (од линије 30.)дефинисани су пролази кроз низ, од првог *(i=0)* до претпоследњег *(i=n-2)* елемента,
- у унутрашњој петљи (од линије 31.) реализује се поређење и евентуална замена места, у сваком пролазу кроз низ.

  - У првом пролазу кроз низ (*i=0)*:
    за *j=i+1=1*, ако је *niz[0] > niz[1]*, следи замена места,
    за *j=i+2=2*, ако је *niz[0] > niz[2],* следи замена места,
    :
    :
    за *j=n-1*, ако је *niz[0] > niz[n-1],* следи замена места.
    После овог пролаза на месту првог елемента у низу је вредност која није већа од вредности осталих елемената.

  - У другом пролазу кроз низ (*i=1)*:
    за *j=i+1=2*, ако је *niz[1] > niz[2]*, следи замена места,
    за *j=i+2=3*, ако је *niz[1] > niz[3]*, следи замена места,
    :
    :
    за *j=n-1*, ако је *niz[1] > niz[n-1],* следи замена места.
    После другог пролаза, на месту другог елемента у низу је вредност која није већа од вредности трећег, четвртог и свих даље елемената.

  - У последњем пролазу кроз низ (*i= n-2)*:
    за *j=i+1= n-1*, ако је niz[*n-2*] > niz[*n-1*], следи замена места,

После овог пролаза, на месту претпоследњег елемента у низу је вредност која није већа од вредности последњег елемента.

Број итерација (поређења парова елемената) за дужину низа *n* је:

за *i=0*, *(n-1)* поређења,
за *i=1*, *(n-2)* поређења,
:
за *i= n-3*, два поређења,
за *i= n-2*, једно поређење,

што је сума аритметичког реда:

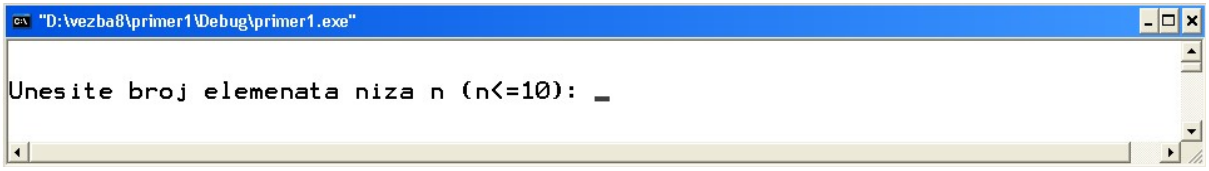
S = 1 + 2 +...+ (n-2) + (n-1), а то је број *(1+(n-1))∗(n-1)/2 = n∗(n-1)/2.*

(За 4 елемента низа број поређења је *4∗(4-1)/2 = 6*).

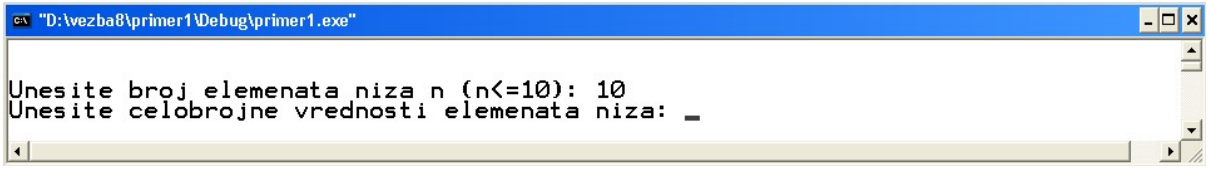
Број замена зависи од задатог низа, у најбољем случају је 0,а у најгорем случају је исти као и број поређења, за дужину низа *n* је *n∗(n-1)/2* (у овом случају то је број 6).

Задатак програма је био да поређа елементе задатог низа у неопадајући поредак. За нерастући поредак задатог низа једина разлика би била у услову у линији 32. овог програма, уместо *if(niz[i] > niz[j])* био би услов *if(niz[i] < niz[j]).*

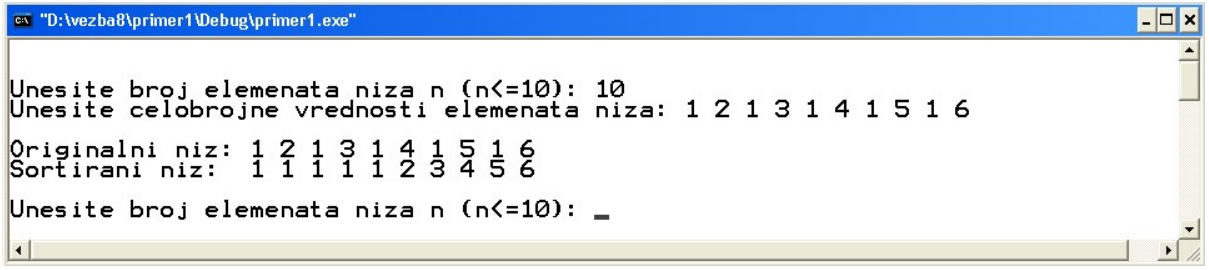
На сликама 8.1 а) и 8.1 б) приказани су промптови програма кориснику, за задавање низа, а на слици 8.1 в) приказ на екрану једног задатог низа и сортираног истог низа у неопадајући поредак. На слици 8.1 г) може се видети да се извршавање програма прекида када се зада дужина низа, ван траженог опсега, од 0 до 10.



Слика 8.1 а) Излаз програма **primer8_1** (први део)



Слика 8.1 б) Излаз програма **primer8_1** (други део)



Слика 8.1 в) Излаз програма **primer8_1** (трећи део)

Слика 8.1 г) Излаз програма **primer8_1** (четврти део)

Изворни кôд програма **primer8_2**

```
1:      /*primer8_2.c*/
2:      /*Sortiranje svakog zadatog niza celih brojeva*/
3:      /*u neopadajuci poredak, METODOM UMETANJA*/
4:
5:      #include <stdio.h>
6:      #include <stdlib.h>
7:      #define MAX 10
8:
9:      main()
10:     {       int i, j, pom, niz[MAX];
11:
12:             /*Prikaz na ekranu originalnog niza*/
13:             printf("\nOriginalni niz:\t");
14:             for(i=0; i<MAX; i++)
15:                     printf("%d ", niz[i]= rand( )/((double)RAND_MAX+1)*10);
16:
17:             /*Sortiranje niza u neopadajuci poredak*/
18:             for(i=1; i<MAX; i++)
19:             {       pom = niz[i];
20:
21:                     for(j=i-1; j>=0; j--)
22:                             if(niz[j] > pom)
23:                                     niz[j+1] = niz[j];
24:                             else
25:                                     break;
26:
27:                     niz[j+1] = pom;
28:             }
29:
30:             /*Prikaz na ekranu sortiranog niza*/
31:             printf("\nSortirani niz:\t");
32:             for(i=0; i<MAX; i++)
33:                     printf("%d ",niz[i]);
34:             printf("\n\n");
35:     }
```

Анализа програма **primer8_2**

У овом примеру је програм који реализује алгоритам за сортирање низа случајно генерисаних бројева, такође у неопадајући поредак, али методом уметања.

За низ случајно генерисаних целих бројева, у опсегу од 0 до 10 (линије 14. и 15.) у део програма од 18. до 28. линије реализује само сортирање:

- У спољашњој петљи (од линије 18.) дефинисани су опет пролази кроз низ, од другог *(i=1)* до последњег *(i=n-1)* елемента и у сваком од тих пролаза помоћна променљива сачува вредност одговарајућег елемента (*pom = niz[i]).*
- У унутрашњој петљи (од линије 21.) реализује се уметање вредности на одговарајуће место у низу, тако да поредак буде неопадајући.
  - У првом пролазу кроз низ (*i=1)*:
    pom = niz[1]
    за *j=i-1=0*, ако је *niz[0] > pom*, онда следи:
    niz[1] = niz[0],
    на крају итерације: *niz[0] = pom*
    После овог пролаза у неопадајући поредак доведен је само први пар елемената.
  - У другом пролазу кроз низ (*i=2)*:
    pom = niz[2]
    за *j=i-1=1*, ако је *niz[1] > pom,* онда следи:
    niz[2] = niz[1].
    за *j=i-2=0*, ако је *niz[0] > pom*, онда следи:
    niz[1] = niz[0],
    на крају итерације: *niz[0] = pom*.
    После другог пролаза у неопадајући поредак доведена су прва три елемента.
- У последњем пролазу кроз низ (*i*= MAX-1*)*:
  за *j=i-1*== MAX-2, ако је *niz[MAX-2] > pom,* онда следи:
  niz[MAX-1] = niz[MAX-2].
  :
  за *j=i-(MAX-1)=0*, ако је *niz[0] > pom*, онда следи:
  niz[1] = niz[0],
  на крају итерације: *niz[0] = pom*.
  После последњег пролаза, у неопадајући поредак доведени су сви елементи низа.

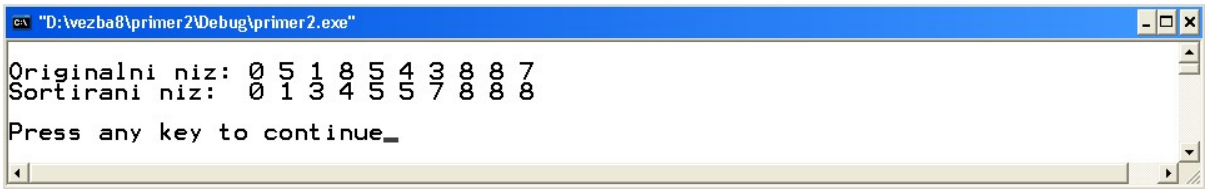
Број итерација (поређења парова елемената) за дужину низа *n* је:

- У најгорем случају је исти број поређења као и код претходне методе, а то је број *n∗(n-1)/2* (за 4 елемента низа: *4∗(4-1)/2 = 6*).
- У најбољем случају:
  за *i=1*, једно поређење и прелаз на следећу итерацију,
  за *i=2*, једно поређење и прелаз на следећу итерацију,
  :
  за *i= MAX-1*, једно поређење и крај петље,
  и онда је број поређења *n-1* (за 4 елемента низа: *4-1=3*).

Број замена зависи од задатог низа, у најбољем случају је 0,а у најгорем случају је исти као и број итерација, за дужину низа *n*, је *n∗(n-1)/2.*

За нерастући поредак задатог низа, једина разлика је у услову у линији 22. овог програма, уместо услова *if if(niz[j] > pom* био би услов if(niz[j] < pom).

На слици 8.2 може се видети резултат овог програма.



Слика 8.2 Излаз програма **primer8_2**

Изворни кôд програма **primer8_3**

```
1:      /∗primer8_3.c∗/
2:      /∗Prikaz na ekranu vrednosti elemenata zadatog niza, bez ponavljanja∗/
3:
4:      #include <stdio.h>
5:      #define MAX 10
6:
7:      main()
8:      {       int i, j, nadjeni_isti;
9:              float niz[MAX];
10:
11:             /∗Zadavanje niza∗/
12:             printf("\n\nUnesite %d realnih brojeva:\n\n",MAX);
13:             for(i=0; i<MAX; i++)
14:                     scanf("%f", &niz[i]);
15:
16:             /∗Prikaz elemenata niza bez ponavljanja∗/
17:             printf("\nElementi zadatog niza, bez ponavljanja:\n\n");
18:             for(i=0; i<MAX-1; i++)
19:             {       nadjeni_isti=0;
20:
21:                     for(j=i+1; j<MAX; j++)
22:                             if(niz[i]==niz[j])
23:                             {       nadjeni_isti=1;
24:                                     break;
25:                             }
26:
27:                     if(!nadjeni_isti)
28:                             printf("%f ", niz[i]);
29:             }
30:             printf("%f\n\n", niz[MAX-1]);
31:     }
```

Анализа програма **primer8_3**

У програму овог примера заступљен је алгоритам сортирања (методом избора) за приказ елемената задатог низа, без понављања.

У линији 8. декларисана је статусна променљива *nadjeni_isti*, која је у линији 19. иницијализована на вредност 0, јер на почетку пролаза нема истих.

После читања низа са тастатуре у линијама од 12. до 14. (слика 8.3 а)) програм има две петље (карактеристичне за методу избора):

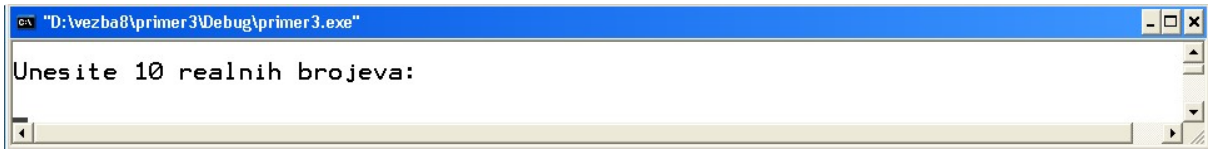
- У спољашњој петљи (од линије 18.) прво се претпостави да нису нађени исти (линија 19.)
- Онда се у унутрашњој петљи реализују поређења за одговарајуће парове елемената:
  - У првом пролазу (*i=0*) први елемент се пореди са свим елементима редом, од другог.
  - У другом пролазу (*i=1*) други елемент се пореди са свим осталим елементима редом.
  - У последњем пролазу (*i=n-1*) предпоследњи елемент се пореди са последњим елементом.

  Програм, у унутрашњој петљи, тражи да ли се вредност елемента *niz[i]* понавља и први пут када се понови, статус *nadjeni_isti* поставља на вредност 1 и прекида поређење са осталим елементима (наредба *break* у линији 24.)

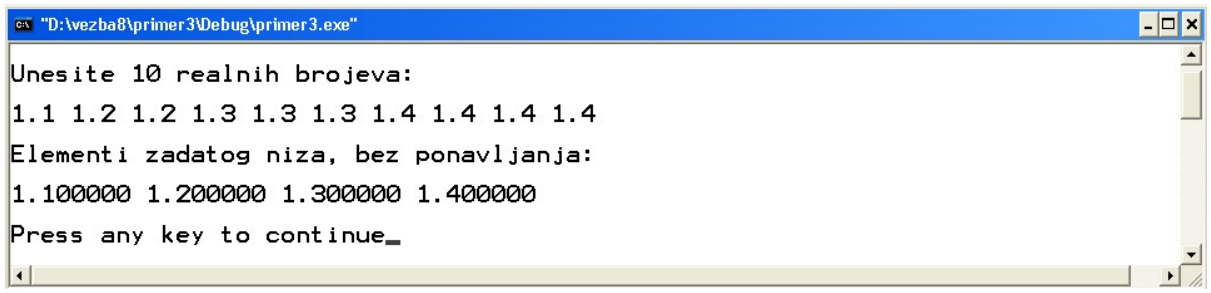
По излазу из унутрашње петље, ако није било понављања елемената, статус *nadjeni_isti* има вредност 0 и вредност елемента niz[i] приказује се на екрану.

Дакле, на екрану ће бити приказане вредности елемената задатог низа које немају понављања, као и оних елемента задатог низа у којима се последњи пут појављују вредности са понављањем.

На слици 8.3 б) може се видети резултат овог програма.



Слика 8.3 а) Излаз програма **primer8_3** (први део)



Слика 8.3 б) Излаз програма **primer8_3** (други део)

Изворни кôд програма **primer8_4**

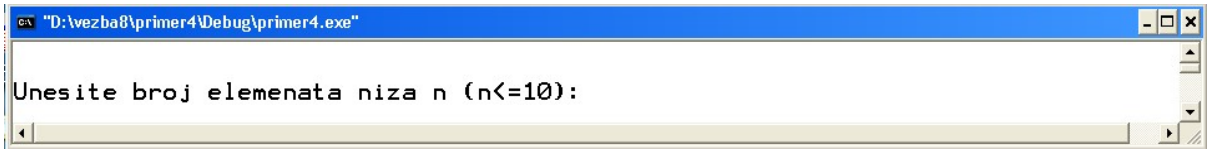
```
1:      /*primer8_4.c*/
2:      /*Pretrazivanje zadatog niza celih brojeva,*/
3:      /*METODOM LINEARNOG PRETRAZIVANJA*/
4:
5:      #include <stdio.h>
6:      #define MAX 10
7:
8:      main()
9:      {       int i, n, niz[MAX], broj, nadjen;
10:
11:             while(1)
12:             {
13:             /*Zadavanje duzine niza*/
14:             printf("\n\nUnesite broj elemenata niza n (n<=%d): ",MAX);
15:             scanf("%d", &n);
16:
17:             if(n<1 || n>MAX)
18:                     break;
19:
20:             /*Zadavanje elemenata niza*/
21:             printf("Unesite celobrojne vrednosti elemenata niza: ");
22:             for(i=0; i<n; i++)
23:                     scanf("%d", &niz[i]);
24:
25:             /*Zadavanje trazene vrednosti*/
26:             printf("Unesite trazenu celobrojnu vrednost: ");
27:             scanf("%d", &broj);
28:             nadjen = 0;
29:
30:             /*Pretrazivanje niza*/
31:             for(i=0; i<n; i++)
32:                     if(niz[i]==broj)
33:                     {       nadjen = 1;
34:                             printf("\nVrednost %d ima %d. element niza.",broj,i+1);
35:                     }
36:
37:             if(!nadjen)
38:                     printf("\nVrednost %d nije nadjena u nizu.",broj);
39:             }
40:     }
```

Анализа програма **primer8_4**

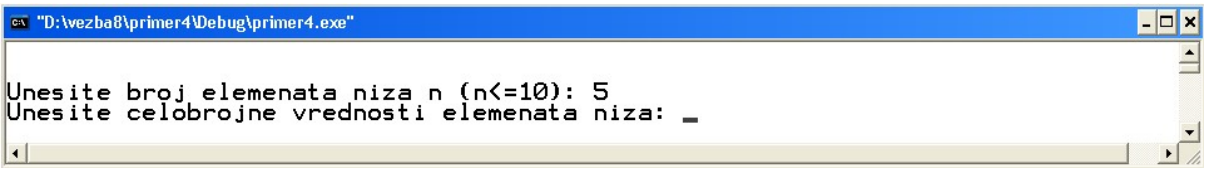
Претраживања низа је, такође, програмски задатак који се често појављује. За тражење одређене вредности међу елементима низа могу се користити две методе: једна је метода линеарног, а друга бинарног претраживања низа.

У програму овог примера коришћена је метода линеарног претраживања. Почетак програма (задавање низа) је исти као кôд програма *primer8_1*. После читања задатог низа (слике 8.4 а) и 8.4 б)), као и задате вредности која се тражи у низу (променљива *broj*) у линијама 26. и 27. (слика 8.4 в)) и постављања вредности статусне променљиве *nadjen* на 0 (линија 28.), програм у петљи (линије од 31. до 35.) пролази редом кроз низ и за сваки елемент који има тражену вредност пријављује на екрану његов редни број (слика 8.4 г)) и поставља вредност променљиве *nadjen* на 1. Понавља описани поступак за сваки учитани низ док се за дужину низа не зада број 0 (слика 8.4 д)).

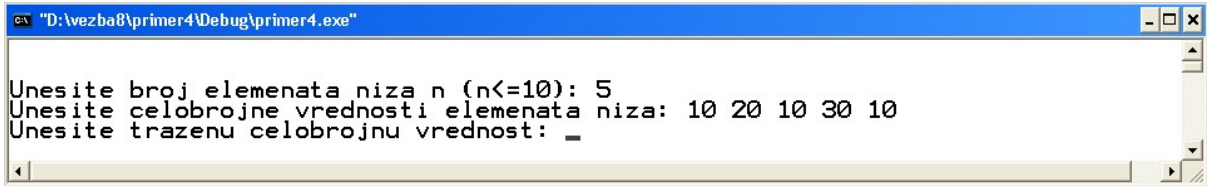
Ова метода претраживања низова је применљива и код произвољно уређених задатих низова, али има велики број пролаза кроз низ (то је број елемената низа) и због тога није погодна код низова велике дужине.



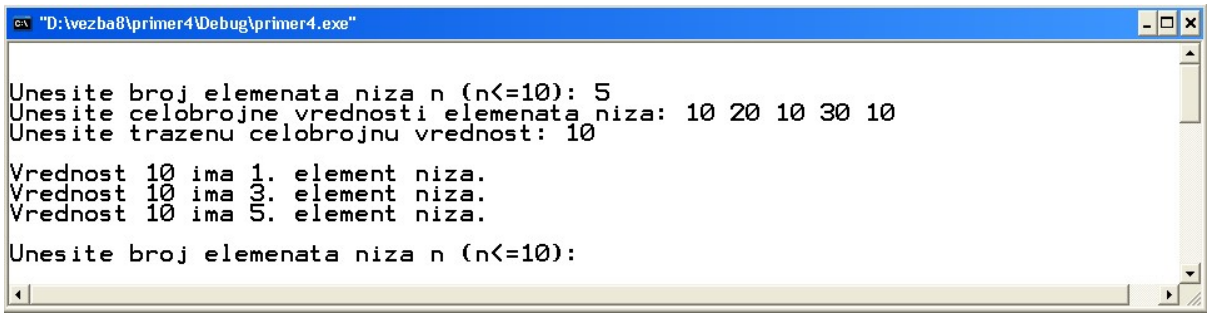
Слика 8.4 а) Излаз програма **primer8_4** (први део)



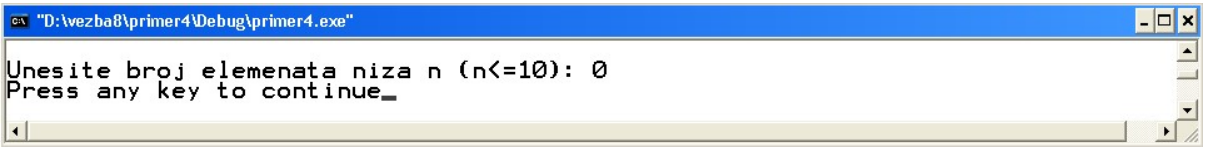
Слика 8.4 б) Излаз програма **primer8_4** (други део)



Слика 8.4 в) Излаз програма **primer8_4** (трећи део)



Слика 8.4 г) Излаз програма **primer8_4** (четврти део)



Слика 8.4 д) Излаз програма **primer8_4** (пети део)

Изворни кôд програма **primer8_5**

```
1:      /*primer8_5.c*/
2:      /*Pretrazivanje zadatog niza celih brojeva,*/
3:      /*METODOM BINARNOG PRETRAZIVANJA*/
4:
5:      #include <stdio.h>
6:      #define MAX 10
7:
8:      main()
9:      {       int i, niz[MAX], broj, nadjen=0;
10:             int i_min=0, i_max=MAX-1, i_srednji;
11:
12:             /*Formiranje niza*/
13:             for(i=0; i<MAX; i++)
14:                     niz[i] = 2*i;
15:
16:             /*Prikaz na ekranu originalnog niza*/
17:             printf("\nOriginalni niz:\t");
18:             for(i=0; i<MAX; i++)
19:                     printf("%d ",niz[i]);
20:
21:             /*Zadavanje trazene vrednosti*/
22:             printf("\n\nUnesite trazenu celobrojnu vrednost: ");
23:             scanf("%d", &broj);
24:
25:             /*Pretrazivanje niza*/
26:             while(i_min <= i_max)
27:             {       i_srednji = (i_min + i_max)/2;
28:
29:                     if(broj == niz[i_srednji])
30:                     {       nadjen = 1;
31:                             printf("\nVrednost %d ima ",broj);
32:                             printf("\n%d. element niza.\n\n",i_srednji+1);
33:                             break;
34:                     }
35:                     else if(broj < niz[i_srednji])
36:                             i_max = i_srednji - 1;
37:                     else
38:                             i_min = i_srednji +1;
39:             }
40:
41:             if(!nadjen)
42:                     printf("\nVrednost %d nije nadjena u nizu.\n\n",broj);
43:     }
```

Анализа програма **primer8_5**

Метода линеарног претраживања није ефикасна код дугих низова и због тога је развијена метода бинарног претраживања, која је применљива само ако је низ сортиран у растући или опадајући поредак.

За потребе алгоритма који ће бити примењен у програму овог примера декларисани су индекси *i_min, i_max* и *i_srednji*. Индекс *i_min* је за почетак дефинисан као индекс првог елемента (вредност 0), а индекс *i_max* као индекс последњег елемента (вредност *MAX-1*).

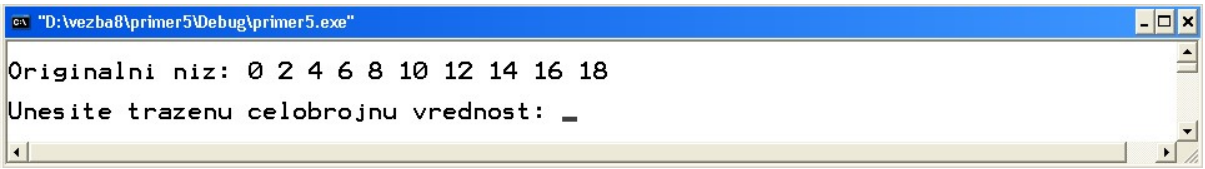
У линијама 13. и 14. формиран је низ (niz*[i] = 2*i,* од *i=0* до *i<MAX*), дакле елементи низа су парни бројеви редом, од 0 до *2* (MAX-1)*.

После задате тражене вредности у низу (променљива *broj*) у линијама 22. и 23., следи само претраживање низа, на следећи начин:

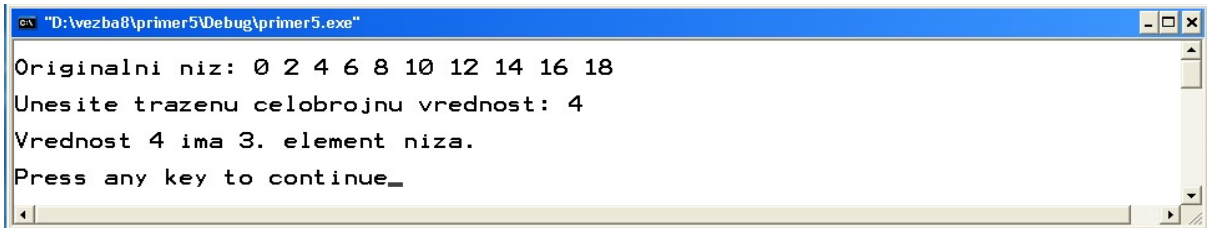
- У *while* петљи, све док је *i_min* <= *i_max*, индекс i_srednji добија средњу вредност индекса *i_min* и *i_max* (линија 27.).
- Ако елемент низа са индексом *i_srednji* има тражену вредност програм ово пријављује на екрану и прекида петљу (део кôда од 29. до 34. линије). У овом случају задати низ је приказан на слици 8.5 а) ,а задавање траженог броја се може видети на слици 8.5 б).
- Ако то није случај и ако тражени број има вредност мању од вредности елемента са индексом *i_srednji* , то значи да ће програм даље претраживати само елементе низа до елемента са индексом *i_srednji-1* (*i_max* добија вредност *i_srednji-1)*.
- Ако тражени број има вредност већу од вредности елемента са индексом *i_srednji* , онда ће програм даље претраживати само елементе низа од елемента са индексом *i_srednji+1* (*i_min* добија вредност *i_srednji+1)*.

У свакој следећој итерацији петље (под условом да још није нађен тражени број) број елемената који се претражује се преполови. Следи опис петље у овом случају (тражи се број 4).

- У првој итерацији претражује се низ: 0 2 4 6 8 10 12 14 16 18,
  *i_srednji = (0 + 9)/2 = 4 (*целобројно дељење*),*
  задовољен је услов *broj < niz[i_srednji],* јер је *4<niz[4],* и онда је *i_max=i_srednji-1=3.*
- У другој итерацији претражује се низ: 0 2 4 6,
  *i_srednji = (0 + 3)/2 = 1 (*целобројно дељење*),*
  задовољен је услов *broj > niz[i_srednji],* јер је *4>niz[1],* и онда је *i_min=i_srednji+1=2.*
- У трећој итерацији претражује се низ: 4 6,
  *i_srednji = (2 + 3)/2 = 2 (*целобројно дељење*),*
  задовољен је услов *broj = niz[i_srednji],* јер је *4=niz[2],* и онда је тражена вредност 4 нађена у елементу низа са индексом 2.



Слика 8.5 а) Излаз програма **primer8_5** (први део)



Слика 8.5 б) Излаз програма **primer8_5** (други део)

Изворни кôд програма **primer8_6**

```
1:      /*primer8_6.c*/
2:      /*Sortiranje zadatog niza imena po abecednom redu.*/
3:      /*Za kraj izvrsavanja programa treba uneti dve kose crte*/
4:
5:      #include <stdio.h>
6:      #include <string.h>
7:      #define BROJ_IMENA 100
8:      #define DUZINA_IMENA 20
9:
10:     main( )
11:     {       char    spisak[BROJ_IMENA][DUZINA_IMENA+1],
12:                     pom_ime[DUZINA_IMENA+1];
13:             int i, j, n = 0;
14:
15:             /*Zadavanje niza imena*/
16:             printf("\nOriginalni niz imena:");
17:             printf("\n(Za kraj uneti dve kose crte)\n\n");
18:
19:             do
20:                     gets(spisak[n]);
21:             while(strcmp(spisak[n++],"//") != 0);
22:             n--;
23:
24:             /*Sortiranje po abecednom redu*/
25:             printf("\nNiz imena, sortiran po abecednom redu:\n");
26:             for(i=0; i<n-1; i++)
27:                     for(j=i+1; j<n; j++)
28:                             if(strcmp(spisak[i], spisak[j]) > 0)
29:                             {       strcpy(pom_ime, spisak[i]);
30:                                     strcpy(spisak[i], spisak[j]);
31:                                     strcpy(spisak[j], pom_ime);
32:                             }
33:
34:             /*Prikaz sortiranog niza imena*/
35:             for(i=0; i<n; i++)
36:                     puts(spisak[i]);
37:             printf("\n");
38:     }
```

Анализа програма **primer8_6**

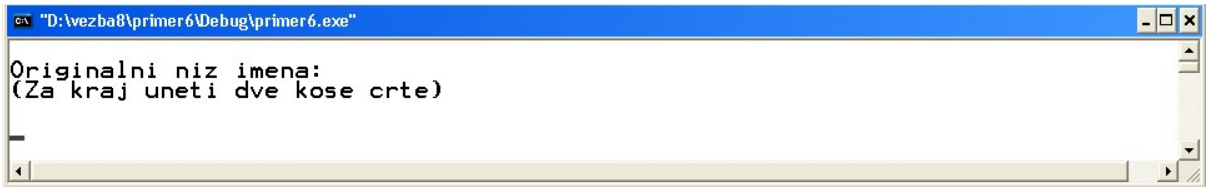
У овом програму искоришћена је метода избора за сортирање задатог низа имена по абецедном реду.

За задати низ имена (линије од 16. до 22.) пролази се дефинишу спољашњом петљом, а поређења и евентуалне замене у унутрашњој петљи.

Поређење се реализује помоћу функције *strcmp*(), а замена помоћу функције *strcpy()*. Обе наведене функције су из стандардне библиотеке *string.h.* Абецедни ред значи да треба да буде*:*

$$spisak[0] \leq spisak[1] \leq spisak[1] \leq spisak[BROJ\_IMENA-1]$$

На слици 8.6 а) приказан је почетни промпт програма, а на слици 8.6 б) задата комплетна имена и задати знак за крај уноса, као и резултат програма (имена сортирана на тражени начин).



Слика 8.6 а) Излаз програма **primer8_6** (први део)

Слика 8.6 б) Излаз програма **primer8_6** (други део)

Практични савети

- ✓ Алгоритам за сортирање и претраживање низа изабрати у зависности од самог низа.

Евентуалне грешке

- ➢ Не треба мешати кораке из разних алгоритама за сортирање и претраживање, треба се држати само једног алгоритма.

Задаци за припрему вежбе 8. у лабораторији

Задатак 1.

Написати програм на језику *C* који сортира сваки задати низ целих бројева, дужине *n (n<=20)* у нерастући поредак.

Задаци за вежбу 8. у лабораторији

Задатак 1.

Написати програм на језику *C* који од низа целих бројева дужине *n (n<=20)*, задатог преко тастатуре, формира:

а) други низ, чији су елементи поређани у растући поредак,
б) трећи низ, чији су елементи поређани у опадајући поредак.

Задатак 2.

Написати програм на језику *C* који од елемената неопадајућег низа целих бројева, дужине *n (n<=20)*, задатог преко тастатуре, и целог броја, задатог такође преко тастатуре, формира неопадајући низ дужине *n+1.*

Задатак 3.

Написати програм на језику *C* који приказује на екрану одговарајућу поруку, у зависности од тога да ли је низ реалних бројева дужине *n (n<=20)*, задат преко тастатуре, уређен у растући поредак или не.

Задатак 4.

Написати програм на језику *C* који елементе низа реалних бројева дужине *n (n<=20)*, задатог преко тастатуре, поставља у инверзни поредак (замењује места елементима: првом и последњем, другом и претпоследњем,...).

Задатак 5.

Написати програм на језику *C* који испитује да ли се задати *string* исто чита од почетка и од краја.

Вежба 9.

Показивачи и примена показивача код низова у програмима на језику *C*

Декларација показивача

Иницијализација показивача

Аритметика показивача

Показивачи и низови

Примери

Изворни кôд програма **primer9_1**

```
1:      /*primer9_1.c*/
2:      /*Prikaz na ekranu  sadrzaja i adrese celobrojne promenljive*/
3:      /*na dva nacina: direktno i indirektno (pomocu pokazivaca)*/
4:
5:      # include <stdio.h>
6:
7:      main( )
8:      {       int x = 1;
9:              int  *ptr;              /*deklaracija pokazivaca*/
10:             ptr = &x;               /*inicijalizacija na adresu promenljive x*/
11:
12:             printf("\nDirektan pristup: x = %d", x);
13:             printf("\nIndirektan pristup: x = %d", *ptr);
14:             printf("\nAdresa promenljive prikazana na 1. nacin: %d", &x);
15:             printf("\nAdresa promenljive prikazana na 2. nacin: %d\n\n", ptr);
16:     }
```

Анализа програма **primer9_1**

Показивач (*pointer*) је променљива која чува адресу објекта (променљиве основног типа, било ког елемента у низу променљивих основног типа, структуре, било ког елемента у низу структура...) или функције.

Већина програма почетничког нивоа користи само статички декларисане променљиве. Свака статички декларисана променљива има своју адресу у оперативној меморији. Даље у програму овим променљивама може се приступити помоћу имена променљиве (ако се ради о променљивој основног типа), или помоћу имена низа и индекса елемента (ако се ради о низу).

Адреса променљиве, декларисане на било који начин у програму, може се добити декларацијом и иницијализацијом одговарајућег показивача (тачније показивачке променљиве). На овај начин, адреса доспева у први план програма и може се користити за повећање ефикасности програма (динамичку доделу и ослобађање меморије).

Програм у овом примеру користи показивач који треба да чува адресу променљиве типа *int* и због тога декларација показивача који се зове *ptr* има облик наредбе у линији 9. (*int *ptr*;). Оператор индиректног адресирања (*) каже да се ради о показивачкој, а не о променљивој основног типа. Даље у програму, ова показивачка променљива може бити коришћена само за чување адресе променљиве типа *int.* За чување адресе променљиве типа *char* декларација би имала облик: *char *ptr;* за чување адресе променљиве типа *double*: *double *ptr… .* Ако је потребно да се показивач користи за чување

адреса променљивих различитог типа декларација има облик: *void ∗ptr;* (*void* значи било који тип дефинисан у језику *C*).

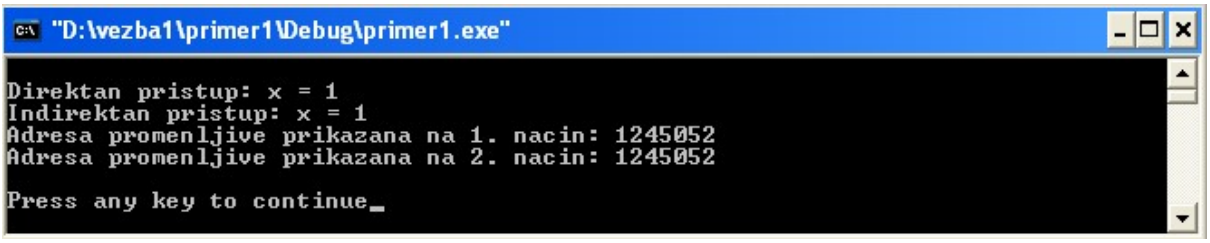
Без обзира на тип променљиве чију адресу чува, тип показивачке променљиве увек је целобројни а опсег показивачке променљиве зависи од капацитета оперативне меморије у рачунару.

Од линије 10. показивач *ptr* је и иницијализован. Показивачкој променљивој *ptr* додељена је адреса променљиве *x* (*ptr = &x)*. Један пример примене оператора адресе (*&*) је и стандардна функција *scanf("%d", &broj)* која чита податак до првог белог знака и чува га од одговарајуће адресе, резервисане за претходно декларисану променљиву *broj;*

Након описане иницијализације показивача *ptr:*

- садржају променљиве *x* може се приступити даље у програму на два начина: директно *(x)* и индиректно *(∗ptr)*,
- адреси променљиве *x* може се приступити даље у програму на два начина: директно *(&x)* и индиректно *(ptr)*.

На слици 9.1 може се видети да наредбе у линијама 12. и 13. приказују на екрану исто. Прва приказује на екрану садржај променљиве која се зове *x,* а друга садржај променљиве на коју показује *ptr,* то значи исти садржај. Исто важи и за наредбе у линијама 14. и 15. (слика 9.1). Прва приказује на екрану адресу променљиве која се зове *x*, а друга садржај показивачке променљиве која се зове *ptr,* то значи исту адресу.



Слика 9.1 Излаз програма **primer9_1**

Изворни кôд програма **primer9_2**

```
1:      /*primer9_2.c*/
2:      /*Prikaz na ekranu adresa elemenata nizova razlicitih tipova*/
3:
4:      #include <stdio.h>
5:
6:      main( )
7:      {       int i;
8:              char cniz[10];
9:              int iniz[10];
10:             double dniz[10];
11:
12:             printf("\t\tchar\tint\tdouble\n\n");
13:
14:             for(i=0; i<10; i++)
15:                     printf("Element %d.\t%d\t%d\t%d\n",i+1,&cniz[i],&iniz[i],&dniz[i]);
16:     }
```

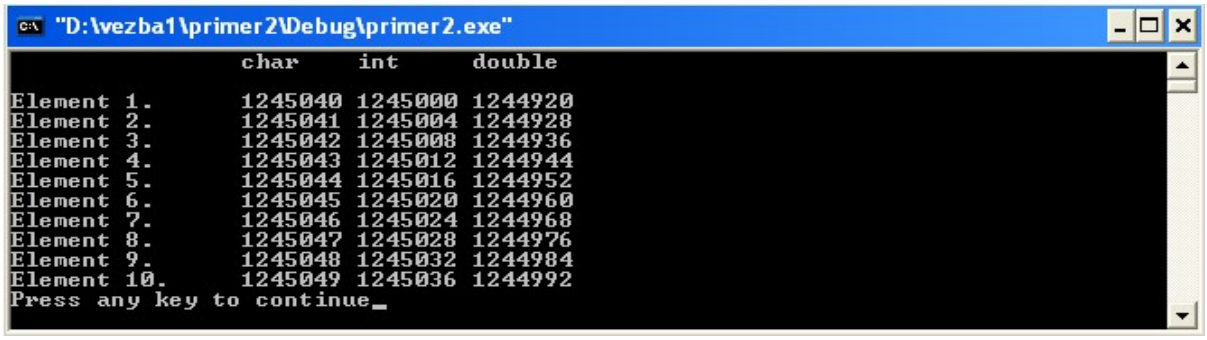
Анализа програма **primer9_2**

Програм у овом примеру користи адресни оператор за приказ адреса у оперативној меморији, од низова променљивих различитог типа.

У линијама од 8. до 10. декларисани су: низ карактера, низ целих бројева у опсегу *int* и низ реалних бројева у опсегу *double* (дужина сваког од поменутих низова је 10). После наредбе у линији 12. која штампа на екрану заглавље табеле (називе типова декларисаних низова), програм у *for* петљи (линије 14. и 15.) приказује за сваки елемент сваког низа редом: редни број и адресу у оперативној меморији. На слици 9.2 може се видети да се адресе суседних елемената разликују:

- код низа типа *char* : за 1,
- код низа типа *int* : за 4,
- код низа типа *double*: за 8.

Дакле, адреса сваког елемента низа померена је у односу на адресу претходног елемента истог низа: за величину податка датог типа (елемента низа) у бајтовима.



Слика 9.2 Излаз програма **primer9_2**

Изворни кôд програма **primer9_3**

```
1:    /*primer9_3.c*/
2:    /*Prikaz  na ekranu vrednosti elemenata*/
3:    /*jednodimenzionalnog niza celih brojeva pomocu pokazivaca na niz*/
4:
5:    #include <stdio.h>
6:
7:    main( )
8:    {       int niz[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
9:            int  *ptr, i;
10:           ptr = niz;              /*pokazivac sadrzi adresu prvog elementa niza*/
11:
12:           for(i=0; i<10; i++)
13:                   printf("%d\n",*ptr++);
14:   }
```

Анализа програма **primer9_3**

У програму овог примера декларисан је и иницијализован низ целих бројева (линија 8.), као и показивачка променљива *ptr* која може бити коришћена за чување адресе целобројне променљиве типа *int* (линија 9.).

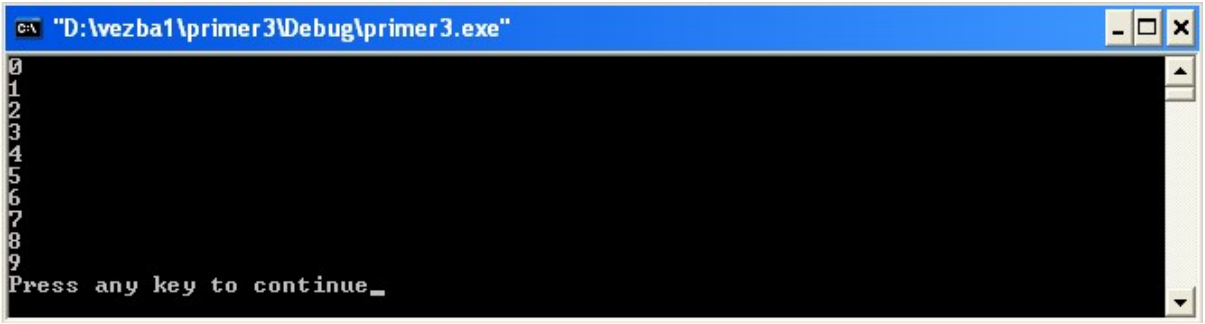
У програмима на језику *C* важи следеће: само име низа (било ког типа) представља константну адресу тог низа (адресу његовог првог елемента). Наредба у линији 10. (*ptr = niz*;) значи да је показивач *ptr* иницијализован на адресу низа (садржи адресу почетка низа).

Једна од операција адресне аритметике (аритметике показивача) је инкремент показивача. Оператор инкремент примењен над показивачком променљивом не увећава садржај променљиве за 1, као код променљиве основног типа. Код показивача декларисаног као *int *ptr;* и иницијализованог на адресу почетка низа, после извршене наредбе *ptr++;* садржај истог показивача биће: адреса почетка низа увећана за 4 (број бајтова типа *int)*, што значи да ће садржати адресу следећег елемента у низу. Инкремент показивача увећава садржај показивача за величину типа на који показује. На овај начин може се помоћу показивача на почетак низа приступити било ком елементу у низу. (Декремент показивача умањује његов садржај за величину типа на који показује.)

Наредба *for* петље у линијама 12. и 13. еквивалентна је наредби:

```
for(i=0; i<10; i++)
{       printf("%d\n",*ptr);
        ptr++;
}
```

која помоћу показивача приказује на екрану садржај сваког елемента низа, а онда инкрементира показивач на адресу следећег елемента у низу. Резултат програма је приказ на екрану садржаја свих елемената задатог низа редом (слика 9.3).



Слика 9.3 Излаз програма **primer9_3**

Изворни кôд програма **primer9_4**

```
1:      /*primer9_4.c*/
2:      /*Prikaz na ekranu  sadrzaja znakovne promenljive*/
3:      /*na dva nacina: direktno i indirektno (pomocu pokazivaca)*/
4:
5:      #include <stdio.h>
6:
7:      main( )
8:      {       char c;
9:              char *ptr;
10:             ptr = &c;               /*ptr sadrzi adresu promenljive c*/
11:
12:             c = 'A';
13:             printf("\nDirektna dodela vrednosti:\tc = ‘%c’", c);
14:
15:             *ptr = 'a';
16:             printf("\nIndirektna dodela vrednosti:\tc = ‘%c’", c);
17:
18:             /*Prikaz vrednosti promenljive c indirektno, pomocu ptr*/
19:             printf("\nPosredan pristup:\t\t*ptr = ‘%c’\n\n", *ptr);
20:     }
```

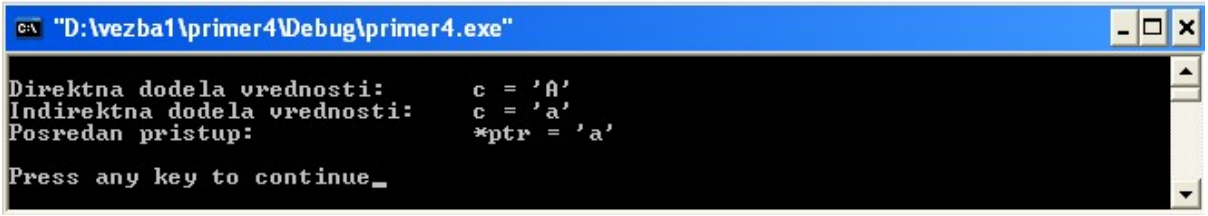
Анализа програма **primer9_4**

У програму овог примера декларисан је показивач на променљиву типа *char* (линија 9.) и иницијализован на конкретну адресу променљиве *c* истог типа (линија 10.).

Наредбама у линијама 12. и 13. додељен је садржај (*ASCII* кôд слова *A*) променљивој *c*, на директан начин и приказан на екрану садржај исте променљиве, на директан начин.

Наредбама у линијама 15. и 16. додељен је садржај (*ASCII* кôд слова *a*) променљивој *c*, на индиректан начин, помоћу показивача и приказан на екрану садржај исте променљиве, на директан начин.

Наредбом у линији 19. приказан је на екрану садржај променљиве *c* на индиректан начин, помоћу показивача. На слици 9.4 је резултат овог програма.



Слика 9.4 Излаз програма **primer9_4**

Изворни кôд програма **primer9_5**

```
1:    /*primer9_5.c*/
2:    /*Prikaz na ekranu  sadrzaja znakovnog niza*/
3:    /*na dva nacina: direktno i indirektno (pomocu pokazivaca)*/
4:
5:    #include <stdio.h>
6:
7:    main( )
8:    {       int i;
9:            char *ptr;
10:           char string[] = "Jedan znakovni niz.\n";
11:
12:           /*Primena indeksiranja za prikaz znakovnog niza*/
13:           printf("\nPrikaz znakovnog niza pomocu indeksiranja:\t");
14:           for(i=0; string[i]!='\0'; i++)
15:                   putchar(string[i]);
16:
17:           /*Primena pokazivaca na znak za prikaz znakovnog niza*/
18:           printf("\nPrikaz znakovnog niza pomocu pokazivaca na niz:\t");
19:           for(ptr = string; *ptr != '\0'; ptr++)
20:                   putchar(*ptr);
21:
22:           putchar('\n');
23:   }
```

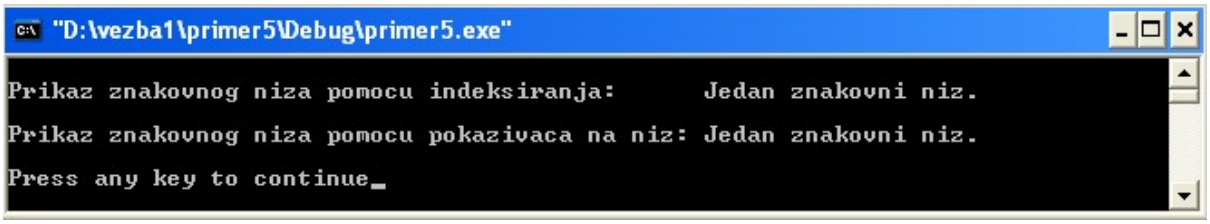
Анализа програма **primer9_5**

Програм у овом примеру садржи на почетку декларацију показивача на променљиву типа *char* (линија 9.) и декларацију и иницијализацију низа променљивих типа *char* (линија 10.).

У *for* петљи у линијама 14. и 15. програм све до последњег карактера ('\0'-ознаке за крај низа) приказује на екрану садржај једног по једног карактера (помоћу функције *putchar()*). У овој *for* петљи коришћен је директан приступ сваком елементу у низу (*string[i],* за вредности променљиве *i* од индекса првог до индекса претпоследњег карактера у низу).

У *for* петљи у линијама 19. и 20. програм опет све до последњег карактера ('\0'-ознаке за крај низа) приказује на екрану садржај једног по једног карактера низа (помоћу функције *putchar()*). Овде је коришћен други индиректан приступ сваком елементу у низу карактера помоћу показивача. У почетном изразу *for* петље показивач је иницијализован на адресу првог карактера у низу *(ptr = string)*. Све до последњег карактера, док је задовољен услов **ptr!='\0',* приказује се садржај карактера помоћу показивача *(putchar(*ptr);)* и инкрементира показивач на адресу следећег карактера у низу *(ptr++).*

Оба начина дају исти резултат (слика 9.5) и у овом случају потпуно је свеједно који ће начин бити коришћен.



Слика 9.5 Излаз програма **primer9_5**

Изворни кôд програма **primer9_6**

```
1:      /*primer9_6.c*/
2:      /*Prikaz na ekranu  sadrzaja niza karaktera*/
3:      /*u direktnom i inverznom poretku*/
4:
5:      #include <stdio.h>
6:
7:      main( )
8:      {       char  string[] = "abcdefg", *ptr;
9:              ptr = string;                   /*ptr sadrzi adresu prvog znaka u nizu*/
10:
11:             printf("\nZadati znakovni niz u direktnom poretku:\t");
12:             while(*ptr)
13:                     putchar(*ptr++);      /*putchar(*ptr++);⇔{putchar(*ptr);ptr++;}*/
14:
15:             printf("\n\nZadati znakovni niz u inverznom poretku:\t");
16:             while(--ptr >= string)
17:                     putchar(*ptr);
18:
19:             printf("\n\n");
20:     }
```

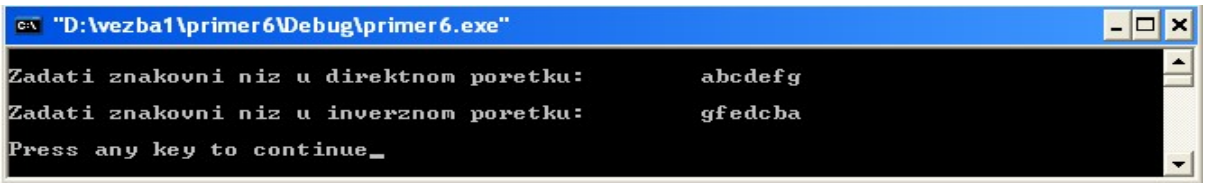
Анализа програма **primer9_6**

У програму овог примера декларисан је и иницијализован један низ карактера, као и показивач *ptr* на променљиву типа *char* (декларације променљивих једног типа и показивача на променљиву истог типа могу бити написане у истој линији) (линија 8.). Онда је показивач *ptr* иницијализован на адресу низа (првог карактера у низу) (линија 9.).

Петља *while* у линијама 12. и 13. тестира садржај на који показује *ptr.* Све док овај садржај није 0 (*'\0')* услов *while* петље биће испуњен и биће приказан садржај на који показује *ptr* и инкрементиран показивач *ptr* на адресу следећег карактера у низу. Резултат је приказ на екрану низа карактера у директном поретку (слика 9.6).

Петља *while* у линијама 16. и 17. декрементира (од адресе последњег карактера (‘*\0*’) низа на адресу претходног карактера у низу*)* а онда тестира адресу коју садржи *ptr.* Све док је ова адреса већа или једнака адреси почетка низа (*string)* услов *while* петље (у линији 12.) биће испуњен и биће приказан

садржај на који показује *ptr.* Резултат програма је приказ на екрану низа карактера у инверзном поретку (од претпоследњег до првог) (слика 9.6).



Слика 9.6 Излаз програма **primer9_6**

Изворни кôд програма **primer9_7**

```
1:      /*primer9_7.c*/
2:      /*Prikaz na ekranu odredjenog znaka u jednom redu teksta*/
3:      /*na dva nacina: pomocu indeksa i pomocu pokazivaca*/
4:
5:      #include <stdio.h>
6:      #include <string.h>
7:      #define MAX    80                  /*broj korisnih karaktera u stringu*/
8:
9:      main( )
10:     {       int rbr, l;
11:             char tekst[MAX+1], *ptr;
12:             ptr = tekst;                /*ptr sadrzi adresu prvog znaka u nizu*/
13:
14:             printf("\nUnesite jedan red teksta:\n\n");
15:             gets(tekst);
16:
17:             do
18:             {       printf("\nUnesite redni broj znaka u redu teksta: ");
19:                     scanf("%d", &rbr);
20:             }while(rbr<1 || rbr>(l=strlen(tekst)));
21:
22:             /*Prikaz odredjenog znaka u tekstu pomocu indeksa*/
23:             printf("\nPrikaz trazenog znaka pomocu indeksa:\t\t");
24:             putchar(tekst[rbr-1]);
25:
26:             /*Prikaz odredjenog znaka u tekstu pomocu pokazivaca*/
27:             printf("\nPrikaz trazenog znaka pomocu pokazivaca:\t");
28:             putchar(*(ptr + rbr-1));
29:
30:             putchar('\n');
31:     }
```

Анализа програма **primer9_7**

Програм у овом примеру декларише један низ карактера који се зове *tekst*, као и показивач *ptr* на овај низ (линије 11. и 12.). Онда исписује на екрану поруку да се унесе ред текста (слика 9.7 а)) и чита преко тастатуре задати знаковни низ, што је у овом случају један ред текста (линије 14. и 15.). Онда, у *do..while* петљи тражи да се преко тастатуре зада редни број знака у задатом тексту (слика 9.7 б)) све док задати редни број нема смисла или није у опсегу дужине задатог низа (линије од 17. до 20.).

Наредба у линији 24. приступа директно (помоћу индекса) карактеру са задатим редним бројем у задатом низу и приказује његов садржај на екрану.

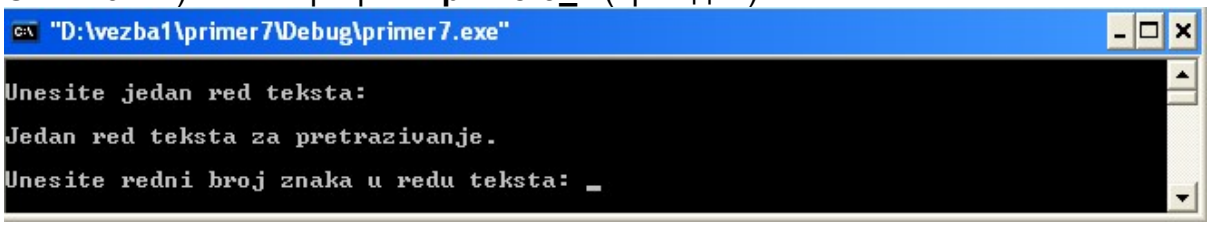
Наредба у линији 28. приступа индиректно (помоћу показивача) карактеру са задатим редним бројем у задатом низу и приказује његов садржај на екрану. У овом случају, елементу низа са индексом (*rbr-1*) приступа као садржају на који показује показивач (*ptr)* на почетак низа, инкрементиран за вредност (*rbr-1*). За декларисан низ дужине *n* и показивач *ptr* на исти низ важи:

директан приступ	1. индиректан приступ	2. индиректан приступ
niz[0]	∗niz	∗ptr
niz[1]	∗(niz+1)	∗(ptr+1)
:	:	:
niz[i]	∗(niz+i)	∗(ptr+i)
:	:	:
niz[n-1]	∗(niz+n-1)	∗(ptr+n-1)

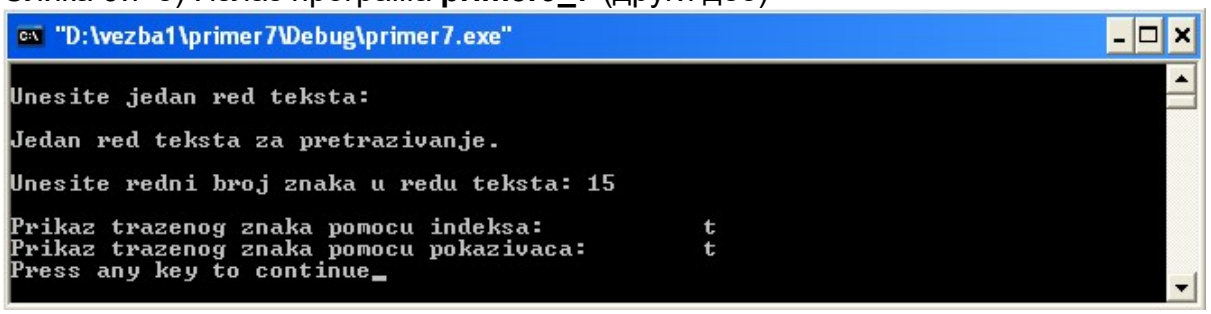
За први наведен индиректан приступ није потребна декларација показивача, само име низа користи се као показивач на почетак низа. За други наведен индиректан приступ користи се посебно декларисани показивач, али овај показивач може бити коришћен даље у програму и за чување адресе неке друге променљиве или низа, типа на који је показивач декларисан на почетку.

Директан и индиректан приступ дају исти резултат (слика 9.7в)).

Слика 9.7 а) Излаз програма **primer9_7** (први део)



Слика 9.7 б) Излаз програма **primer9_7** (други део)



Слика 9.7 в) Излаз програма **primer9_7** (трећи део)

Изворни кôд програма **primer9_8**

```
1:      /*primer9_8.c*/
2:      /*Pomeranje pozitivnih elemenata zadatog niza celih brojeva*/
3:      /*na pocetak niza i popunjavanje ostatka niza nulama*/
4:
5:      #include <stdio.h>
6:      #define MAX     10                      /*dozvoljena duzina niza*/
7:
8:      main()
9:      {       int i, n1, n2, niz[MAX];
10:             int *ptr1, *ptr2;
11:             ptr1 = ptr2 = niz;
12:
13:             do
14:             {       printf("\nBroj elemenata niza, n<=%d? ", MAX);
15:                     scanf("%d", &n1);
16:             }while(n1<1 || n1>MAX);
17:
18:             printf("\nZadati niz:\n\n");
19:             for(i=0; i<n1; i++)
20:                     scanf("%d", niz+i);
21:
22:             /*Sazimanje niza*/
23:             for(i=0; i<n1; i++)
24:             {       if(*ptr1 > 0)
25:                     {       *ptr2 = *ptr1;
26:                             ptr2++;
27:                     }
28:                     ptr1++;
29:             }
30:
31:             n2 = ptr2 - niz;
32:
33:             /*Popunjavanje ostatka niza nulama*/
34:             if(n2<n1)
35:                     for(i=n2; i<n1; i++)
36:                             *(niz+i) = 0;
37:
38:             printf("\nIsti niz posle sazimanja:\n\n");
39:             for(i=0; i<n1; i++)
40:                     printf("%d ", *(niz+i));
41:             printf("\n\n");
42:     }
```

Анализа програма **primer9_8**

Задатак програма у овом примеру је да задати низ целих бројева измени на следећи начин: да избаци елементе који нису позитивни, да позитивне елементе помери на почетак низа и остатак елемената попуни нулама.

Програм на почетку декларише низ целих бројева (линија 9.) и показиваче *ptr1* и *ptr2* (линија 10.), тако да сваки од њих може чувати адресу целобројне променљиве, било ког елемента декларисаног низа.

Наредба у линији 11. иницијализује сваки од декларисаних показивача (*ptr1* и *ptr2)* на адресу почетка декларисаног низа (првог елемента у низу).

Програм даље тражи број елемената низа *n1* (слика 9.8 а)) као и саме елементе низа (слика 9.8 б)) и учитава их са тастатуре (линије од 13. до 20.).

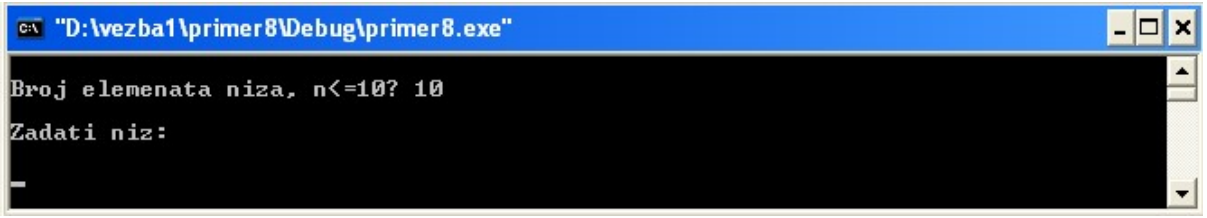
У *for* петљи, од линије 23., програм помера позитивне елементе задатог низа на његов почетак. За сваки елемент задатог низа, чији је садржај (садржај на који показује *ptr1*) већи од 0, *if* услов у линији 24. је испуњен и извршава се блок наредби у линијама 25. и 26.: садржај на који показује *ptr2* добија вредност садржаја на који показује *ptr1* и показивач *ptr2* инкрементира се на адресу следећег елемента у низу. Показивач *ptr1* инкрементира се у сваком случају на адресу следећег елемента у низу (линија 28.).

Укратко: показивач *ptr1* у петљи добија вредности адреса свих елемената задатог низа редом, док показивач *ptr2* у истој петљи добија вредности адреса само позитивних елемената, померених од почетка низа. На тај начин, садржаји свих позитивних елемената померају се на почетак низа а адресу последњег позитивног елемента у тако помереним елементима чува показивач *ptr2.* По завршеној *for* петљи променљивој *n2* додељује се број позитивних елемената, који се добија као разлика адресе *ptr2* и адресе *niz* (линија 31.).

Ако је број позитивних елемената задатог низа (померених на почетак) мањи од задатог броја елемената, наредба *for* у линијама 35. и 36. попуњава остатак елемената нулама.

Наредба *for* у линијама 39. и 40. приказује на екрану нови низ, формиран на горе описан начин (слика 9.8 в).

Слика 9.8 а) Излаз програма **primer9_8** (први део)



Слика 9.8 б) Излаз програма **primer9_8** (други део)

Слика 9.8 в) Излаз програма **primer9_8** (трећи део)

Практични савети

- ✓ У програму је могућа измена садржаја показивачке променљиве. Једна показивачка променљива на тип *int* може у једном делу програма чувати адресу једне променљиве типа *int*, у другом делу програма адресу друге променљиве типа *int*...
- ✓ Име низа (било ког типа) је показивачка константа и може се користити само као адреса тог низа (није дозвољена промена њеног садржаја).

Евентуалне грешке

- ➢ Показивач се не може користити док се не иницијализује (док му се не додели адреса неког објекта у програму).
- ➢ Не додељивати адресу променљиве једног типа (на пример *int*) показивачу који очекује адресу променљиве другог типа (на пример *float*).
- ➢ Не везивати тип саме показивачке променљиве за неки одређени тип (због тога што опсег целобројног типа показивачке променљиве зависи од меморије рачунара).
- ➢ При сваком инкременту и декременту показивача, садржај показивача не увећава се и не умањује за 1, већ за број бајтова типа на који показује.

Задаци за припрему вежбе 9. у лабораторији

Задатак 1.

Додати наредбе које недостају у следећем програму на језику *C* да би програм имао смисла:

```
#include  <stdio.h>
main( )
{       char c, *ptr_c;

        *ptr_c = 'X';

        putchar(*ptr_c);      }
```

Задатак 2.

За декларисане и иницијализоване променљиве у програму на језику *C*:

```
int i, x = 10, y = 20, z[ ] = {10, 20, 30, 40};
int *ptr;
ptr = &x;
```

и примењене операторе следећим редом:

```
y = *ptr;   *ptr = 0;   ptr = z;   *ptr = 1; *(ptr+1) = 2; *(ptr+2) = 3; *(ptr+3) = 4;
```

написати шта исписују на екрану следеће наредбе:

```
printf("%d %d\n", x, y);
 for(i=0; i<4; i++)
        printf("%d ",*(z+i));
```

Задатак 3.

Написати програм на језику C који елементе задатог низа целих бројева дужине *n (n≤10)* инвертује (мења места елементима: првом и последњем, другом и претпоследњем…). Користити индиректан приступ променљивама у низу.

Задаци за вежбу 9. у лабораторији

Урадити следеће програме на језику *C* применом показивача на низ.

Задатак 1.

Написати програм на језику *C* који чита са тастатуре текст дужине *n (n≤80)* знакова и цео број *k* и приказује на екрану:

а) првих *k* знакова задатог текста,
б) последњих *k* знакова задатог текста,
в) део задатог текста од задате позиције *k*.

Задатак 2.

Написати програм на језику *C* који чита са тастатуре текст дужине *n (n≤80)* знакова и приказује на екрану:

а) знакове задатог текста у инверзном поретку (замењује места знаковима: првом и последњем, другом и претпоследњем...),
б) речи задатог текста у инверзном поретку (замењује места речима: првој и последњој, другој и претпоследњој...).

Задатак 3.

Написати програм на језику *C* који чита са тастатуре низ целих бројева дужине *n (n≤10)*, а онда мења задати низ тако што избацује из њега сваки други елемент. Преостале елементе помера на почетак низа, а остатак низа попуњава нулама.

Задатак 4.

Написати програм на језику *C* који чита са тастатуре низ знакова дужине *n (n≤80)* и прихвата га у низ карактера, а онда мења овај низ тако што избацује из њега следеће беле знакове: ‘ ‘, ‘\t’, и ‘\v’, преостале знакове помера на почетак низа, а остатак низа попуњава карактерима ‘\0’.

Вежба 10.

Функције
у програмима на језику *C*

Дефиниција, декларација и позиви функције

Аргументи функције

Повратне вредности од функције

Рекурзивне функције

Примери

Изворни кôд програма **primer10_1**

```
1:      /*primer10_1.c*/
2:      /*Odredjivanje veceg od dva broja pomocu odgovarajuce funkcije*/
3:
4:      #include  <stdio.h>
5:
6:      int max(int x, int y);                          /*deklaracija funkcije*/
7:
8:      main( )
9:      {       int a, b, c;
10:
11:             printf("\nZadati dva cela broja:  ");
12:             scanf("%d%d", &a, &b);
13:
14:             c = max(a, b);                          /*poziv funkcije*/
15:             if(c == 0)
16:                     printf("\nZadati brojevi su isti!!!\n");
17:             else
18:                     printf("\nVeci od dva zadata broja je broj %d\n\n", c);
19:
20:             return 0;
21:     }
22:     /*Definicija funkcije koja odredjuje veci od dva broja*/
23:     int max(int x, int y)
24:     {       if(x == y)
25:                     return 0;
26:             else
27:                     return ((x>y)?x:y);
28:     }
```

Анализа програма **primer10_1**

Почетак сваког програма на језику *C* је извршавање прве наредбе његове главне функције *main()*. У примерима до овог главна функција је, да би обавила свој задатак, позивала функције из стандардних *C* библиотека (*printf(), scanf(), getchar(), putchar(), gets(), puts()*…).

Поред стандардних функција, које се испоручују уз сваког *C* преводиоца, главна функција може садржати и позиве функција које програмер пише за конкретан програм. Писање функција, различитих од главне, омогућује велику читљивост, ефикасност и преносивост програма на језику *C*. Изворни кôд сваке функције пише се једном, а може се позвати неограничен број пута из било ког програма на језику *C*.

За разлику од функција из стандардних *C* библиотека, за које је у програму потребно написати само позив, за остале функције стандард *ANSII C* прописује: декларацију (прототип), дефиницију и позив.

Програм у овом примеру одређује већи од два броја задата преко тастатуре. У линији 6. написана је декларација функције која:

- има име *max*,
- користи од остатка програма два аргумента (два целобројна податка *x* и *y*, сваки у опсегу *int*),
- враћа једну вредност (целобројни податак у опсегу *int*) остатку програма.

Декларација једне функције (овај програм је има у линији 6.) пружа преводиоцу информације о броју и типовима аргумената и о типу повратне вредности. Типови аргумената (података које користи функција да би одрадила свој задатак) пишу се између малих заграда, после имена функције (и није обавезно у декларацији писати и имена аргумената). Тип повратне вредности (резултата функције) пише се испред имена функције. Свака функција, позвана у *C* програму може имати нула, један или више аргумената и може имати нула или једну повратну вредност (остатак програма може добити више резултата од функције, али не помоћу повратне вредности, касније ће бити оваквих примера). Функције које не користе податке од остатка програма, имају реч *void* (празно) унутар заграда за листу аргумената.

После потребних декларација променљивих програм тражи са тастатуре два цела броја (слика 10.1 а)), учитава их и позива функцију *max* из линије 14. Функција *max* треба да одреди већи од два цела броја које чувају променљиве *a* и *b* (то су стварни аргументи функције) и да резултат додели променљивој *c*.

Да би се наредба позива функције *max* из линије 14. могла извршити, програму је неопходна дефиниција те функције (линије од 23. до 28.). Дефиниција садржи наредбе по којима функција извршава свој задатак. Линија 23. је прва линија дефиниције (заглавље) функције *max.* Заглавље, за разлику од декларације функције, има обавезно имена аргумената и нема знак за крај наредбе (;), већ следе наредбе функције написане између великих заграда (линије 24. и 28.). Сваки аргумент, написан у заглављу функције (формални аргумент), има тип и неко формално име, које ће даље, у наредбама те функције бити коришћено. Наредбом у линији 24. функција *max* проверава да ли је вредност првог аргумента једнака вредности другог и ако је то случај, помоћу наредбе повратка *return;* враћа вредност 0 главној функцији (линија 25.) тамо где је позвана. У супротном (линија 26.) помоћу наредбе повратка *return;* враћа већи од два аргумента (вредност израза *(x>y)?x:y*).

Дакле, наредба повратка враћа вредност променљиве, константе или неког сложенијег израза оног типа који је наведен као тип повратне вредности функције. (Главна функција нема написан тип повратне вредности, онда је подразумевани тип *int* и због тога може имати на крају наредбу *return 0;.*) Функције које не враћају вредност остатку програма имају реч *void* написану испред имена функције и немају обавезно наредбу *return* (оваква функција може имати само реч *return* без назначене вредности, или може бити без наредбе *return.* Други начин се чешће може видети у програмима).

По позиву функције *max* формални аргументи ове функције (*x* и *y*) добијају вредности стварних аргумената (*a* и *b*). Функција користи стварне вредности аргумената да би извршила свој задатак. У функцији *max* нису могуће измене, већ само коришћење садржаја променљивих (функција не добија од остатка програма адресе променљивих, већ само копије њихових садржаја). Овај начин је предаја функцији аргумената по вредности. Други начин, предаја аргумената по референци, биће објашњен касније.

Када се изврше наредбе функције *max*, до прве наредбе *return,* резултујућа вредност функције (0 или већи од два задата броја) додељује се променљивој *c* (линија 14.). Главна функција онда (линије од 15. до 18.) има проверу вредности променљиве *c* и приказ резултата на екрану (слика 10.1 б)).

Наредба у линији 20. главне функције *(return 0;)* је наредба која је у једноставним програмима на језику *C* (сви до сада примери ове збирке) изостављена. Међутим, сваки озбиљан програм на овом језику има наредбе које предвиђају грешке у разним случајевима и превремени крај извршавања програма ако се десе предвиђене грешке. Могуће су и наредбе за превремени крај извршавања програма под одређеним условом и без појаве грешке. Због ових наредби регуларни програм на језику *C* има као последњу наредбу главне функције наредбу повратка *return 0;*. Оперативни систем под којим се позива програм позива главну функцију, а ако се у међувремену не деси превремени крај извршавања програма, наредбом повратка од главне функције добија статус да је програм извршен до краја.

Слика 10.1 а) Излаз програма **primer10_1** (први део)

Слика 10.1 б) Излаз програма **primer10_1** (други део)

Изворни кôд програма **primer10_2**

```
1:      /*primer10_2.c*/
2:      /*Odredjivanje najveceg od cetiri broja pomocu odgovarajucih funkcija*/
3:      /* (ako ima jednakih brojeva, vazi redosled zadavanja) */
4:
5:      #include  <stdio.h>
6:
7:      int max(int x, int y);
8:      int max4(int x, int y, int z, int w);
9:
10:     main( )
11:     {       int a, b, c, d;
12:
13:             printf("\nZadati cetiri cela broja: ");
14:             scanf("%d%d%d%d", &a, &b, &c, &d);
15:
16:             printf("\nNajveci od cetiri zadata broja je %d\n\n", max4(a,b,c,d));
17:
18:             return 0;
19:     }
```

```
20:    /*Funkcija koja odredjuje veci od dva broja*/
21:    int max(int x, int y)
22:    {       return ((x>=y)?x:y);
23:    }
24:    /*Funkcija koja odredjuje najveci od cetiri broja*/
25:    int max4 (int x, int y, int z, int w)
26:    {       return (max(max(x,y), max(z,w)));
27:    }
```

Анализа програма **primer10_2**

Главна функција држи организацију програма и из ове функције се на посредан или непосредан начин могу позвати све остале функције програма.

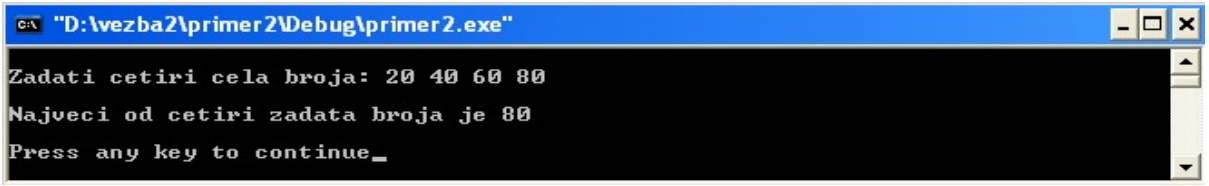
Програм у овом примеру одређује највећи од четири задата цела броја помоћу функција декларисаних у линијама 7. и 8.

У линији 7. написана је декларација функције *max()* која показује да ће функција користити два аргумента типа *int* и вратити резултат типа *int*. У линијама од 21. до 23. написана је дефиниција функције *max()*. Тело функције има само једну наредбу повратка (линија 22.) која помоћу условног оператора враћа већи или први од два једнака задата броја (за разлику од претходног примера).

У линији 8. написана је декларација функције *max4()*, која показује да ће функција користити четири аргумента типа *int* и вратити резултат типа *int*. У линијама од 25. до 27. написана је дефиниција функције *max4()*. Тело функције има само једну наредбу повратка (линија 26.) која враћа резултат функције *max()* за следеће аргументе: један је резултат функције *max()* за први и други задати број (x,y), а други резултат функције *max()* за трећи и четврти задати број (z,w).

Програм из главне функције тражи четири цела броја (слика 10.2 а)) и после учитаних бројева са тастатуре (линија 14.) позива стандардну функцију *printf()* да прикаже на екрану (слика 10.2 б)) резултат функције *max4()* за четири стварна аргумента (*a, b, c* и *d*).

Слика 10.2 а) Излаз програма **primer10_2** (први део)



Слика 10.2 б) Излаз програма **primer10_2** (други део)

Изворни кôд програма **primer10_3**

```
1:    /*primer10_3.c*/
2:    /* Odredjivanje faktorijela n! = n*(n-1) *(n-2) *…*3*2*1 */
3:    /* zadatog prirodnog broja n, pomocu iterativne funkcije */
4:
5:    #include <stdio.h>
6:    #define OPSEG 12          /*dozvoljeni opseg za broj*/
7:                              /*da faktorijel broja bude u opsegu long*/
8:    long ifakt(int x);
9:
10:   main()
11:   {     int broj;
12:
13:         do
14:         {     printf("\nUnesite ceo broj u opsegu od 1 do %d: ", OPSEG);
15:               scanf("%d", &broj);
16:         }while(broj<1 || broj>OPSEG);
17:
18:         printf("\nFaktorijel broja %d je %ld\n\n", broj, ifakt(broj));
19:
20:         return 0;
21:   }
22:   /*Funkcija koja primenom iteracije odredjuje faktorijel broja*/
23:   long ifakt(int x)
24:   {     long rezultat = 1;
25:
26:         while(x>1)
27:         {     rezultat *=x;
28:               x--;
29:         }
30:
31:         return rezultat;
32:   }
```

Анализа програма **primer10_3**

Програм у овом примеру одређује факторијел (*n!=n*(n-1)*(n-2)*…*3*2*1*) задатог целог броја (*n*) итеративном методом (у *while* петљи).

У линији 8. написана је декларација функције *ifakt()*, која показује да ће функција користити један аргумент типа *int* и вратити резултат типа *long*. У линијама од 23. до 32. написана је дефиниција функције *ifakt()*. У овом случају, тело функције има следеће наредбе:

- наредбу декларације и иницијализације променљиве *rezultat* на вредност 1 (линија 24.),
- наредбу *while* петље у линијама од 26. до 29. Све док је вредност аргумента *x* (број од кога се тражи факторијел) већа од 1 следи: формирање нове вредности променљиве *rezultat* множењем претходне

вредности исте променљиве са вредношћу *x* и декремент садржаја променљиве *x* (множи се са x, (x-1), (x-2),…, 2),
- наредбу у линији 31., која враћа вредност променљиве *rezultat.*

Програм у главној функцији тражи унос целог броја за израчунавање факторијела (слика 10.3 а)) све док се уноси број ван траженог опсега. Када се унесе број у опсегу од 1 до 12 (за који је факторијел у опсегу *long*) програм (из линије 18.) приказује на екрану вредност задатог броја и резултат функције *ifakt()* за задату стварну вредност аргумента (слика 10.3 б)). За задати број *5* формални аргумент функције *ifakt()* добија вредност стварног аргумента (5) и у свим итерацијама петље редом реализују се множења: *rezultat = 1∗5∗4∗3∗2.*

Слика 10.3 а) Излаз програма **primer10_3** (први део)

Слика 10.3 б) Излаз програма **primer10_3** (други део)

Изворни кôд програма **primer10_4**

```
1:    /*primer10_4.c*/
2:    /*Izracunavanje faktorijela zadatog broja pomocu rekurzivne funkcije*/
3:
4:    #include <stdio.h>
5:    #define OPSEG 12          /*dozvoljeni opseg za broj*/
6:                              /*da faktorijel broja bude u opsegu long*/
7:    long rfakt(int x);
8:
9:    main()
10:   {     int broj;
11:
12:         do
13:         {     printf("\nUnesite ceo broj u opsegu od 1 do %d: ", OPSEG);
14:               scanf("%d", &broj);
15:         }while(broj<1 || broj>OPSEG);
16:
17:         printf("\nFaktorijel broja %d je %ld\n\n", broj, rfakt(broj));
18:
19:         return 0;
20:   }
21:   /*Funkcija koja primenom rekurzije odredjuje faktorijel broja*/
22:   long rfakt(int x)
23:   {     return ((x>1)? (x*rfakt(x-1)) : 1);
24:   }
```

Анализа програма **primer10_4**

Програм у овом примеру одређује факторијел (*n!=n∗(n-1)∗(n-2)∗…∗3∗2∗1*) задатог целог броја (*n*), помоћу рекурзивне функције. Рекурзивна функција позива саму себе коначан број пута, сваки пут са измењеним аргументима (треба обезбедити у програму услов за излаз из петље са бесконачним бројем рекурзивних позива).

У линији 7. написана је декларација функције *rfakt()*, која показује да ће функција користити један аргумент типа *int* и вратити резултат типа *long*. У линијама од 22. до 24. написана је дефиниција функције *rfakt()*. Тело функције има само једну наредбу повратка помоћу које:

- све док је вредност аргумента већа од 1 функција позива саму себе са аргументом за један мањим од претходно прихваћеног аргумента и враћа резултат овог позива помножен вредношћу аргумента,
- када вредност аргумента више није већа од 1, враћа вредност 1.

Програм из главне функције тражи унос целог броја за израчунавање факторијела (слика 10.4 а)) све док се не унесе број у траженом опсегу (од 1 до 12), за који је факторијел у опсегу *long*.

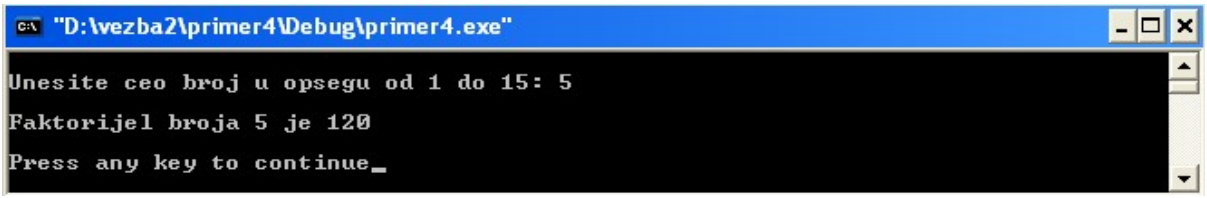
Из линије 17. програм приказује на екрану вредност задатог броја и резултат функције *rfakt()* за задату стварну вредност аргумента (број 5):

- функција printf() позива функцију *rfakt(5),*
- функција *rfakt(5)* позива функцију *rfakt(4)* и враћа вредност: 5∗ *rfakt(4),*
- функција *rfakt(4)* позива функцију *rfakt(3)* и враћа вредност: 4∗ *rfakt(3),*
- функција *rfakt(3)* позива функцију *rfakt(2)* и враћа вредност: 3∗ *rfakt(2),*
- функција *rfakt(2)* позива функцију *rfakt(1)* и враћа вредност: 2∗ *rfakt(1),*
- функција *rfakt(1)* враћа вредност: *1.*

Резултат програма (слика 10.4 б)) је исти као код програма који итеративно решава факторијел (*5∗4∗3∗2∗1*).

Рекурзивна метода даје једноставније и елегантније решење у односу на итеративну методу али захтева велики број позива функције (троши време на позив функције и повратак из функције, као и оперативну меморију за смештање свих потребних информација сачуваних при позиву функције).

Слика 10.4 а) Излаз програма **primer10_4** (први део)



Слика 10.4 б) Излаз програма **primer10_4** (други део)

Изворни кôд програма **primer10_5**

```
1:    /*primer10_5.c*/
2:    /*Izracunavanje vrednosti izraza a!/(b!*(a-b)!)*/
3:    /*pomocu funkcije koja odredjuje faktorijel broja*/
4:
5:    #include <stdio.h>
6:    #define OPSEG 12          /*dozvoljeni opseg za broj*/
7:                              /*da faktorijel broja bude u opsegu long*/
8:    long rfakt(int x);
9:
10:   main()
11:   {       int a, b;
12:
13:           do
14:           {   printf("\nUnesite dva cela broja,u opsegu od 1 do %d,", OPSEG);
15:               printf("\nprvi broj treba da bude veci od drugog:\t");
16:               scanf("%d%d", &a, &b);
17:           }while(a<1 || a>OPSEG || b<1 || a<=b);
18:
19:           printf("\nVrednost izraza  %d!/(%d!*(%d-%d)!) je ",a,b,a,b);
20:           printf("%ld\n\n", rfakt(a)/(rfakt(b)*rfakt(a-b)));
21:
22:           return 0;
23:   }
24:   /*Funkcija koja primenom rekurzije odredjuje faktorijel broja*/
25:   long rfakt(int x)
26:   {       return ((x>1)? (x*rfakt(x-1)) : 1);
27:   }
```

Анализа програма **primer10_5**

Програм у овом примеру одређује вредност израза: *a!/(b!*(a-b)!)* за задате целе бројеве *a* и *b* (слика 10.5 а)). Користи се већ описана функција за израчунавање факторијела броја (декларација у линији 8. и дефиниција у линијама од 25. до 27.) позвана за различите вредности аргумената.

Да би на екрану била приказана вредност поменутог израза (слика 10.5 б)) главна функција има три позива исте рекурзивне функције: за задати број *a* (позив *rfakt(a))*, за задати број *b* (позив *rfakt(b))* и за разлику задатих бројева *a* и *b (*позив *rfakt(a-b)),* као и примену тражених оператора над резултатима појединих позива функција (линија 20.).

Слика 10.5 а) Излаз програма **primer10_5** (први део)

Слика 10.5 б) Излаз програма **primer10_5** (други део)

Изворни кôд програма **primer10_6**

```
1:    /*primer10_6.c*/
2:    /*Postavljanje vrednosti bita na zadatoj poziciji*/
3:    /*u zadatom broju, pomocu odgovarajucih funkcija*/
4:
5:    #include <stdio.h>
6:    #define JEDAN_BAJT      8       /*broj bitova u bajtu*/
7:    #define BAJTOVA         2       /*opseg od dva bajta*/
8:    #define MASKA           0xffff  /*svi bitovi maske imaju vrednost 1*/
9:
10:   unsigned bit_set(unsigned x, unsigned y);
11:   unsigned bit_reset(unsigned x, unsigned y);
12:
13:   main()
14:   {       unsigned broj, poz, set;
15:
16:           while(1)
17:           {  printf("\nUnesite pozitivan ceo broj (opseg %d bajta),", BAJTOVA);
18:              printf("\nu heksadecimalnom formatu:\t\t");
19:              scanf("%x", &broj);
20:
21:              if(broj>MASKA)
22:                   break;
23:              do
24:              {    printf("\nUnesite poziciju bita,");
25:                   printf("\nredni broj od bita najmanje tezine:\t");
26:                   scanf("%u", &poz);
27:              }while(poz>JEDAN_BAJT*BAJTOVA);
28:              do
29:              {    printf("\nUnesite vrednost bita (1 ili 0):\t");
30:                   scanf("%u", &set);
31:              }while(set<0 || set>1);
32:
33:              if(set)
34:                   printf("\nRezultujuci broj:\t\t\t%#x\n", bit_set(broj, poz));
35:              else
36:                   printf("\nRezultujuci broj:\t\t\t%#x\n", bit_reset(broj, poz));
37:           }
38:           return 0;
39:   }
```

```
40:   /*Funkcija koja dodeljuje vrednost 1 bitu na zadatoj poziciji u broju*/
41:   unsigned bit_set(unsigned x, unsigned y)
42:   {       return (x | (1 << (y-1)));
43:   }
44:   /*Funkcija koja dodeljuje vrednost 0 bitu na zadatoj poziciji u broju*/
45:   unsigned bit_reset(unsigned x, unsigned y)
46:   {       return (x & (MASKA ^ (1 << (y-1))));
47:   }
```

Анализа програма **primer10_6**

Програм у овом примеру користи одговарајуће функције да постави на вредност 1 *(set)* или на вредност 0 *(reset)* задати бит у задатом броју.

У линијама 10. и 11. написане су декларације функција *bit_set()* и *bit_reset(),* које показују да ће свака од њих користити два аргумента типа *int* и враћати податак типа *unsigned*.

У линијама од 41. до 43. написана је дефиниција функције *bit_set()*. Тело функције има само једну наредбу за повратак вредности израза: *x | (1 << (y-1)).* Први аргумент функције (*x)* је број коме се поставља бит на вредност 1, а други (*y)* је редни број бита (од бита најмање тежине). За ово се користи оператор логички ИЛИ на нивоу бита (|).

У линијама од 45. до 47. написана је дефиниција функције *bit_* reset *()*. Тело функције има само једну наредбу за повратак вредности следећег израза: *x & (MASKA ^ (1 << (y-1))).* У овом изразу, први аргумент функције (*x)* је број у коме се поставља бит на вредност 0, а други (*y)* је редни број бита (од најмање тежине). За ово се користе оператор логички искључиво ИЛИ на нивоу бита (^) и маска (0xfff).

Програм у главној функцији (линије од 17.) тражи унос позитивног целог броја у опсегу два бајта, редни број бита и команде (1 за *set* или 0 за *reset*).

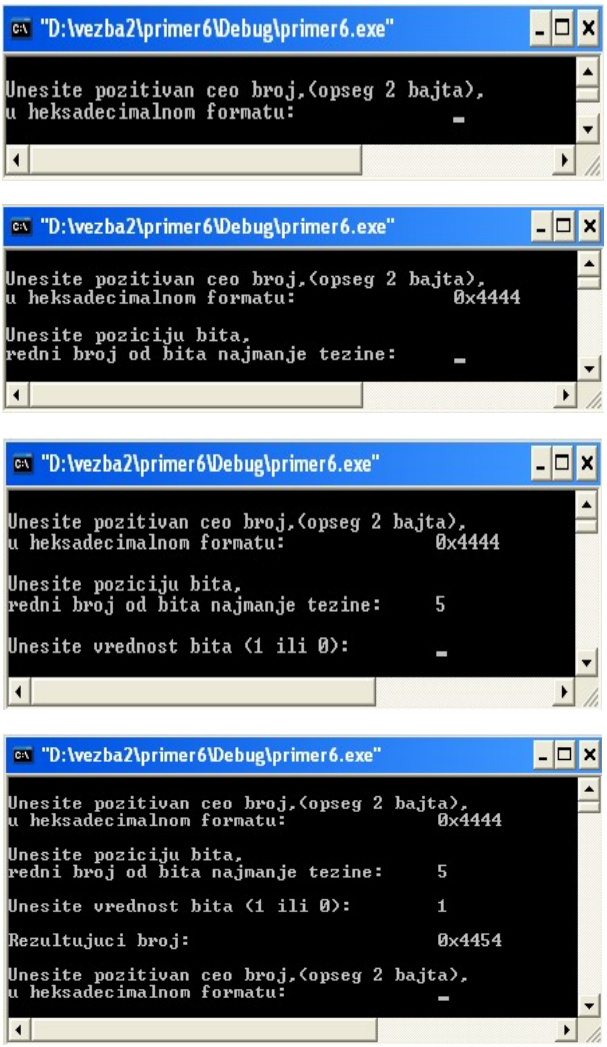
За задату команду, следи позив одговарајуће функције (линија 34. или линија 36.) и предаја функцији стварних аргумената: *bro*j (позитиван број у опсегу два бајта) и *poz(*редни број бита*)*, као и приказ резултата на екрану.

За изабрану команду сет, вредност броја 0x4444 и позицију бита 5, добија се вредност израза *x | (1 << (y-1))* као на слици 10.6 а):

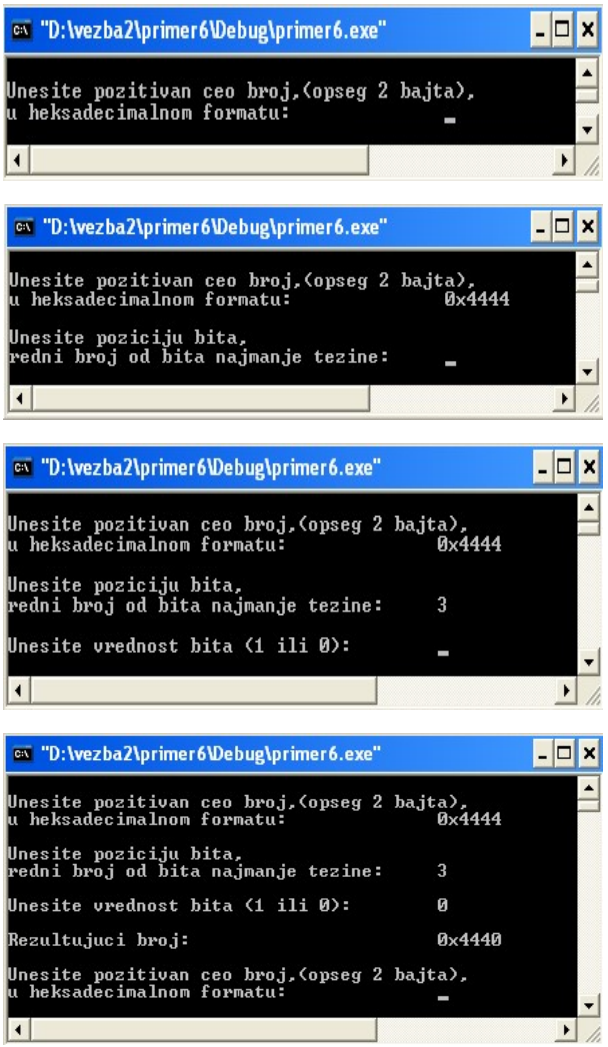
```
        01000100 01000100      (0x4444)
|       00000000 00010000      (1 << (5-1))
        01000100 01010100      (0x4454)
```

За изабрану команду ресет, вредност броја 0x4444 и позицију бита 3, добија се вредност израза *x & (MASKA ^ (1 << (y-1)))* као на слици 10.6 б):

```
        01000100 01000100      (0x4444)
&       11111111 11111011      (0xffff  ^ (1 << (3-1))
        01000100 01000000      (0x4440)
```



Слика 10.6 а) Излаз програма **primer10_6** (први део)



Слика 10.6 б) Излаз програма **Primer10_6** (други део)

Изворни кôд програма **primer10_7**

```
1:    /*primer10_7.c*/
2:    /*Pomeranje bitova u zadatom broju, za zadati broj mesta,*/
3:    /*u zadatom smeru i pomocu odgovarajucih funkcija*/
4:
5:    #include <stdio.h>
6:    #include <ctype.h>
7:    #define JEDAN_BAJT      8       /*broj bitova u bajtu*/
8:    #define BAJTOVA         2       /*opseg od dva bajta*/
9:    #define OPSEG           0xffff  /*svi bitovi maske imaju vrednost 1*/
10:
11:   unsigned pomeren(unsigned x, unsigned y, unsigned z);
12:
13:   main()
14:   {  unsigned int broj, pomeraj, smer;
15:
16:      while(1)
17:      { do
18:        {    printf("\nUnesite pozitivan ceo broj (opseg %d bajta),", BAJTOVA);
19:             printf("\nu heksadecimalnom formatu (0 za kraj):\t");
20:             scanf("%x", &broj);
21:        }while(broj>OPSEG);
22:
23:        if(broj==0)
24:             break;
25:        do
26:        {    printf("\nUnesite broj mesta za pomeranje:\t");
27:             scanf("%u", &pomeraj);
28:             getchar();
29:        }while(pomeraj>=JEDAN_BAJT*BAJTOVA);
30:        do
31:        {    printf("\nZadati smer slovom: D(d) ili L(l):\t");
32:             smer = toupper(getchar());
33:             getchar();
34:        }while(smer != 'D' && smer != 'L');
35:
36:        printf("\nRezultujuci broj:\t\t\t%#x\n", pomeren(broj, pomeraj, smer));
37:      }
38:      return 0;
39:   }
40:   /*Funkcija koja bitove zadatog broja pomera*/
41:   /*za zadati broj mesta u zadatom smeru*/
42:   unsigned pomeren(unsigned x, unsigned y, unsigned z)
43:   {       return ((z=='D')?(x>>=y):(x<<=y));
44:   }
```

Анализа програма **primer10_7**

Програм у овом примеру користи одговарајућу функцију да помери битове задатог броја за задати број помераја у задатом смеру.

У линији 11. написана је декларација функције *pomeren(),* која показује да ће функција користити три аргумента типа *int* и вратити резултат типа *int*.

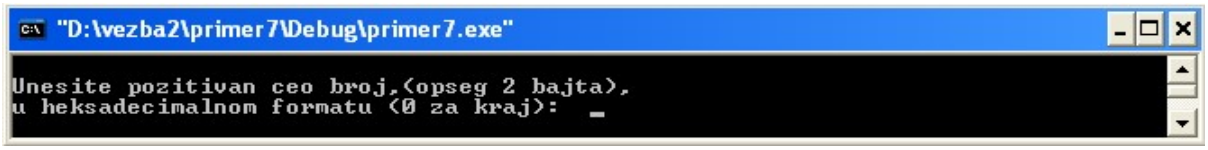
У линијама 42. до 44. написана је дефиниција функције. Функција проверава да ли је смер, који је одређен трећим аргументом функције *(z),* десно *('D')* или не и у зависности од тога помера битове првог аргумента *(x)* за број помераја који је одређен другим аргументом функције *(y)*. За ову проверу искоришћен је условни оператор (линија 43.). Тако измењен број *(x)* функција враћа остатку програма.

Програм из главне функције тражи унос позитивног целог броја у опсегу два бајта (*broj*), број места за померање битова (*pomeraj*) и смер померања (*smer*) (слике 10.7 а), 10.7 б) и 10.7 в)). Онда (из линије 36.) приказује на екрану резултат позива функције *pomeren()*, за вредности стварних аргумената (*broj, pomeraj* и *smer*). Резултујућа вредност функције приказује се са ознаком за хексадецимални формат испред вредности броја (%#x).

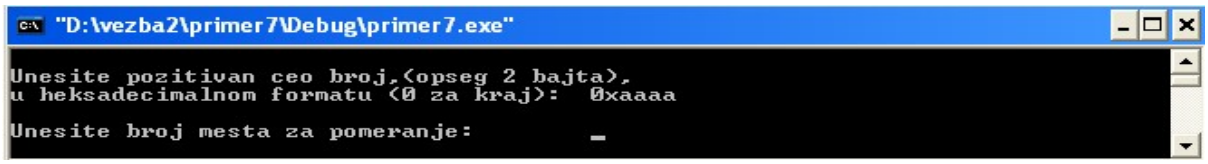
За изабрани број 0xaaaa, број помераја 3 и смер десно (‘*D’*), на екрану се приказује резултујућа вредност функције као на слици 10.7 г):

10101010 10101010 (0xaaaa)
00010101 01010101 (0xaaaa >> 3: ⇔ 0x1555)

Све описане наредбе главне функције смештене су у петљу *while(1)* (линије од 16. до 37.) и понављају се све док се не зада број 0 (линије 23. и 24.)

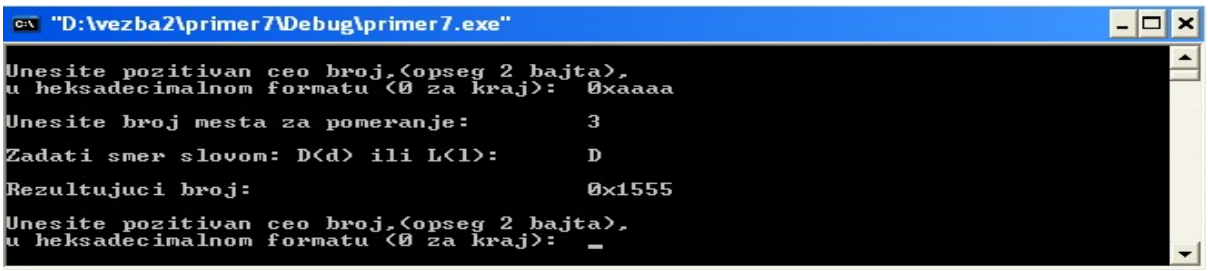


Слика 10.7 а) Излаз програма **primer10_7** (први део)



Слика 10.7 б) Излаз програма **primer10_7** (други део)

Слика 10.7 в) Излаз програма **primer10_7** (трећи део)



Слика 10.7 г) Излаз програма **primer10_7** (четврти део)

Изворни кôд програма **primer10_8**

```
1:    /*primer10_8.c*/
2:    /*Crtanje na ekranu okvira zadatih dimenzija.*/
3:
4:    #include <stdio.h>
5:    #define HLIN        '-'     /*znak horizontalna crta okvira*/
6:    #define VLIN        '|'     /*znak vertikalna crta okvira*/
7:    #define UGAO        '+'     /*znak za ugao okvira*/
8:    #define PRAZNO      ' '     /*prazan znak*/
9:    #define MAX1        50      /*dozvoljena duzina okvira*/
10:   #define MAX2        20      /*dozvoljena sirina okvira*/
11:
12:   void hokvir(int l);
13:   void vokvir(int l);
14:   void znakovi(int n, char c);
15:
16:   main()
17:   {       int i, duzina, sirina;
18:
19:           do
20:           {       printf("\nUnesite duzinu okvira (od 1 do %d): ",MAX1);
21:                   scanf("%d", &duzina);
22:           }while(duzina<1 || duzina>MAX1);
23:
24:           do
25:           {       printf("\nUnesite sirinu okvira (od 1 do %d): ",MAX2);
26:                   scanf("%d", &sirina);
27:           }while(sirina<1 || sirina>MAX2);
28:
29:           /*Stampanje gornjeg dela okvira*/
30:           hokvir(duzina);
31:
32:           /*Stampanje srednjeg dela okvira*/
33:           for(i=0; i<sirina; i++)
34:                   vokvir(duzina);
35:
36:           /*Stampanje donjeg dela okvira*/
37:           hokvir(duzina);
38:
39:           return 0;
40:   }
41:   /*Funkcija koja crta horizontalni deo okvira*/
42:   void hokvir(int l)
43:   {       putchar(UGAO);
44:           znakovi(l, HLIN);
45:           putchar(UGAO);
46:           putchar('\n');
47:   }
```

```
48:   /*Funkcija koja crta jedan red vertikalnog dela okvira*/
49:   void vokvir(int l)
50:   {       putchar(VLIN);
51:           znakovi(l, PRAZNO);
52:           putchar(VLIN);
53:           putchar('\n');
54:   }
55:   /*Funkcija koja prikazuje zadati znak zadati broj puta*/
56:   void znakovi(int n, char c)
57:   {       while(n>0)
58:           {       n--;
59:                   putchar(c);
60:           }
61:   }
```

Анализа програма **primer10_8**

Програм у овом примеру црта на екрану оквир димензија задатих преко тастатуре. Поједини делови главног задатка програма (цртање хоризонталног и вертикалног дела оквира издвојени су у посебне функције.

У линијама 12., 13. и 14. написане су декларације функција *hokvir()*, *vokvir()* и *znakovi()* које показују следеће:

- функција *hokvir()* користиће један аргумент типа *int* (дужину оквира) и неће враћати вредност остатку програма (већ ће само цртати хоризонтални део оквира на екрану),
- функција *vokvir()* користиће један аргумент типа *int* (ширину оквира) и неће враћати вредност остатку програма (већ ће само цртати вертикални део оквира на екрану),
- функција *znakovi()* користиће један аргумент типа *int* (број знакова) и један типа *char* (*ASCII* кôд знака који треба да прикаже).

У линијама 42. до 47. написана је дефиниција функције *hokvir()*. Ова функција прво црта на екрану леви угао оквира тако што приказује знак '+' (дефинисан у линији 7. на почетку програма), онда позива функцију *znakovi()* да нацрта хоризонталну линију потребне дужине (линија 44.), црта десни угао оквира (линија 45.) и одрађује прелаз у нови ред (линија 46.).

У линијама од 49. до 54. написана је дефиниција функције *vokvir()*. Ова функција приказује на екрану редом: један знак вертикална црта за један вертикални део оквира (линија 50.), онда позива функцију *znakovi()* да одвоји ред празних знакова дужине која је потребна између вертикалних црта оквира (линија 51.), приказује на екрану један знак вертикална црта за други вертикални део оквира (линија 52.) и одрађује прелаз у нови ред (линија 53.).

У линијама од 56. до 61. написана је дефиниција функције *znakovi()*, која приказује на екрану *n* задатих знакова (вредност првог аргумента је број знакова а другог *ASCII* кôд знака).

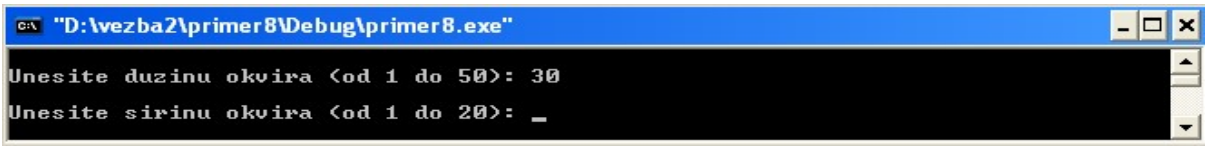
Програм у главној функцији тражи дужину (линије од 19. до 22.) и ширину (линије од 24. до 27.) оквира (слике 10.8 а) и 10.8 б)), а онда :

- позива једном функцију *hokvir(),* која црта горњи хоризонтални део оквира за стварну вредност дужине оквира (линија 30.),

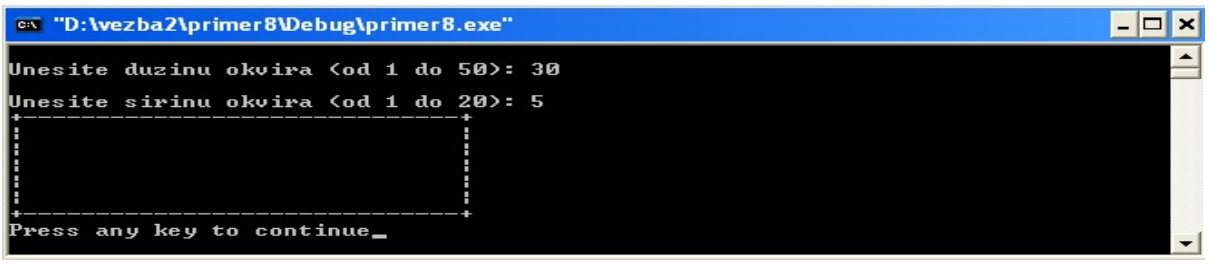
- позива функцију *vokvir()* (линије 33. и 34.) онолико пута колика је стварна вредност ширине оквира,
- позива још једном функцију *hokvir(),* која црта доњи хоризонтални део оквира за стварну вредност дужине оквира (линија 37.).

Излаз програма може се видети на слици 10.8 в).

Слика 10.8 а) Излаз програма **primer10_8** (први део)



Слика 10.8 б) Излаз програма **primer10_8** (други део)



Слика 10.8 в) Излаз програма **primer10_8** (трећи део)

Практични савети

- ✓ Функција треба да има име које добро описује њен задатак. Ово доприноси доброј документованости програма.
- ✓ Декларације функција писати ван дефиниција функција (глобално), јер се оне у модуларним програмима (дужим програмима распоређеним у више датотека) смештају у посебне датотеке.
- ✓ Прва линија дефиниције функције може се добити копирањем декларације (прототипа) исте функције и брисањем знака тачка-зарез.

Евентуалне грешке

- ➢ Када се аргумент предаје по вредности функција добија само копију његовог садржаја и нема могућност измене оригиналног садржаја, без наредбе повратка.
- ➢ Дефиниција једне функције не може бити написана унутар дефиниције неке друге функције.
- ➢ Ако више аргумената функције има исти тип није дозвољена замена листе аргумената (*int x, int)* листом (*int x, y)*.
- ➢ Наредбом повратка функција може вратити само једну вредност. Може бити више наредби *return*, али само једна од њих може се извршити.

Задаци за припрему вежбе 10. у лабораторији

Задатак 1.

Написати декларацију функција на језику *C*:
а) *f1*, која користи два аргумента типа *int* и враћа вредност типа *float*,
б) *f2*, која користи три аргумента типа *float* и не враћа вредност.

Задатак 2.

Исправити синтаксне грешке у дефиницији функције на језику *C*:

```
void duplo_vece(int x);
{       return (2*x);           }
```

Задатак 3.

Исправити синтаксне грешке у изворном кôду програма на језику *C:*

```
#include <stdio.h>
void stampa_poruku(void);
main()
{       stampa_poruku("\nU ovom kodu postoje sintaksne greske"); }
void stampa_poruku(void);
{       printf("\nU ovom kodu postoje sintaksne greske");
        return 0;                                                }
```

Задатак 4.

Написати програм на језику *C* који помоћу одговарајуће функције приказује на екрану количник од два цела броја *a* и *b* са *k* децимала ако је број различит од 0, или одговарајућу поруку ако је број *b* једнак 0 (бројеве *a*, *b* и *k* програм прихвата са тастатуре).

Задаци за вежбу 10. у лабораторији

Задатак 1.

Написати програм на језику *C* који одређује највећи и најмањи од три цела броја задата преко тастатуре, помоћу функција *min*(x,y) и *max*(x,y).

Задатак 2.

Написати програм на језику *C* који помоћу одговарајућих функција приказује на екрану бинарни облик целог броја задатог преко тастатуре: у оригиналном и у инверзном поретку (замењена места битовима: првом и последњем, другом и претпоследњем...).

Задатак 3.

Написати програм на језику *C* који из једног реда текста дужине *n* (*n*≤80) извлачи први знак који није слово или децимална цифра и помоћу одговарајуће функције приказује на екрану троугао од десет редова ових знакова, у првом реду један знак, у другом два знака..., уз лево или десно поравнање троугла знакова, у зависности од опције задате преко тастатуре (L или D).

Задатак 4.

Написати програм на језику *C* који израчунава вредност суме:

$$S = 1 + \frac{1}{2!!} - \frac{1}{3!!} + \frac{1}{4!!} - \ldots$$

помоћу функције:

$$dupli\_fakt(n) = n!! = \begin{cases} n*(n-2)*\ldots*3*1, za\_neparno\_n \\ n*(n-2)*\ldots*4*2, za\_parno\_n \end{cases}$$

до члана који је по апсолутној вредности мањи од броја, задатог преко тастатуре.

Задатак 5.

Написати програм на језику *C* који израчунава вредност израза:

F(k,x) = f(f(…f(x)…)).

Функција *f* позива се *k* пута и облика је: $f(x) = 2^x - x^2$. Вредности *x* и *k* програм прихвата са тастатуре.

Решени задаци на програмском језику *C*

Задатак 1.

Написати програм на језику *C* који одређује и приказује на екрану приближну вредност интеграла (1)

$$(1) \quad f(x) = \int_0^{+\infty} \frac{e^{-t}}{1 + xt} dt$$

применом сумирања асимптотског реда

$$(2) \qquad f(x) = 1 - 1!x + 2!x^2 - 3!x^3 + \ldots = \sum_{n=0}^{+\infty} (-1)^n n! x^n$$

за целобројне вредности броја *n*. Ред (2) је дивергентан за све вредности *x* изузев за *x=0* и због тога се приближна вредност интеграла одређује сумирањем по апсолутној вредности опадајућих чланова реда $\left(|x_n| > |x_{n+1}|,\ x_n = (-1)^n n! x^n\right)$ и додавањем половине вредности првог елемента реда, који по апсолутној вредности расте. Тестирати рад програма за вредности *x* задате преко тастатуре (0<*x<1)*.

Решење 1.

```
/*Program za odredjivanje priblizne vrednosti zadatog integrala*/
#include <stdio.h>

main()
{
        double x,s=1,znak=-1,preth;
        int i;

        do
        {       printf("Unesite vrednost za x (0<x<1)\n");
                scanf("%lf",&x);
        }while(x<0||x>1);

        s+=znak*x;
        preth=x;
        i=2;

        while(1)
        {       znak*=-1;
                if(preth<preth*x*i) break;

                preth*=x;
                s+=znak*preth;

                i++;
        }
        s+=znak*preth*i/2;
        printf("\ns=%lf\n\nRezultat je dobijen sumiranjem prvih %d clanova.\n",s,i);
}
```

Задатак 2.

Написати програм на језику *C* који, за четири податка задата преко тастатуре (*p1, p2, p3* и *p4*), сваки величине два бајта, рачуна вредност *disk parity* (*dp*) на следећи начин:

$$dp = p1 \oplus p2 \oplus p3 \oplus p4$$

где је $\oplus$ операција *XOR* (искључиво или) и

а) приказује на екрану вредности свих задатих података, као и резултујућу вредност у децималном и бинарном облику,

б) тражи од корисника да зада нову вредност податка *p3*, приказује на екрану ову вредност, а затим реконструише и приказује на екрану претходну вредност податка *p3* на основу вредности *dp* и вредности података *p1, p2* и *p4* (*p3 = dp* $\oplus$ *p1* $\oplus$ *p2* $\oplus$ *p4*),

в) израчунава и приказује на екрану нову вредност *disk parity* (*dp)* добијену на основу старе вредности *dp* и нове вредности *p3* (*dp = dp* $\oplus$ *p3*).

Решење 2.

```
/*Program za izracunavanje vrednosti disk parity na osnovu zadatih
  odgovarajucih podataka*/
#include <stdio.h>

main()
{       unsigned short int p1,p2,p3,p4,dp,preth,i;

        printf("Unesite cetiri prirodna broja\n");
        scanf("%u%u%u%u",&p1,&p2,&p3,&p4);

        printf("p1= %d u binarnom zapisu ",p1);
        for(i=0;i<16;i++)
                printf("%d",(p1>>(15-i))&1);
        printf("\n");

        printf("p2= %d u binarnom zapisu ",p2);
        for(i=0;i<16;i++)
                printf("%d",(p2>>(15-i))&1);
        printf("\n");

        printf("p3= %d u binarnom zapisu ",p3);
        for(i=0;i<16;i++)
                printf("%d",(p3>>(15-i))&1);
        printf("\n");

        printf("p4= %d u binarnom zapisu ",p4);
        for(i=0;i<16;i++)
                printf("%d",(p4>>(15-i))&1);
        printf("\n");
```

```
        /*izracunavanje vrednosti disk parity*/
        dp=p1^p2^p3^p4;

        printf("disk parity= %d u binarnom zapisu ",dp);
        for(i=0;i<16;i++)
                printf("%d",(dp>>(15-i))&1);
        printf("\n\n");

        printf("Unesite novu vrednost za p3: ");
        scanf("%u",&p3);

        printf("Nova vrednost promenljive p3 je %u.\n",p3);
        /*rekonstrukcija izgubljene vrednosti iz ostalih vrednosti i dp*/
        preth=dp^p1^p2^p4;

        printf("\nStara vrednost promenljive p3");
        printf("\nrekonstruisana iz disk parity i p1,p2 i p4 je %u.\n",preth);

        /*izracunavanje i prikaz nove vrednosti dp*/
        printf("\nNova vrednost disk parity ");
        printf("posle promene vrednosti p3 je %d\n",dp=dp^p3);
}
```

Задатак 3.

Написати програм на језику *C* који приказује на екрану у бинарном облику вредност податка *x* (величине два бајта) задату преко тастатуре, као и податка *y* који се добија тако што се инвертује *n* битова у податку *x,* почевши од позиције *p,* док остали битови остају непромењени. На пример:

x=0000 1001 1101 1111*,* *p*=3, *n*=5
y=0000 1001 0010 0111*.*

Решење 3.

```
/*Program za invertovanje odredjenih bitova u zadatom podatku*/
#include <stdio.h>

main()
{       unsigned short x,y,i,p,n,pom;

        printf("Unesite vrednost promenljive x:");
        scanf("%u",&x);

        printf("x= %d u binarnom zapisu x=",x);
        for(i=0;i<16;i++){
                if(i%4==0) printf(" ");
                printf("%d",(x>>(15-i))&1);
        }
        printf("\n");

        do
        {       printf("Pozicija i broj bitova za invertovanje (0<p<16 i n+p<16)");
                scanf("%u%u",&p,&n);
        }while(p<0||p>15||p+n>15);

        pom=0xfff;
        pom>>=p;
        pom<<=p;
        pom<<=(16-p-n);
        pom>>=(16-p-n);
        y=(x^pom);

        printf("y= %d u binarnom zapisu y=",y);
        for(i=0;i<16;i++){
                if(i%4==0) printf(" ");
                printf("%d",(y>>(15-i))&1);
        }
        printf("\n\n");
}
```

Задатак 4.

Написати програм на језику C који израчунава и приказује на екрану вредности функције у некој тачки из задатог интервала, помоћу Lagranzovog полинома. Функција се не задаје експлицитно формулом, већ својим вредностима у одређеним тачкама. Тачке, као и вредности функције у тим тачкама задају се преко тастатуре. Вредност функције у било којој тачки из задатог интервала добија се по формули:

$$L_n = \sum_{i=1}^{n} \frac{\prod_{\substack{j=1 \\ j\neq i}}^{n} (x - x_j)}{\prod_{\substack{j=1 \\ j\neq i}}^{n} (x_i - x_j)}$$ (Lagranžov полином)

Решење 4.

```
/*Program za izracunavanje vrednosti funkcije u nekoj tacki iz zadatog
  intervala, pomocu Lagranzovog polinoma*/
#include<stdio.h>
#define MAX 20

main()
{       float cvor[MAX],vrednost[MAX],tacka,s,p;
        int n,imin=0,imax=0,i,j;

        /*unosi se broj tacaka*/
        while(1)
        {       printf("\nKoliko ima tacaka (najvise %i)? ",MAX);
                scanf("%i",&n);

                if(n>0 && n<=MAX) break;
                printf("\nVan opsega");
        }

        /*unos tacaka i vrednosti funkcije u tim tackama*/
        printf("\nZadati tacke i vrednosti funkcije u datim tackama\n");
        for(i=0;i<n;i++)
        {       printf("\n%i. tacka i vrednost funkcije u datoj tacki:\n ",i+1);
                scanf("%f%f",&cvor[i],&vrednost[i]);

                for(j=0;j<i;j++)
                        if(cvor[i]==cvor[j])
                        {       printf("\nTacka vec postoji\n");i--;break;}

                /*trazenje gornje i donje granice intervala*/
                if(cvor[i]>cvor[imax]) imax=i;
                if(cvor[i]<cvor[imin]) imin=i;
        }
```

```
        while(1)
        {       printf("\nUneti tacku iz intervala (%.2f,%.2f)", cvor[imin],cvor[imax]);
                printf(" u kojoj se izracunava vrednost funkcije ");
                scanf("%f",&tacka);

                if(tacka<cvor[imin] || tacka>cvor[imax])
                {       printf("\nTacka se nalazi van zadatog intervala\n");break;
                }

                s=0.0;
                for(i=0;i<n;i++)
                {       p=1.0;
                        for(j=0;j<n;j++)
                                if(i!=j) p*=(tacka-cvor[j])/(cvor[i]-cvor[j]);

                        s+=p*vrednost[i];
                }

                printf("\nTrazena vrednost je %f\n",s);
        }
}
```

Задатак 5.

Написати програм на језику *C* који регулише осветљеност на следећи начин: у једном 16-битном регистру сваки бит користи се за контролу по једне од 16 сијалица: када је вредност бита 1 сијалица светли, а када је вредност бита 0 она не светли. Програм треба да прикаже на екрану садржај поменутог регистра у бинарном облику, за сваки од захтева:

а) светле све сијалице,
б) светли свака друга сијалица,
в) светли група сијалица,
г) не светли ни једна сијалица.

Решење 5.

```
/*Program za kontrolu osvetljenosti pomocu bitova odgovarajuceg registra*/
#include <stdio.h>

main()
{       unsigned short int registar,i, m;

        printf("Pocetni (slucajan) sadrzaj registra:\t");
        for(i=0;i<16;i++)
        {       if(i%4==0)
                        printf(" ");
                printf("%d",(registar>>(15-i))&1);
        }

        /*Svetle sve sijalice*/
        printf("\nSvetle sve sijalice:\t\t\t");
        registar=registar|(~registar);

        for(i=0;i<16;i++)
        {       if(i%4==0)
                        printf(" ");
                printf("%d",(registar>>(15-i))&1);
        }

        /*Svetli svaka druga sijalica*/
        for(i=0;i<4;i++)
        {       registar<<=4;
                registar|=10;
        }
        printf("\nSvetli svaka druga sijalica:\t\t");
        for(i=0;i<16;i++)
        {       if(i%4==0)
                        printf(" ");
                printf("%d",(registar>>(15-i))&1);
        }
```

```
	/*Svetli grupa od tri sijalice pocevsi od m-te*/
	do
	{	printf("\n\nUnesite m: ");
		scanf("%d", &m);
	}while(m>16 || m<0);

	registar&=(7<<(m-1));
	registar|=(7<<(m-1));

	printf("\nSvetle tri sijalice pocevsi od %d.:\t",m);
	for(i=0;i<16;i++)
	{	if(i%4==0)
			printf(" ");
		printf("%d",(registar>>(15-i))&1);
	}

	/*Ne svetli ni jedna sijalica*/
	registar^=registar;

	printf("\nNe svetli ni jedna sijalica:\t\t");
	for(i=0;i<16;i++){
		if(i%4==0)
			printf(" ");
		printf("%d",(registar>>(15-i))&1);
	}
	printf("\n\n");
}
```

Задатак 6.

Написати програм на језику *C* који исписује на екрану листу операција дозвољених над задатим полиномима и у зависности од избора опције преко тастатуре извршава једну операцију и приказује на екрану резултат. Понуђена листа треба да има следеће опције:

+) Сабирање полинома
-) Одузимање полинома
*) Множење полинома
') Извод полинома
x) Вредност полинома у тачки
i) Излаз из програма

Решење 6.

```
/*Program za realizaciju dozvoljenih operacija nad zadatim polinomima*/
#include<stdio.h>
#include<ctype.h>
#define MAX 20

main()
{       float prvi[MAX+1],drugi[MAX+1],rezultat[2*MAX+1],s,x;
        int n,m,i,j,znak,p,izvod;
        char opcija;

        while(1)
        {       printf("\n\nOvo je program za rad sa polinomima\n");
                printf("\n+) Sabiranje polinoma");
                printf("\n-) Oduzimanje polinoma");
                printf("\n*) Mnozenje polinoma");
                printf("\n') Izvod polinoma");
                printf("\nx) Vrednost polinoma u tacki");
                printf("\ni) Izlaz");
                printf("\n\nIzaberite opciju  ");

                opcija=tolower(getchar());
                if(opcija=='i') break;
                if(     opcija !='+' && opcija !='-' &&
                        opcija !='*' && opcija !='\'' && opcija !='x')
                {       printf("\nNepoznata opcija\n");getchar();continue;       }

                for(i=0;i<=MAX;i++)
                        prvi[i]=drugi[i]=rezultat[i]=rezultat[MAX+i]=0;

                /*stepen prvog polinoma*/
                do
                {       printf("\nUnesite stepen prvog polinoma (<=%i)",MAX);
                        scanf("%i",&n);
                }while (n<0 || n>20);
```

```
/*koeficijenti prvog polinoma*/
printf("\nUnesite koeficijente prvog polinoma\n");
for(i=n;i>=0;i--)
{       printf("koeficijent uz %i. stepen: ",i);
        scanf("%f",&prvi[i]);

        if(prvi[n]==0)
        {       printf("\nVodeci koeficijent ne sme da bude 0\n");i++;}
}
if(opcija=='+' || opcija=='-' || opcija=='*')
{       //stepen drugog polinoma
        do
        {       printf("\nUnesite stepen drugog polinoma (<=%i)",MAX);
                scanf("%i",&m);
        }
        while (m<0 || m>MAX);

/*koeficijenti drugog polinoma*/
printf("\nUnesite koeficijente drugog polinoma\n");
for(i=m;i>=0;i--)
{       printf("koeficijent uz %i. stepen: ",i);
        scanf("%f",&drugi[i]);

        if(drugi[m]==0)
        {       printf("\nVodeci koeficijent ne sme da bude 0\n");
                i++;}
        }
}
switch(opcija)
{case '+': case '-':
        if(opcija=='+') znak=1; else znak=-1;
        p=(m>n?m:n);
        //racunanje zbira-razlike
        for(i=0;i<=p;i++)
                rezultat[i]=prvi[i]+znak*drugi[i];
        break;
case '*':
        for(i=0;i<=n;i++)
                for(j=0;j<=m;j++)
                        rezultat[i+j]+=prvi[i]*drugi[j];
        p=m+n;
        break;
case '\'':
        p=n;
        printf("\nKoji izvod po redu: ");
        scanf("%i",&izvod);
        if(izvod>n) {p=0;rezultat[p]=0;
                break;
        }
```

```
                        for(i=p;i>=0;i--)
                                rezultat[i]=prvi[i];
                        for(j=1;j<=izvod;j++)
                        {       p--;
                                for(i=0;i<=p;i++)
                                        rezultat[i]=rezultat[i+1]*(i+1);
                        }
                                break;
                case 'x':
                        printf("\nUnesite tacku u kojoj se racuna vrednost polinoma ");
                        scanf("%f",&x);
                        s=prvi[n];

                        for(i=n-1;i>=0;i--)
                                s=s*x+prvi[i];
                        rezultat[0]=s;

                        p=0;
                        break;
                default:
                        break;
        }

        /*stampanje rezultata*/
        printf("\nRezultat je:\n");
        for(i=p;i>0;i--)
                printf("(%.2f)*x^%i+",rezultat[i],i);
        printf("(%.2f)",rezultat[0]);

        fflush(stdin);
        }
}
```

Задатак 7.

Написати програм на језику *C* који сортира времена задата преко тастатуре (у облику: година, месец, дан, сат, минут, секунда) од најстаријег до најновијег или обрнуто (у зависности од изабране опције преко тастатуре) и приказује на екрану резултат овог сортирања.

Решење 7.

```
/*Program za sortiranje zadatih vremena*/
/*Program pakuje vreme u 32 bita na sledeci nacin, od bita najvece tezine redom:*/
/*godina: 6 bitova (uzimaju se u obzir posledenje 2 cifre u godini)*/
/*mesec (<=12): 4 bita*/
/*dan (<=31):  5 bitova*/
/*sat (<24): 5 bitova*/
/*minute (<60): 6 bitova*/
/*sekunde(<60): 6 bitova*/

#include<stdio.h>
#include<ctype.h>
#define MAX 100

main()
{       unsigned int vremena[MAX]={0},pom,ispravno,n,i,j;
        unsigned int godina,mesec,dan,sat,minut,sekunda, prestupna;
        int z;
        char opcija;

        do
        {       printf("\nKoliko vremena unosite ? ");
                scanf("%u",&n);
        }while(n<=0 || n>MAX );

        for(i=0;i<n;i++)
        {       /*zadaje se godina*/
                do
                {       printf("\nUnesite godinu (2000<=god<=2010) ");
                        scanf("%u",&godina);
                }while (godina<2000 || godina>2010);

                /*zadaje se mesec*/
                do
                {       printf("Unesite mesec ");
                        scanf("%u",&mesec);
                }while (mesec<1 || mesec >12);

                /*zadaje se dan*/
                do
                {       prestupna=0;  ispravno=1;
```

```
                printf("Unesite dan ");
                scanf("%u",&dan);

                if(dan<=0)   continue;

                switch(mesec)
                {       case 1:case 3:case 5:case 7:case 8:case 10:case 12:
                                if (dan>31) ispravno =0;
                                break;
                        case 4:case 6:case 9:case 11: if (dan>30) ispravno =0;
                                break;
                        case 2:
                                if(godina%400==0)
                                        prestupna =1;
                                else if(godina%100!=0 && godina%4==0)
                                        prestupna=1;
                                if(dan>28+prestupna)
                                        ispravno=(unsigned)0;
                                break;
                }
        }while(!ispravno);

        /*zadaje se sat*/
        do
        {       printf("Unesite sat (0-23) ");
                scanf("%u",&sat);
        }while (sat<0 || sat >23);

        /*unosi se minut*/
        do
        {       printf("Unesite minute (0-59) ");
                scanf("%u",&minut);
        }while (minut<0|| minut>59);

        /*unose se sekunde*/
        do
        {       printf("Unesite sekunde (0-59) ");
                scanf("%u",&sekunda);
        }while (sekunda<0 || sekunda>59);

        /*pakovanje vremena*/
        godina-=2000;
        vremena[i]=  (godina<<26)+(mesec<<22)+
                        (dan<<17)+(sat<<12)+(minut<<6)+sekunda;
}
getchar();
printf("\nSredjivanje: opadajuci ili rastuci poredak (o/r): ");
opcija=tolower(getchar());
```

```
	/*rastuci ili opadajuci poredak*/
	do
	{	if(opcija=='o') z=-1;
		else if (opcija=='r') z=1;
		else printf("\nNepoznata opcija\n");
	}while (opcija!='o' && opcija!='r');


	/*sredjivanje vremena*/
	for(i=1;i<n;i++)
		for(j=0;j<i;j++)
			if (z*vremena[i]<z*vremena[j])
			{	pom=vremena[i],vremena[i]=vremena[j],vremena[j]=pom;}

	/*vracanje vremena i stampanje*/
	putchar('\n');
	for(i=0;i<n;i++)
	{	printf("%2u.",(vremena[i]<<10)>>27);
		printf("%2u.",(vremena[i]<<6)>>28);
		printf("%4u. ",(vremena[i]>>26)+2000);

		printf("%2u:",(vremena[i]<<15)>>27);
		printf("%2u:",(vremena[i]<<20)>>26);
		printf("%2u\n",(vremena[i]<<26)>>26);
	}
	putchar('\n');
}
```

Задатак 8.

Написати програм на језику *C* који проверава да ли су правилно написане све заграде у тексту задатом преко тастатуре (да ли свака отворена заграда има одговарајућу затворену) и штампа на екрану поруку о томе.

Решење 8.

```
/*Program za proveru pravilno napisanih svih vrsta zagrada u zadatom tekstu*/
#include<stdio.h>
#include<string.h>
#define VO '{'
#define VZ '}'
#define SO  '['
#define SZ  ']'
#define MO  '('
#define MZ  ')'
#define MAX 100000

main()
{       char tekst [MAX],tekst_c[MAX],znak;
        int i=0,polazak=-1,duzina,pravilno=1,vo,vz,so,sz,mo,mz;
        int levo=-1,desno=-1;

        vo=vz=so=sz=mo=mz=0;

        printf("\nUnesite tekst, za kraj Ctrl+z\n");
        while((znak=getchar())!=EOF)
                tekst[i++]=znak;
        tekst[i]='\0';
        duzina=i;

        /*pravi se kopija, jer se ispravno napisan par zagrada
          zamenjuje prazninom i bio bi izgubljen originalan sadrzaj*/
        strcpy(tekst_c,tekst);
        i=0;

        /*prvo se proverava da li je isti broj otvorenih i zatvorenih zagrada
          istog tipa i odredjuje se pozicija prve zatvorene zagrade*/
        while(i<duzina)
        {       switch(tekst_c[i])

                {       case VO:    vo++;break;
                        case SO:    so++;break;
                        case MO:    mo++;break;
                        case VZ:    if(desno==-1)
                                            desno=i;
                                    vz++;
                                    break;
```

```
                case SZ:    if(desno==-1)
                                desno=i;
                            sz++;
                            break;
                case MZ:    if(desno==-1)
                                desno=i;
                            mz++;
                            break;
                default:    break;
            }
            i++;
        }

        if(vo!=vz || so!=sz || mo!=mz)
            pravilno=0;
        else                /*ako je jednak broj otvorenih i zatvorenih zagrada
                             proverava se da li su odgovarajuce*/
        {   levo=desno-1;
            while(pravilno && levo>=0 && desno<duzina)
            {   /*ako se nadje otvorena zagrada, proverava se da li
                 odgovara zatvorenoj*/

                if(tekst_c[levo]==VO || tekst_c[levo]==SO || tekst_c[levo]==MO)
                    if(tekst_c[levo]==tekst_c[desno]-2 ||
                       tekst_c[levo]==tekst_c[desno]-1)
                    {   tekst_c[levo]=tekst_c[desno]=' ';

                        /*ako jeste brisanje zagrada*/
                        while(tekst_c[desno]!=VZ && tekst_c[desno]!=SZ
                            && tekst_c[desno]!=MZ && desno<duzina)
                            desno++ ;
                        levo=desno-1;
                    }
                    else
                        pravilno=0;    /*ako nije odgovarajuca zatvorena*/
                else levo--;    /*pomeraj u levo da se nadje otvorena zagrada*/
            }
        }

        if(pravilno)
            printf("\nSve zagrade su pravilno napisane\n");
        else
            printf("\nSve zagrade nisu pravilno napisane\n");
}
```

Задатак 9.

Написати програм на језику *C* који чита текст са тастатуре и штампа га на екрану у колонама. Програм прихвата са тастатуре број колона, који треба да буде у опсегу вредности од 1 до 4. Једна реч текста не треба да буде подељена у више редова, а максималан број редова по страни је 10. Рад овог програма тестирати са што већим бројем речи, да би била реализована подела у колоне.

Решење 9.

```
/*Program za  formatiranje zadatog teksta po kolonama*/
#include<stdio.h>
#include<string.h>
#define PRAZNO 3          /*razmak izmedju kolona*/
#define EKRAN 80          /*sirina ekranskog reda*/
#define MAXKOL 4          /*maksimalan broj kolona*/
#define REDOVI 10         /*broj redova po strani*/
#define TEKST 100000      /*maksimalan broj znakova u tekstu*/

main()
{       char znak, prethodni;
        char tekst[TEKST], red[TEKST/10][EKRAN+1], prazno[PRAZNO+1];
        int i=0,kolona,sirina,duzina_teksta,n=0,pocetak_reda,kraj_reda,p,j;
        int blok,pocetak_bloka,kraj_bloka, poslednja, maxn=0;

        for(i=0;i<PRAZNO;i++)
                prazno[i]=' ';
        prazno[i]='\0';

        printf("\nUnesite tekst, za kraj Ctrl+z\n");
        p=0;
        while((znak=getchar())!=EOF)
                tekst[p++]=znak;
        tekst[p]='\0';
        duzina_teksta=p;

        do
        {       printf("\nKoliko kolona (1<=n<=%i) ",MAXKOL);
                scanf("%i",&kolona);
        }while (kolona<1 || kolona>=MAXKOL);

        sirina=(EKRAN-(kolona-1)*PRAZNO)/kolona; /*sirina: karaktera po koloni*/
        p=0;

        /*preciscavanje teksta: uklanjaju se znak za tabulator, znak za prelaz u novi
        red i svaka visestruka praznina zamenjuje se jednom a na kraj se dodaje
        jedna praznina*/
        prethodni=' ';
```

```
for(i=0;i<duzina_teksta;i++)
{       if(tekst[i]=='\n' || tekst[i]=='\t')
        {       tekst[i--]=' ';
                continue;
        }
        if(!(tekst[i]==' ' && prethodni==' '))
                tekst[p++]=tekst[i];
        prethodni=tekst[i];
}
 tekst[p++]=' ';tekst[p]='\0';

/*provera da li je duzina reci veca od sirine kolone*/
for(i=0;i<p;i++)
        if (tekst[i]!=' ')
                n++;
        else
        {       if (n>maxn)
                        maxn=n;
                n=0;
        }
/*ako je duzina najduze reci veca od sirine kolone */
/*smanjuje se broj kolona na najveci moguci broj*/
if(maxn>sirina)
{       printf("\nPostoji rec cija je duzina veca od sirine kolone\n");
        while(maxn>sirina)
        {       kolona--;
                sirina=(EKRAN-(kolona-1)*PRAZNO)/kolona;
        }
        printf("\nNajveci moguci broj kolona je: %i\n\n", kolona);
}

/*odvajaju se redovi velicine jedne kolone*/
poslednja=0;n=0;
pocetak_reda=0;

while(1)
{       kraj_reda=pocetak_reda+sirina-1;
        if(kraj_reda>=duzina_teksta)
        {       kraj_reda=duzina_teksta-1;
                poslednja=1;
        }
        /*ako je registrovan kraj teksta, poslednji red nece imati punu duzinu
        praznina se dodaje na kraj ako se kraj reda poklopi sa krajem stringa
        da se tekst ne bi prekidao u pola reci*/
        while(tekst[kraj_reda]!=' ')
        kraj_reda--;
        p=0;
        for(i=pocetak_reda;i<kraj_reda;i++)
                red[n][p++]=tekst[i];
```

```
                red[n][p]='\0';
                if(poslednja)
                        break;
                pocetak_reda=kraj_reda+1;n++;
        }
        n++;

         /*sada se svaki red dopunjuje do pune sirine kolone*/
        for(i=0;i<n;i++)
        {       for(j=strlen(red[i]);j<sirina;j++)
                        red[i][j]=' ';
                red[i][j]='\0';
        }
        blok=REDOVI*kolona; /*blok je broj stringova po jednoj strani*/

        /*stampanje formatiranog teksta na ekranu*/
        getchar();
        pocetak_bloka=0;
        poslednja=0;
        while(1)
        {       printf("\n\n");
                kraj_bloka=pocetak_bloka+blok-1;
                if(kraj_bloka>=n)
                        poslednja=1;

                for(i=0;i<REDOVI;i++)
                        for(j=0;j<kolona;j++)
                        {       p=j*REDOVI+i+pocetak_bloka;
                                if(j==0 && p>=n)
                                        break;
                                else if(p>=n)
                                {       if(j==kolona-1)
                                                printf("\n");
                                        continue;
                                }
                                printf("%s",red[p]);
                                if(j!=kolona-1)
                                        printf("%s",prazno);
                                else if(kolona!=1)
                                        printf("\n");
                                                        }
                                if((j==0 && p>=n) || poslednja)
                                        break;
                                printf("\nPritisni bilo koji taster za nastavak  ");
                                pocetak_bloka=kraj_bloka+1;
                                getchar();
        }
        printf("\n\n");
}
```

Задатак 10.

Написати програм на језику *C* који уређује бројеве од 1 до 15, случајно распоређене у пољима квадрата димензије 4x4 на следећи начин: помера их за по једно поље горе, доле, лево или десно, уз коришћење једног празног поља и тако поставља све постојеће бројеве у растући поредак по врстама, почев од горњег левог угла. За сваки померај програм приказује тренутан распоред бројева у квадрату, онда од корисника тражи да изабере број и када овај број добије са тастатуре, приказује на екрану резултат изведеног помераја.

Решење 10.

```
/*Program za uredjivanje slucajno rasporedjenih brojeva u kvadratu 4x4*/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 16
#define GRANICA 4

main()
{       int i,j,broj,p,imax,jmax,slozeno,identicno;
        int mat[GRANICA+2][GRANICA+2],pom[MAX];
        time_t timer;

        printf("\nKoristeci prazno mesto, svaki broj postaviti na svoje mesto.");
        printf("\nMoguce je samo horizontalno i vertikalno pomeranje broja.\n");

        /*pokretanje generatora slucajnih brojeva pomocu tekuceg vremena*/
        srand(time(&timer));

        /*formiranje matrice 4x4*/
        for(i=0;i<MAX;i++)
        {       broj=rand();
                if( broj>MAX || broj==0 )
                {       i--;continue;           }
        for(j=0;j<i;j++)
                if(broj==pom[j])
                {       i--;break;      }
        if(j==i)
                pom[i]=broj;
        else
                i--;
        }
        p=0;

        for(i=1;i<GRANICA+1;i++)
                for(j=1;j<GRANICA+1;j++)
                        mat[i][j]=pom[p++];
```

```
	/*prosirenje matrice nulama  sa strane*/
	for(i=0;i<6;i++)
	mat[0][i]=mat[i][0]=mat[GRANICA+1][i]=mat[i][GRANICA+1]=0;
	slozeno=0;

	while(1)
	{	/*stampanje matrice*/
		putchar('\n');
		for(i=1;i<GRANICA+1;i++)
		{	for(j=1;j<GRANICA+1;j++)
			{	if (mat[i][j]==MAX)
				{	imax=i,jmax=j; printf("    "); }
				else
					printf("%4d", mat[i][j]);
			}
			printf("\n");}

		/*provera da li je matrica slozena*/
		identicno =1;
		for(i=1;i<GRANICA+1;i++)
		{	for(j=1;j<GRANICA+1;j++)
				if(mat[i][j]!=(i-1)*GRANICA+j)
				{	identicno=0;break;   }
				if(!identicno)
					break;
		}
		if (identicno)
			slozeno=1;
		if (slozeno)
			break;

		printf("\nkoji broj se pomera ");
		scanf("%i",&broj);

		if(broj<1 || broj>MAX-1)
			broj=MAX;

		/*trazenje tog broja u okolini MAX */
		if (mat[imax][jmax-1]==broj)
			mat[imax][jmax-1]=MAX,mat[imax][jmax]=broj;
		else if(mat[imax][jmax+1]==broj)
			mat[imax][jmax+1]=MAX,mat[imax][jmax]=broj;
		else if (mat[imax-1][jmax]==broj)
			mat[imax-1][jmax]=MAX,mat[imax][jmax]=broj;
		else if(mat[imax+1][jmax]==broj)
			mat[imax+1][jmax]=MAX,mat[imax][jmax]=broj;
	}
	printf("\nBRAVO\n");
}
```

Задатак 11.

Написати програм на језику *C* за игру Витезови округлог стола. За округлим столом седи *n* (*n<=50*) витезова. Преко тастатуре се задају њихова имена, сва међусобно различита. Почевши од једног изабраног витеза, сваки *k*-ти се удаљава од стола. Име витеза од кога почиње бројање, као и број *k* задају се преко тастатуре. Знак задатог броја *k* одређује смер кретања при бројању (позитивна вредност значи смер у десно, а негативна у лево). Програм треба да прикаже на екрану поруку о томе којим се редом витезови удаљавају од стола, као и то који витез остаје последњи за столом.

Решење 11.

```
/*Program za igru Vitezovi okruglog stola*/
#include<stdio.h>
#include<string.h>
#define MAX 50              /*dozvoljen broj vitezova*/
#define MAXS 30             /*najveci broj znakova u imenu*/

main()
{       int n,i,indeks,k,rbr,izbacujemo,j;
        char vitez[MAX][MAXS+1],pv[MAXS+1];

        /*unos broja vitezova*/
        do
        {       printf("\nKoliko ima vitezova  ");
                scanf("%d",&n);
        }while (n<0 || n>MAX);

        /*Unos imena vitezova i provera da li ima istih*/
        getchar();
        printf("\nUnesite imena vitezova:\n");

        for(i=0;i<n;i++)
        {       gets(vitez[i]);
                for(j=0;j<i;j++)
                        if(strcmp(vitez[i],vitez[j])==0)
                        {       printf("\nIme je vec uneto\n");
                                i--;
                                break;
                        }
        }

        /*Pronalazenje imena viteza koji je proglasen pocetnim*/
        rbr=-1;
        while(1)
        {       printf("\nUnesite ime viteza od koga se krece\n");
                gets(pv);
```

```
		for(i=0;i<n;i++)
			if(strcmp(vitez[i],pv)==0)
			{	rbr=i;
				break;
			}
		if(rbr<0)
			printf("\nTaj vitez ne postoji\n");
		else
			break;
	}

	/*Ako je k negativno smer je u levo a ako je pozitivno smer je u desno*/
	while(1)
	{	printf("\nKoji po redu vitez odlazi? \n");
		printf("(minus je za smer u levo, plus za smer u desno) ");
		scanf("%i",&k);
		if(k!=0)
			break;
		printf("\nBroj mora biti razlicit od nule.\n");
	}

	/*Ako je k<0 invertujemo niz*/
	if(k<0)
	{	for(i=0;i<n/2;i++)
		{	strcpy(pv,vitez[i]);
			strcpy(vitez[i],vitez[n-1-i]);
			strcpy(vitez[n-1-i],pv);
		}
		indeks=n-1-rbr;
		k=-k;
	}
	else
		indeks=rbr;

	/*Uklanjanje vitezova*/
	printf("\nVitezovi odlaze sledecim redom:\n");

	while(n!=1)
	{	izbacujemo =(indeks+k-1)%n;
		printf("%s\n",vitez[izbacujemo]);
		for(i=izbacujemo;i<n-1;i++)
			strcpy(vitez[i],vitez[i+1]);
		n--;
		indeks=izbacujemo;
	}
	printf("\n\nPoslednji ostaje: %s\n\n",vitez[0]);
}
```

Задатак 12.

Написати програм на језику *C* за игру *Master Mind*. Програм случајно генерише 4 броја, сваки у опсегу од 1 до 6 уз могућност понављања бројева. Играч покушава из највише 10 пута да погоди која је то комбинација, тако што задаје своје комбинације бројева преко тастатуре. За сваки покушај играч добија одговор колико бројева је погодио и колико од погођених бројева је на свом месту. Број покушаја играча ограничен је на 10.

Решење 12.

```
/*Program za igru Master Mind*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include <time.h>
#define POKUSAJ 10
#define MAX 4
#define BROJ 6

main()
{       int i,j,na_mestu,van_mesta, broj_pokusaja,u_opsegu;
        int zadata[MAX],czadata[MAX],pogadjam[MAX];
        char c;
        time_t timer;

        /*pokretanje generatora slucajnih brojeva pomocu tekuceg vremena*/
        srand(time(&timer));
        printf("\nOvo je igra pogadjanja zadate kombinacije od 4 broja");
        printf("\nU igri su brojevi od 1 do %i i mogu se ponavljati\n",BROJ);

        do
        {       /*pravi se slucajna kombinacija*/
                for(i=0;i<MAX;i++)
                 {      zadata[i]=rand();
                        if(zadata[i]<1 || zadata[i]>BROJ)
                                i--;
                 }

                /*pocinje pogadjanje*/
                printf("\n\t\t\t\t\tBroj pogodaka\n");
                printf("\n\t\t\t\tna mestu\tnije na mestu");
                broj_pokusaja=0;
                while(broj_pokusaja<POKUSAJ)
                {       broj_pokusaja++;

                        /*unosi se kombinacija igraca*/
                        u_opsegu=0;
                        printf("\nVas %i pokusaj: ",broj_pokusaja);
```

```
                    for(i=0;i<MAX;i++)
                    {       scanf("%i",&pogadjam[i]);
                            if(pogadjam[i]<=0 || pogadjam[i]>BROJ)
                                    u_opsegu=1;
                    }
                    na_mestu=0;   van_mesta=0;

                    /*pravi se kopija zadate kombinacije */
                    for(i=0;i<MAX;i++)
                            czadata[i]=zadata[i];

                    /*trazenje broja koji je na svom mestu*/
                    for(i=0;i<MAX;i++)
                            if(zadata[i]==pogadjam[i])
                            {       na_mestu++;
                                    pogadjam[i]=0;czadata[i]=0;
                            }
                    if(na_mestu==MAX)
                    {       printf("\nBravo, pogodili ste zadatu kombinaciju ");
                            printf("iz %i. puta\n\n",broj_pokusaja);
                            break;
                    }

                    /*trazenje broja koji je van svog mesta*/
                    for(i=0;i<MAX;i++)
                            for(j=0;j<MAX;j++)
                                    if(czadata[i]!=0)
                                            if(czadata[i]==pogadjam[j])
                                            {       pogadjam[j]=czadata[i]=0;
                                                    van_mesta++;
                                            }
                    printf("%37i%18i\n",na_mestu,van_mesta);
                    if(u_opsegu)
                      printf("\nNAPOMENA: Birati brojeve od 1 do %i\n",BROJ);
            }

            if(na_mestu!=MAX)
            {       printf("\nZao mi je niste uspeli da pogodite kombinaciju ");

                    for(i=0;i<MAX;i++)
                            printf("%d ",zadata[i]);printf("\n");
            }

            getchar();
            puts("Da li zelite jos da igrate (d/n) ?");
            c=getch();
        } while (tolower(c)=='d');
}
```

Евиденција урађених лабораторијских вежби

Вежба	Датум	Вежбу прегледао
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		

Име и презиме студента: _____

Број индекса и година уписа: _____

Студијски програм: _____

Прилог

Табела 1. Службене речи језика *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

Табела 2. Контролне секвенце стандардних излазних функција језика *C*

Секвенца	Значење
\n	Прелаз у нови ред (NewLine NL)
\t	Хоризонтални табулар (HorizontalTabulation HT)
\v	Вертикални табулар (VerticalTabulation VT)
\b	Повратак за једно место (BackSpace BS)
\r	Повратак на почетак реда (CarriageReturn CR)
\f	Прелаз на следећу страну (FormFeed FF)
\a	Аларм/Звоно (Alarm/Bell BELL)
\\	Коса црта уназад (BackSlash)
\'	Апостроф
\"	Наводник

Табела 3. Основни типови података у језику *C*

Тип	Ознака	Величина (бајтова)	Опсег
Character	char	1	-128 do 127
Short integer	short	2	-32768 do 32767(1)
Integer	int	2/4	(1)/(2)
Long integer	long	4	-2,147,483,648 до 2,147,438,647(2)
Unsigned character	unsigned char	1	0 do 255
Unsigned short integer	unsigned short	2	0 do 65535 (3)
Unsigned integer	unsigned	2/4	(3)/(4)
Unsigned long integer	unsigned long	4	0 до 4,294,967,295 (4)
Single-precision Floating_point	float	4	1.2E-38 до 3.4E38 (прецизност=7 цифара)
Double-precision Floating-point	double	8	2.2E-308 до 1.8E308 (прецизност=16 цифара)

Табела 4. Контроле за форматирање излаза у језику *C*

Ознака	Тип податка за који се користи
%c	char
%d %u %o %x (X)	int (децимални облик) unsigned int (децимални облик) int/unsigned int (октални облик) int/unsigned int (хексадецимални облик)
%f %e %g	float, double (облик са децималном тачком) float, double (облик са експонентом) float, double (најједноставнији облик)

Табела 5. Контроле за форматирање улаза у језику *C*

Ознака	Тип податка за који се користи
%c	char
%d %u %o %x (X) %i	int (децимални облик) unsigned int (децимални облик) int/unsigned int (октални облик) int/unsigned int (хексадецимални облик) int (било који од три горе поменута облика)
%f %e %g	float (облик са децималном тачком или експонентом) float (облик са децималном тачком или експонентом) float (облик са децималном тачком или експонентом)
%lf %le %lg	double (облик са децималном тачком или експонентом) double (облик са децималном тачком или експонентом) double (облик са децималном тачком или експонентом)

Табела 6. Приоритети оператора у језику *C*

Приоритет	Број операнада	Оператор
15	2	[ ] ( ) . ->
14	1	! ~ ++ -- + - * & (тип) sizeof
13	2	* / %
12	2	+ -
11	2	<< >>
10	2	< <= > >=
9	2	== !=
8	2	&
7	2	^
6	2	|
5	2	&&
4	2	||
3	3	?:
2	2	= += -= *= /= %= &= ^= |= <<= >>=
1	2	,

Табела 7. Стандардна библиотека *ctype.h*

Свака даље поменута функција библиотеке *ctype.h* враћа вредност типа *int* различиту од 0 (*true*), ако аргумент *c* задовољава описани услов (аргумент *c* је типа је *int* и чува *ASCII* кôд знака који се испитује).

isalnum(c)	Вредност функције је различита од 0 ако је вредност *c* ASCII кôд слова или децималног броја.
isalpha(c)	Вредност функције је различита од 0 ако је вредност *c* ASCII кôд слова.
isdigit(c)	Вредност функције је различита од 0 ако је вредност *c* ASCII кôд децималне цифре.
islower(c)	Вредност функције је различита од 0 ако је вредност *c* ASCII кôд малог слова.
isupper(c)	Вредност функције је различита од 0 ако је вредност *c* ASCII кôд великог слова.
tolower(c)	Ако је вредност *c* ASCII кôд великог слова, вредност функције је ASCII кôд истог малог слова.
toupper(c)	Ако је вредност *c* ASCII кôд малог слова, вредност функције је ASCII кôд истог великог слова.

Табела 8. Стандардна библиотека *math.h*

У свим даље поменутим функцијама библиотеке *math.h* вредности *x* и *y* су типа *double* и све функције враћају вредност типа *double.*

cos(x)	Вредност функције је *cos x*
fabs(x)	Вредност функције је |x|
log(x)	Вредност функције је $\log_e$ x, x>0.
log10(x)	Вредност функције је $\log_{10}$ x, x>0.
pow(x, y)	Вредност функције је x^y
sin(x)	Вредност функције је sin x.
sqrt(x)	Вредност функције је $\sqrt{x}$
tan(x)	Вредност функције је tg x.

Табела 9. Стандардна библиотека *stdlib.h*

Даље су поменуте само неке коришћене функције библиотеке *stdlib.h*.

atof(s)	Функција враћа вредност типа *double*, добијену конверзијом цифара из низа знакова *s.*
atoi(s)	Функција враћа вредност типа *int*, добијену конверзијом цифара из низа знакова *s.*
atoll(s)	Функција враћа вредност типа *long*, добијену конверзијом цифара из низа знакова *s.*
rand()	Функција враћа псеудо-случајну вредност типа *int*, у опсегу бројева од 0 до *RAND_MAX* (32767).

Табела 10. Стандардна библиотека *string.h*

У свим даље поменутим функцијама библиотеке *string.h*, *s, s1* и *s2* су низови карактера, а *n* је вредност типа *int*.

strcat(s1, s2)	Функција дописује садржај низа *s2* на крај постојећег садржаја низа *s1* и враћа показивач на резултујући низ *s1.*
strcmp(s1, s2)	Функција пореди садржаје низова *s1* и *s2* и враћа вредност типа *int:* <0, ако је низ *s1* мањи по *ASCII* табели од низа *s2,* =0, ако је низ *s1* исти по *ASCII* табели као и низ *s2,* >0, ако је низ *s1* већи по *ASCII* табели од низа *s2.*
strcpy(s1, s2)	Функција копира садржај низа *s2* од почетка низа *s1* (укључује и \0) и враћа показивач на резултујући низ *s1.*
strlen(s)	Функција враћа дужину низа *s* (не укључује \0).
strncat(s1, s2, n)	Функција дописује садржај првих *n* карактера низа *s2* на крај постојећег садржаја низа *s1* и враћа показивач на резултујући низ *s1.*
strncmp(s1, s2, n)	Функција пореди садржаје првих *n* карактера низова *s1* и *s2* и враћа вредност типа *int:* <0, ако је првих *n* карактера низа *s1* мање по *ASCII* табели од првих *n* карактера низа *s2,* =0, ако је првих *n* карактера низа *s1* исто по *ASCII* табели као и првих *n* карактера низа *s2,* >0, ако је првих *n* карактера низа *s1* веће по *ASCII* табели од првих *n* карактера низа *s2.*
strncpy(s1, s2, n)	Функција копира садржај првих *n* карактера низа *s2* од почетка низа *s1* (укључује и \0) и враћа показивач на резултујући низ *s1.*

Табела 11. Табела *ASCII* кодова

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Литература

[1] Ласло Краус, Програмски језик *C* са решеним задацима, Академска мисао, Београд 2004.

[2] Слободан Обрадовић, Вештина доброг програмирања, Виша електротехничка школа, Београд 1997.

[3] Милан Чабаркапа, *C* основи програмирања, Круг, Београд 1996.

[4] *Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language - ANSI C, Prentice Hall Software Series*, 1988.

[5] *Augie Hansen,* Програмирање на језику *C* – потпуни водич за програмски језик *C*, Микро књига, Београд 1991.

[6] *Peter Aitken, Bradley Jones,Teach Yourself C in 21 Days,CD edition*,1998.

[7] *Chris H. Pappas, William H. Murray*, *C/C++* Водич за програмере, Микро књига, Београд 1996.