

**Петар Спалевић
Бранимир Јакшић
Стефан Панић**

ЗБИРКА РЕШЕНИХ ЗАДАТАКА ИЗ ПРОГРАМСКОГ ЈЕЗИКА С

II ДЕО

Косовска Митровица, 2011.

Збирка решених задатака из програмског језика **C**, II део

Прво издање

Аутори: Петар Спалевић, ванредни професор Факултета техничких наука у Косовској Митровици
Бранимир Јакшић, асистент на Факултету техничких наука у Косовској Митровици
Стефан Панић, доцент Природно-математичког факултета у Косовској Митровици

Рецензенти: Градимир Миловановић, редовни професор Мегатренд универзитета и
члан Српске академије наука и уметности
Братислав Мирић, редовни професор Државног универзитета у Новом Пазару

Технички уредници: Аутори

Издавач:

Штампа:

Тираж: 100 примерака

НАПОМЕНА: Фотокопирање или умножавање на било који начин или поновно објављивање ове књиге – у целини или деловима – није дозвољено без претходне сагласности и писменог одобрења издавача.

Предговор

Ова збирка задатака је помоћни уџбеник за учење програмирања на језику C. Задаци прате градиво које одговара предмету Програмирање 2 којег слушају студенти у оквиру студијског програма Електротехничко и рачунарско инжењерство на Факултету техничких наука у Косовској Митровици. Збирку могу користити и студенти других факултета који у оквиру својих предмета изучавају језик C, као и ученици средњих школа.

Збирка је тако конципирана да је могу користити и почетници у програмирању. Задаци су у свакој области изложени по тежини, од најлакших ка тежим. Кроз задатке, поред елемената самог језика, приказане су најчешће коришћени поступци у програмирању: претраживање и уређивање низова, обрада знаковних података, рад са показивачима и структурама, као и рад са датотекама.

Решења свих задатака су потпуна у смислу да се дати програмски кодови могу ивршавати на рачунари. Сви задаци су урађени, проверени и тестирани коришћењем програма DEV C/C++. Поред самих програмских кодова дати су изгледи на екрану након тестирања програма. За сложене задатке дата су објашњења у виду текста или пропратног коментара у програмском коду.

Збирка је подељена у два дела. Први део обухвата основне елементе и конструкције језика C: просте линијске структуре, грањање у програму, петље, скокови, улаз-излаз, функције, низови и матрице.

Други део је наставак градива из претходног дела, који обухвата и нешто сложеније задатке: стрингови, показивачи (обухватајући и рад са показивачима кроз све елементе језика из првог дела збирке), динамичка зона меморије, структуре и рад са датотекама.

*У Косовској Митровици,
септембар, 2011.*

Аутори

САДРЖАЈ

| | | |
|----------|---|------------|
| 1 | СТРИНГОВИ | 1 |
| 1.1 | Основне конструкције програма са стринговима | 1 |
| 1.2 | Основне операције са стринговима | 6 |
| 1.3 | Стрингови и функције | 12 |
| 1.4 | Претраживање стрингова | 17 |
| 1.5 | Уређивање стрингова | 19 |
| 1.6 | Сортирање стрингова | 24 |
| 2 | ПОКАЗИВАЧИ | 29 |
| 2.1 | Основне конструкције програма са показивачима | 20 |
| 2.2 | Показивачи као аргументи функција | 33 |
| 2.3 | Показивачи и низови | 38 |
| 2.4 | Показивачи на низове као аргументи функције | 42 |
| 2.5 | Показивачи и низови | 53 |
| 2.6 | Показивачи на функције | 61 |
| 2.7 | Полиморфне функције | 66 |
| 3 | ДИНАМИЧКА ЗОНА МЕМОРИЈЕ | 70 |
| 3.1 | Основне конструкције програма са динамичком зоном меморије | 70 |
| 3.2 | Низови и динамичка зона меморије (<code>malloc()</code>) | 72 |
| 3.3 | Низови и динамичка зона меморије (<code>calloc()</code> и <code>realloc()</code>) | 78 |
| 3.4 | Низ показивача и динамичка алокација меморије | 83 |
| 3.5 | Матрице и динамичка зона меморије | 85 |
| 4 | СТРУКТУРЕ | 99 |
| 4.1 | Увод у структуре | 99 |
| 4.2 | Структуре и показивачи | 107 |
| 4.3 | Низови структура | 116 |
| 4.4 | Сортирање низова структура | 129 |
| 4.5 | Структуре и динамичка зона меморије | 137 |
| 4.6 | Уније | 143 |
| 5 | ДАТОТЕКЕ | 146 |
| 5.1 | Основне операције са датотекама | 146 |
| 5.2 | Датотеке са низовима и матрицама | 159 |
| 5.3 | Датотеке са стринговима | 167 |
| 5.4 | Датотеке са структурама | 177 |
| | ЛИТЕРАТУРА | 193 |

1 СТРИНГОВИ

Табела 1.1: Функције за читање и писање знакова без конверзије дефинисане у библиотеци `<stdio.h>`

| функција | значење |
|-------------------------|---|
| <code>getchar()</code> | Функција чита следећи знак, укључујући и беле знакове са главног улаза рачунара (тастатуре). Вредност функције је код прочитаног знака или симболичка константа <code>EOF</code> (<i>end of file</i>) уколико је прочитани сигнал за крај датотеке као и у случају грешке у току читања. Резултат је типа <code>int</code> . Сигнал за крај датотеке преко тастатуре под оперативним системом <i>MS-DOS</i> се задаје управљачким знаком <code>ctrl+Z</code> . |
| <code>putchar(c)</code> | Функција исписује знак <code>c</code> на главном излазу рачунара (екрану). Тип аргумента је <code>int</code> . Вредност функције је код исписаног знака или симболичка константа <code>EOF</code> у случају грешке. Резултат је типа <code>int</code> . |
| <code>gets(s)</code> | Функција чита ред текста (до знака за прелазак у нови ред <code>\n</code>) са главног улаза рачунара (тастатуре) и смешта га, као бочни ефекат у ниску <code>s</code> (типа <code>char[]</code> , односно <code>char*</code>). Уместо знака <code>\n</code> у низ <code>s</code> се поставља знак <code>\0</code> . Вредност функције у случају наилска на крај датотеке, или у случају откривања грешке у току читања, једнака је симболичкој константи <code>NULL</code> , односно почетној адреси низа <code>s</code> у случају успешног читања. |
| <code>puts(s)</code> | Функција исписује ред ниске <code>s</code> (типа <code>char[]</code> , односно <code>char*</code>) до завршног знака <code>\0</code> као ред текста на главном излазу рачунара (екрану), додајући знак за прелазак у нови ред (<code>\n</code>) иза последњег знака. Ако текст у низу <code>s</code> садржи и знакове <code>\n</code> , резултат ће бити више исписаних редова. Вредност функције (типа <code>int</code>) је не-негативни број, односно симболичка константа <code>EOF</code> у случају откривања грешке у току испитивања. |

Табела 1.2: Функције за испитивање знакова дефинисане у библиотеци `<ctype.h>`

| функција | значење |
|--------------------------|--|
| <code>isalnum(c)</code> | Да ли је <code>c</code> слово или цифра? |
| <code>isalpha(c)</code> | Да ли је <code>c</code> слово? |
| <code>islower(c)</code> | Да ли је <code>c</code> мало слово? |
| <code>isupper(c)</code> | Да ли је <code>c</code> велико слово? |
| <code>isdigit(c)</code> | Да ли је <code>c</code> децимална цифра? |
| <code>isxdigit(c)</code> | Да ли је <code>c</code> хексадецимална цифра? |
| <code>isspace(c)</code> | Да ли је <code>c</code> бели знак? |
| <code>isgraph(c)</code> | Да ли је <code>c</code> штампајући знак, али није размак? |
| <code>isprint(c)</code> | Да ли је <code>c</code> штампајући знак (укључујући и размак)? |

| | |
|-------------------------|---|
| <code>ispunct(c)</code> | Да ли је <code>c</code> специјални знак (штампајући али није слово ни цифра)? |
| <code>iscntrl(c)</code> | Да ли је <code>c</code> управљачки знак? |
| <code>tolower(c)</code> | Ако је <code>c</code> велико слово, вредност функције је код одговарајућег малог слова, а иначе вредност функције је <code>c</code> . |
| <code>toupper(c)</code> | Ако је <code>c</code> мало слово, вредност функције је код одговарајућег великог слова, а иначе вредност функције је <code>c</code> . |

* Вредност свих функција је типа *logical*, а тип аргумента `c` је **int** (вредност треба да је код неког знака). За логичку неистину дају неку ненулту вредност, а не обавезно вредност 1, као што дају релацијски и логички оператори.

Табела 1.3: Функције за обраду стрингова дефинисане у библиотеци `<string.h>`

| функција | значење |
|-------------------------------|--|
| <code>strcpy(t, s)</code> | Преписује ниску <code>s</code> у ниску <code>t</code> , укључујући и завршни знак <code>\0</code> . |
| <code>strncpy(t, s, n)</code> | Преписује највише <code>n</code> знакова из ниске <code>s</code> у ниску <code>t</code> . Ако ниска <code>s</code> нема <code>n</code> знакова, резултат се допуњује знаковима <code>\0</code> до дужине <code>n</code> . Ако ниска <code>s</code> садржи бар <code>n</code> знакова, не ставља се <code>\0</code> на крај ниске <code>t</code> . |
| <code>strcat(t, s)</code> | Дописује ниску <code>s</code> на крај ниске <code>t</code> . |
| <code>strncat(t, s, n)</code> | Дописује највише <code>n</code> знакова из ниске <code>s</code> на крај ниске <code>t</code> . |
| <code>strcmp(u, s)</code> | Функција упоређује ниске <code>u</code> и <code>s</code> . Вредност функције је негативна ако је текст <code>u</code> испред текста <code>s</code> по лексикографском поретку, већа од нуле ако је текст <code>u</code> иза текста <code>s</code> , и једнака нули ако су та два текста једнака. Упоређивање се врши на основу вредности кодова појединих знакова на датом рачунару. Тип резултата је int . |
| <code>strncmp(u, s, n)</code> | Функције упоређује највише првих <code>n</code> знакова у нискама <code>u</code> и <code>s</code> . Резултат се образује на исти начин као и код функције <code>strcmp</code> . Тип резултата је int . |
| <code>strlen(s)</code> | Вредност функције је број знакова (тип int) у знаковном низу <code>s</code> . Завршни знак <code>\0</code> се не убраја у резултат. |
| <code>strchr(u, c)</code> | Вредност функције је показивач (тип char*) на први елемент ниске <code>u</code> који садржи знак <code>c</code> . Вредност је <code>NULL</code> ако знак није пронађен. |
| <code>strrchr(u, c)</code> | Вредност функције је показивач (тип char*) на последњи елемент ниске <code>u</code> који садржи знак <code>c</code> . Вредност је <code>NULL</code> ако знак није пронађен. |
| <code>strstr(u, s)</code> | Вредност функције је показивач (тип char*) на први елемент ниске <code>u</code> почев од којег се ниска <code>s</code> појављује као подниска. Вредност је <code>NULL</code> ако подниска није пронађена. |
| <code>strcspn(u, s)</code> | Вредност функције је индекс (тип int) првог елемента ниске <code>u</code> који садржи било који знак из ниске <code>s</code> . Вредност је <code>strlen(u)</code> ако се ниједан знак из <code>u</code> не налази у <code>s</code> . |
| <code>strspn(u, s)</code> | Вредност функције је индекс (тип int) првог елемента ниске <code>u</code> који садржи било који знак који се не појављује у ниски <code>s</code> . Вредност је <code>strlen(u)</code> ако се сваки знак из <code>u</code> налази у <code>s</code> . |

* Аргументи `t`, `u` и `s` су типа *string*, аргумент `n` типа **int**, а аргумент `c` типа **int** (треба да садржи код неког знака). Ниједна од функција не мења садржај аргумента `s` и `u`, а све мењају садржај аргумента `t`. Потребно је водити рачуна о томе да у аргументу `t` (који је у ствари, типа **char []**) има довољно места за прихватање резултујуће ниске. Вредност свих функција које имају аргумент `t` је показивач на тај (одредишни) низ.

Табела 1.4: Функције за конверзију бројева из знаковног облика (низа цифара) у нумеричке типове и обратно дефинисане у библиотеци `<stdlib.h>`

| функција | значење |
|----------------------|--|
| <code>atof(s)</code> | Функција конвертује ASCII низ карактера (цифара) у double . |
| <code>atoi(s)</code> | Функција конвертује ASCII низ карактера (цифара) у int . |
| <code>atol(s)</code> | Функција конвертује ASCII низ карактера (цифара) у long int . |
| <code>itoa(n)</code> | Функција конвертује int у низ карактера (цифара). |

* Аргумент `s` свих наведених функција је типа *string* и треба да садржи низ цифара броја који се конвертује. Аргумент `n` је цео број.

1.1 Основне конструкције програма са стринговима

1.1. Шта се исписује на екрану извршавањем следећег програмског кода:

```
#include <stdio.h>

main()
{
    /*Poslednji bajt niske karaktera se postavlja na '\0' tj. 0*/
    char s[] = {'a', 'b', 'c', '\0'};

    /*Kraci nacin da se postigne prethodno*/
    char t[] = "abc";

    /*Ispis niske s karakter po karakter*/
    int i;
    for(i=0; s[i]!='\0'; i++)
        putchar(s[i]);
    putchar('\n');

    /*Ispis niske s koristeći funkciju printf*/
    printf("%s\n", s);

    /*Ispis niske t karakter po karakter*/
    for(i=0; t[i]!='\0'; i++)
        putchar(t[i]);
    putchar('\n');

    /*Ispis niske t koristeći funkciju printf*/
    printf("%s\n", t);
    getch();
    return 0;
}
```



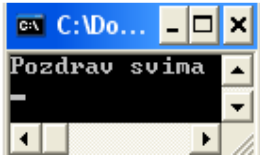
Испис на екрану

1.2. Шта се испишује на екрану извршавањем следећих програмских кодова:

а)

```
#include <stdio.h>

main()
{
    char poruka[] = "Pozdrav svima";
    printf("%s\n", poruka);
    getche();
    return 0;
}
```

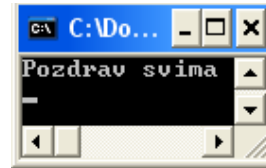


Испис на екрану а)

б)

```
#include <stdio.h>

main()
{
    char poruka[] = "Pozdrav svima";
    puts(poruka);
    getche();
    return 0;
}
```



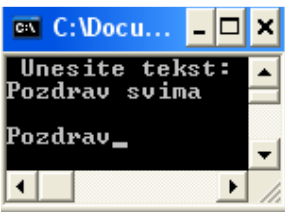
Испис на екрану б)

1.3. Шта се испишује на екрану извршавањем следећих програмских кодова ако се унесе текст Pozdrav svima:

а)

```
#include <stdio.h>
#define MAX 100

main()
{
    char poruka[MAX];
    printf(" Unesite tekst:\n");
    scanf("%s", poruka);
    printf("\n%s", poruka);
    getche();
    return 0;
}
```

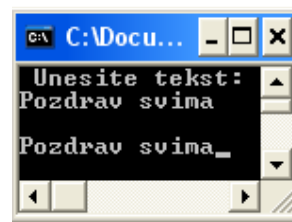


Испис на екрану а)

б)

```
#include <stdio.h>
#define MAX 100

main()
{
    char poruka[MAX];
    printf(" Unesite tekst:\n");
    gets(poruka);
    printf("\n%s", poruka);
    getche();
    return 0;
}
```

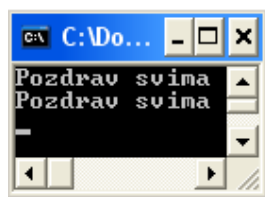


Испис на екрану б)

1.4. Саставити програм који штампа унети низ карактера ред по ред.

```
#include <stdio.h>
#define MAX 100

main()
{
    char s[MAX+1];
    gets(s);
    puts(s);
    getche();
    return 0;
}
```

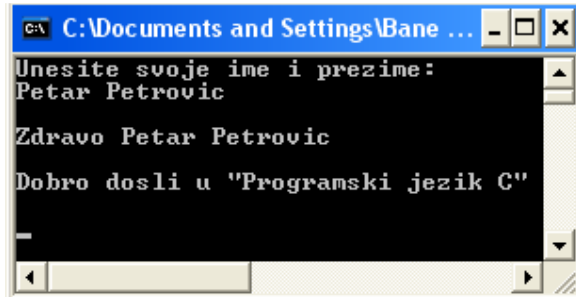


Испис на екрану

1.5. Шта се испишује на екрану извршавањем следећег програмског кода, ако се за унето име унесе Petar Petrovic:

```
#include <stdio.h>
#define MAX 100

main()
{
    char ime[MAX+1];
    const char naslov[]="Programski jezik C";
    puts("Unesite svoje ime i prezime: ");
    gets(ime);
    printf("\nZdravo %s\n", ime);
    printf("\nDobro dosli u \"%s\"\n\n", naslov);
    getche();
    return 0;
}
```

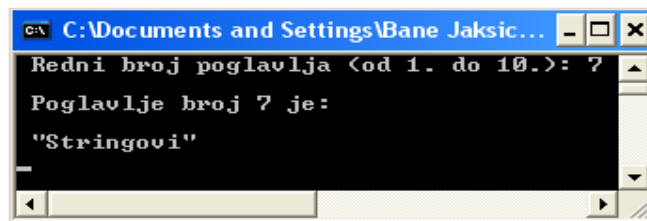


Испис на екрану

1.6. Саставити програм који за унети цео број **n** испишује одговарајуће поглавље дате књиге. Књига има следећа поглавља: Operatori, Ciklusi, Skokovi, Karakteri, Nizovi, Matrice, Stringovi, Pokazivaci, Strukture, Datoteke.

```
#include <stdio.h>
#define BROJ_POGLAVLJA 10
#define IME_POGLAVLJA 100

main()
{
    int n;
    const char poglavlja[BROJ_POGLAVLJA][IME_POGLAVLJA+1]=
    {"Operatori", "Ciklusi", "Skokovi", "Karakter", "Nizovi",
     "Matrice", "Stringovi", "Pokazivaci", "Strukture", "Datoteke"};
    do
    {
        printf(" Redni broj poglavlja (od 1. do %d.): ", BROJ_POGLAVLJA);
        scanf("%d", &n);
    }
    while(n<1 || n>BROJ_POGLAVLJA);
    printf("\n Poglavlje broj %d je:\n\n \"%s\"\n", n, poglavlja[n-1]);
    getche();
    return 0;
}
```

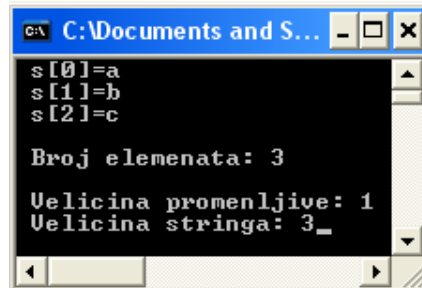


Испис на екрану

1.7. Шта се исписује на екрану извршавањем следећег програмског кода:

```
#include <stdio.h>

main()
{
    char s[] = {'a', 'b', 'c'};
    int BrElem = sizeof(s)/sizeof(char);
    int i;
    for(i=0; i<BrElem; i++)
        printf(" s[%d]=%c\n",i, s[i]);
    printf("\n Broj elemenata: %d \n", BrElem);
    printf("\n Velicina promenljive: %d", sizeof(char));
    printf("\n Velicina stringa: %d", sizeof(s));
    getche();
    return 0;
}
```

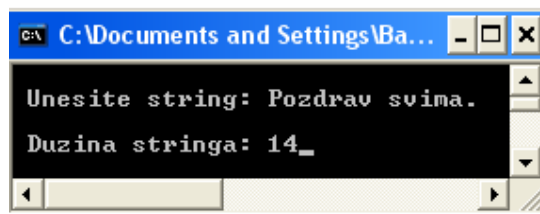
*Испис на екрану***1.2 Основне операције са стринговима****1.8.** Саставити програм за одређивање дужине унетог стринга (без null карактера):

- а) без употребе уграђене функције **strlen()**;
- б) употребом уграђене функције **strlen()**;
- в) саставити програм за одређивање који је дужи од два задата стринга (без null карактера) употребом уграђене функције **strlen()**;

а)

```
#include <stdio.h>
#define MAX 100

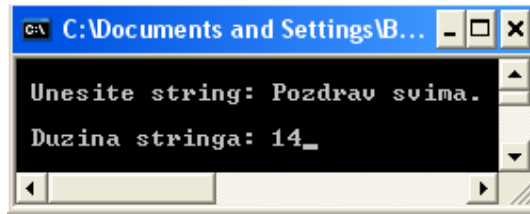
main()
{
    int n=0;
    char s[MAX+1];
    printf("\n Unesite string: ");
    gets(s);
    while(s[n]!='\0')
        n++;
    printf("\n Duzina stringa: %d", n);
    getche();
    return 0;
}
```

*Испис на екрану*

6)

```
#include <stdio.h>
#include <string.h>
#define MAX 100

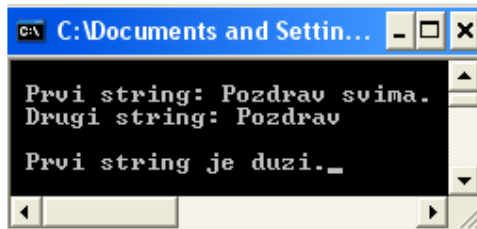
main()
{
    int n;
    char s[MAX+1];
    printf("\n Unesite string: ");
    gets(s);
    n=strlen(s);
    printf("\n Duzina stringa: %d", n);
    getch();
    return 0;
}
```

*Испис на екрану*

B)

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n1, n2;
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    n1=strlen(s1);
    n2=strlen(s2);
    if(n1==n2)
        printf("\n Stringovi su iste duzine.");
    else if(n1>n2)
        printf("\n Prvi string je duzi.");
    else
        printf("\n Drugi string je duzi.");
    getch();
    return 0;
}
```

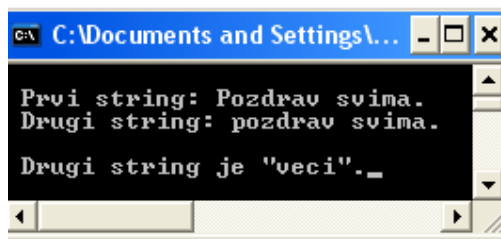
*Испис на екрану***1.9.** Саставити програм који врши:

- a) поређење два задата стринга помоћу функције **strcmp()**.
- b) првих **n** карактера два стринга помоћу функције **strncmp()**.

a)

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
```

*Испис на екрану*

```

if(strcmp(s1,s2)==0)
    printf("\n Stringovi su isti.");
else if(strcmp(s1,s2)>0)
    printf("\n Prvi string je \\"veci\\".");
else
    printf("\n Drugi string je \\"veci\\".");
getche();
return 0;
}

```

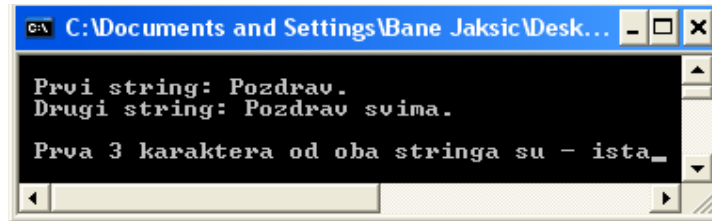
6)

```

#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    if(strncmp(s1,s2,3)==0)
        printf("\n Prva 3 karaktera od oba stringa su - ista");
    getche();
    return 0;
}

```



Испис на екрану

1.10. Саставити програм који:

- a) копира један стринг у други (од почетка другог стринга) помоћу функције **strcpy()**.
 б) копира првих **n** карактера једног стринга у други (од почетка другог стринга) помоћу функције **strncpy()**.

a)

```

#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n1, n2;
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    printf("\n Novi string: ");
    strcpy(s1,s2);
    puts(s1);
    getche();
    return 0;
}

```

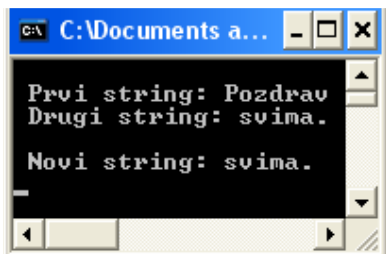
б)

```

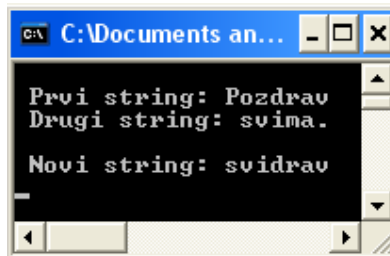
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n1, n2;
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    printf("\n Novi string: ");
    strncpy(s1,s2,3);
    puts(s1);
    getche();
    return 0;
}

```



Испис на екрану а)



Испис на екрану б)

1.11. Саставити програм који:

- а) копира један стринг у продужетку постојећег садржаја другог стринга помоћу функције **strcat()**;
б) копира првих **n** карактера од једног стринга у други, у продужетку постојећег садржаја другог стринга помоћу функције **strncat()**.

а)

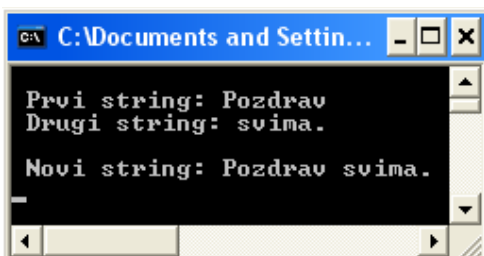
```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n1, n2;
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    printf("\n Novi string: ");
    strcat(s1,s2);
    puts(s1);
    getche();
    return 0;
}
```

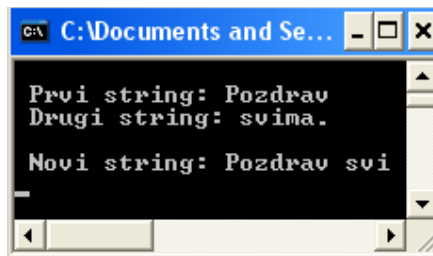
б)

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n1, n2;
    char s1[MAX+1], s2[MAX+1];
    printf("\n Prvi string: ");
    gets(s1);
    printf(" Drugi string: ");
    gets(s2);
    printf("\n Novi string: ");
    strncat(s1,s2,3);
    puts(s1);
    getche();
    return 0;
}
```



Испис на екрану а)

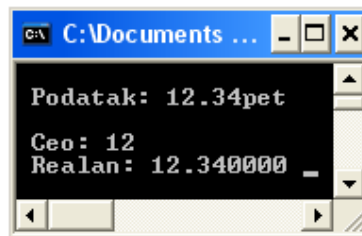


Испис на екрану б)

1.12. Саставити програм који конвертује стринг у цео и реалан број (ако је могуће) употребом уграђених функција **atoi()** и **atof()**. Исписати резултате.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

main()
{
    char s[MAX+1];
    printf("\n Podatak: ");
    gets(s);
    printf("\n Ceo: %d ", atoi(s));
    printf("\n Realan: %f ", atof(s));
    getche();
    return 0;
}
```

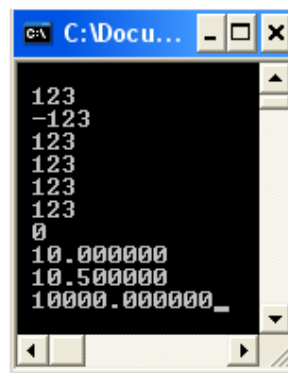


Испис на екрану

1.13. Шта се исписује на екрану извршавањем следећег програмског кода:

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

main()
{
    printf("\n %d", atoi("123"));
    printf("\n %d", atoi("-123"));
    printf("\n %d", atoi(" 123"));
    printf("\n %d", atoi("000123"));
    printf("\n %d", atoi("123 "));
    printf("\n %d", atoi("123abc"));
    printf("\n %d", atoi("abc123"));
    printf("\n %f", atof("10"));
    printf("\n %f", atof("10.5"));
    printf("\n %f", atof("10e+3"));
    getche();
    return 0;
}
```



Испис на екрану

1.14. Саставити програм који симулира калкулатор са четири основне аритметичке операције над реалним бројевима. Бројеви са улаза се прихватају као стрингови. Програм треба да обрађује произвољан број комплета све док не на улазу не прочита уместо броја слово. Исписати резултат операције.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 100

main()
{
    int znak, nepoznat=0;
    double broj1, broj2;
    char s[MAX+1];
    while(1)
    {
        /*Prihvatanje brojeva i operatora*/
        printf("\n Prvi broj: ");
```

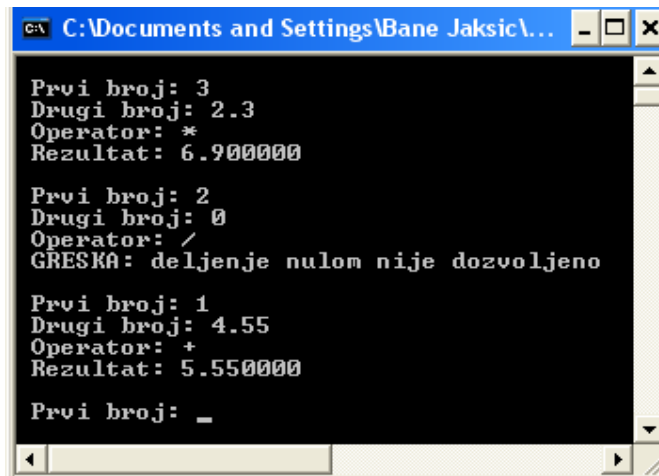
```

gets(s);
if(!isdigit(s[0]) && s[0]!='-')
{
    printf("\n");
    break;
}
broj1=atof(s);
printf(" Drugi broj: ");
gets(s);
if(!isdigit(s[0]) && s[0]!='-')
{
    printf("\n");
    break;
}
broj2=atof(s);
printf(" Operator: ");
znak=getchar();
getchar();

/*Proverava da li je komanda za kraj znak koji nije cifra*/
if(!(isdigit(s[0])))
{
    printf("\n");
    break;
}

/*Izracunavanje i prikaz rezultata*/
switch(znak)
{
    case'+': printf(" Rezultat: %f\n", broj1+broj2);
             break;
    case'-': printf(" Rezultat: %f\n", broj1-broj2);
             break;
    case'*': printf(" Rezultat: %f\n", broj1*broj2);
             break;
    case'/': if(broj2!=0)
              printf(" Rezultat: %f\n", broj1/broj2);
            else
              printf(" GRESKA: deljenje nulom nije dozvoljeno\n");
            break;
    default: printf(" Nepoznat operator\n");
             nepoznat=1;
             break;
}
if(nepoznat) break;
}
getche();
return 0;
}

```



Истис на екрану

1.3 Стрингови и функције

1.15. Саставити функцију за испис ниске карактера, а затим је тестирати у главном програму.

```
#include <stdio.h>

void StampaString(char s[])
{
    int i;
    for(i=0; s[i]; i++)
        putchar(s[i]);
}

main()
{
    StampaString("Zdravo\n");
    getche();
    return 0;
}
```



Испис на екрану

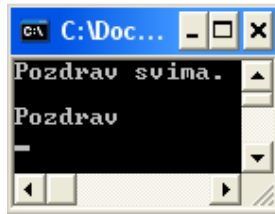
Уз ниску карактера није потребно преносити димензију уколико се поштује договор да се свака ниска завршава карактером '\0'.

1.16. Саставити функцију за учитавање једне речи, а затим је тестирати у главном програму.

```
#include <stdio.h>
#include <ctype.h>
#define MAX 100

void Ucitaj(char s[])
{
    int c, i=0;
    while(!isspace(c=getchar()))
        s[i++]=c;
    s[i]='\0';
}

main()
{
    char s[MAX];
    Ucitaj(s);
    printf("\n%s\n", s);
    getche();
    return 0;
}
```



Испис на екрану

1.17. Саставити функцију која:

- израчунава дужину стринга;
- копира стринг **s2** у стринг **s1**, претпоставља да у **s1** има довољно простора;
- надовезује стринг **s2** на крај стринга **s1**, претпоставља да у **s1** има довољно простора
- врши лексикографско поређење два стринга, враћа: 0 - уколико су стрингови једнаки, <0 - уколико је **s1** лексикографски испред **s2**, >0 - уколико је **s1** лексикографски иза **s2**;
- проналази прву позицију карактера **c** у стрингу **s**, враћа позицију на којој је **c**, односно -1 уколико **s** не садржи **c**;

- ђ) проналази последњу позицију карактера **c** у stringу **s**, враћа позицију на којој је **c**, односно -1 уколико **s** не садржи **c**;
 е) проверава да ли string **s1** садржи string **s2**, враћа позицију на којој **s2** почиње, односно -1 уколико га нема.

Тестирати формиране функције у главном програму.

```
#include <stdio.h>
#define MAX 100

/*Izracunava duzinu stringa*/
int Duzina(char s[])
{
    int i;
    /* Uslov s[i] je ekvivalentan uslovu s[i]!='\0' ili uslovu s[i]! = 0*/
    for(i = 0; s[i]; i++);
    return i;
}

/*Kopira string s2 u string s1. Pretpostavlja da u s1 ima dovoljno prostora.*/
void Kopiraj(char s1[], char s2[])
{
    /*Kopira karakter po karakter, sve dok nije iskopiran karakter '\0'*/
    int i;
    for(i=0; (s1[i]=s2[i]) != '\0'; i++);
}

/*Nadovezuje string s2 na kraj stringa s1.
Pretpostavlja da u s1 ima dovoljno prostora.*/
void Nadovezi(char s1[], char s2[])
{
    int i, j;
    /*Pronalazimo kraj stringa s*/
    for(i=0; s1[i]; i++);
    /*Vrsi se kopiranje, slicno funkciji Kopiraj*/
    for(j=0; s1[i]=s2[j]; j++, i++);
}

/* Vrsi leksikografsko poredjenje dva stringa.
Vraca :
0 - ukoliko su stringovi jednaki
<0 - ukoliko je s1 leksikografski ispred s2
>0 - ukoliko je s1 leksikografski iza s2*/
int Poredi(char s1[], char s2[])
{
    /*Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
    int i;
    for(i = 0; s1[i]==s2[i]; i++)
        if(s1[i] == '\0') /*Naisli smo na kraj oba stringa,a nismo nasli razliku*/
            return 0;
    /*s1[i] i s2[i] su prvi karakteri u kojima se niske razlikuju.
    Na osnovu njihovog odnosa, odredjuje se odnos stringova*/
    return s1[i] - s2[i];
}

/*Pronalazi prvu poziciju karaktera c u stringu s.
Vraca poziciju na kojoj je c, odnosno -1 ukoliko s ne sadrzi c*/
int PrvaPozicija(char s[], char c)
{

```

```

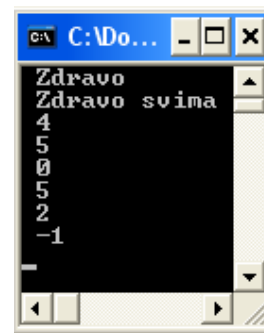
    int i;
    for(i=0; s[i]; i++)
        if(s[i] == c) return i;
    /*Nije nadjeno*/
    return -1;
}

/*Pronalazi poslednju poziciju karaktera c u stringu s.
Vraca poziciju na kojoj je c, odnosno -1 ukoliko s ne sadrzi c*/
int PoslednjaPozicija(char s[], char c)
{
    /*Pronalazimo kraj stringa s1*/
    int i;
    for(i=0; s[i]; i++);
    /*Krecemo od kraja i trazimo s2 unazad*/
    for(i--; i>=0; i--)
        if(s[i] == c) return i;
    /*Nije nadjeno*/
    return -1;
}

/*Proverava da li string s1 sadrzi string s2.
Vraca poziciju na kojoj s2 pocinje, odnosno -1 ukoliko ga nema*/
int StringUString(char s1[], char s2[])
{
    int i, j;
    /*Proveravamo da li s2 pocinje na svakoj poziciji i*/
    for(i=0; s1[i]; i++)
        /*Poredimo s2 sa s1 pocevsi od poziciji i sve dok ne naidjemo na razliku*/
        for(j=0; s1[i+j] == s2[j]; j++)
            /*Nismo naisli na razliku a ispitali smo sve karaktere niske s2*/
            if(s2[j+1]!='\0') return i;
    /*Nije nadjeno*/
    return -1;
}

main()
{
    char s[MAX], t[]="Zdravo", u[]=" svima";
    Kopiraj(s, t);
    printf(" %s\n", s);
    Nadovezi(s, u);
    printf(" %s\n", s);
    printf(" %d\n", PrvaPozicija("racunari", 'n'));
    printf(" %d\n", PoslednjaPozicija("racunari", 'a'));
    printf(" %d\n", StringUString("racunari", "rac"));
    printf(" %d\n", StringUString("racunari", "ari"));
    printf(" %d\n", StringUString("racunari", "cun"));
    printf(" %d\n", StringUString("racunari", "cna"));
    getch();
    return 0;
}

```



Испис на екрану

1.18. Саставити функцију која:

- а) израчунава вредност коју представља карактер у датој основи, функција враћа -1 уколико цифра није валидна, нпр. цифра 'B' у основи 16 има вредност 11, цифра '8' није валидна у основи 6;
 б) израчунава вредност целог броја који је записан у датом низу карактера у датој основи, за израчунавање се користи Хорнерова шема.

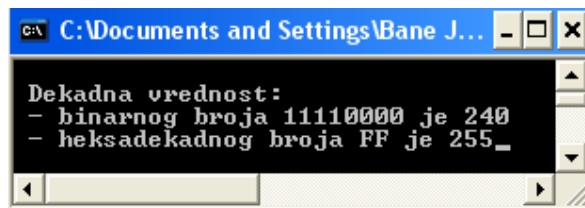
Тестирати формиране функције у главном програму.

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

/* Pomocna funkcija koja izracunava vrednost
koju predstavlja karakter u datoj osnovi*/
int DigitValue(char c, int base)
{
    /*Proveravamo obicne cifre*/
    if(isdigit(c) && c < '0'+base)
        return c-'0';
    /*Proveravamo slovne cifre za mala slova*/
    if('a'<=c && c < 'a'+base-10)
        return c-'a'+10;
    /*Proveravamo slovne cifre za velika slova*/
    if('A'<=c && c < 'A'+base-10)
        return c-'A'+10;
    return -1;
}

/*Funkcija izracunava vrednost celog broja koji
je zapisan u datom nizu karaktera u datoj osnovi.*/
int btoi(char s[], int base)
{
    int sum=0, i, vr;
    /*Obradjuju se karakteri sve dok su cifre*/
    for(i=0; (vr=DigitValue(s[i], base)) != -1; i++)
        sum = base*sum + vr;
    return sum;
}

main()
{
    char bin[] = "11110000";
    char hex[] = "FF";
    printf("\n Dekadna vrednost:");
    printf("\n - binarnog broja %s je %d", bin, btoi(bin,2));
    printf("\n - heksadekadnog broja %s je %d", hex, btoi(hex,16));
    getche();
    return 0;
}
```



Испис на екрану

1.19. Саставити функцију која проверава да ли дат низ карактера представља исправан хексадецимални број у облику **#број** и функцију која претвара дати низ карактера који представља хексадецимални број у декадни. Функције тестирати у главном програму.

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

int cifra(char c)
{
    if(c >= '0' && c <= '9') return 1;
    return 0;
}

int slovoa_f(char c)
{
    if(c >= 'a' && c <= 'f') return 1;
    return 0;
}

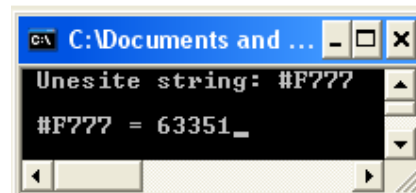
int slovoA_F(char c)
{
    if(c >= 'A' && c <= 'F') return 1;
    return 0;
}

int ispravan(char hex[])
{
    int i;
    if(hex[0] != '#') return 0;
    for(i=1; hex[i] != '\0'; i++)
        if(cifra(hex[i])==0 && slovoa_f(hex[i])==0 && slovoA_F(hex[i])==0)
            return 0;
    return 1;
}

int hexudek(char hex[])
{
    int i, rez=0;
    int d = strlen(hex);
    for(i=1; hex[i] != '\0'; i++)
    {
        if(cifra(hex[i]) == 1)
            rez += (hex[i]-'0') * (int)pow(16, d-i-1);
        else if(slovoa_f(hex[i]) == 1)
            rez += (hex[i]-'a'+10) * (int)pow(16, d-i-1);
        else rez += (hex[i]-'A'+10) * (int)pow(16, d-i-1);
    }
    return rez;
}

main()
{
    char niz[10];
    printf(" Unesite string: "); scanf("%s", &niz);
    if(ispravan(niz) == 0) printf("\n Broj nije ispravan.");
    else printf("\n %s = %d", niz, hexudek(niz));
    getch(); return 0;
}
```

Ради лакшег решавања,
уводимо помоћне функције које
испитују да ли дати карактер
представља цифру, слово између
a и **f** и слово између **A** и **F**.



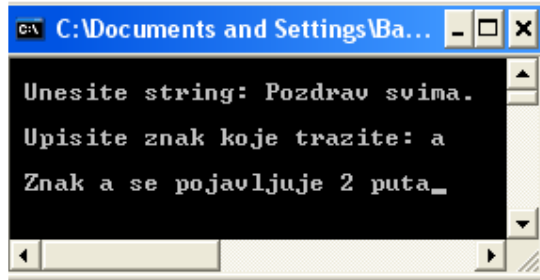
Испис на екрану

1.4 Претраживање стрингова

1.20. Саставити програм који ће исписати колико се пута унети знак појављује у уčitаном стрингу.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define MAX 100

main()
{
    char a[MAX+1], b;
    int n, i, br=0;
    printf("\n Unesite string: ");
    gets(a);
    n=strlen(a);
    printf("\n Upisite znak koje trazite: ");
    scanf("%c",&b);
    for(i=0; i<n; i++)
        if(a[i]==b || a[i]==toupper(b))
            br++;
    if(br!=0)
        printf("\n Znak %c se pojavljuje %d puta", b, br);
    else printf("\n Znaka %c nema u stringu", b);
    getche();
    return 0;
}
```

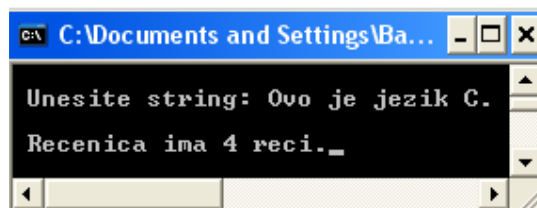


Испис на екрану

1.21. Саставити програм који ће исписати колико речи има учитана реченица.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char a[MAX+1];
    int n, i, br=0;
    printf("\n Unesite string: ");
    gets(a);
    n=strlen(a);
    for(i=0; i<n; i++)
        if(a[i]==' ' && a[i+1]!=' ')
            br++;
    printf("\n Recenica ima %d reci.", br+1);
    getche();
    return 0;
}
```

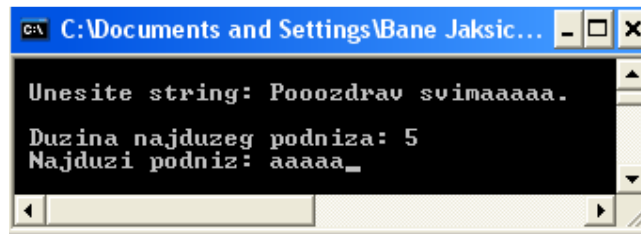


Испис на екрану

1.22. Саставити програм који у задатом стрингу исписује дужину најдужег низа узастопно једнаких знакова. Исписати тај низ.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char a[MAX+1];
    char znak;
    int n, i, br=1, max=0;
    printf("\n Unesite string: ");
    gets(a);
    n=strlen(a);
    for(i=0; i<n; i++)
    {
        if(a[i]== a[i+1])
        {
            br++;
            if(br>max)
            {
                max=br;
                znak=a[i];
            }
        }
        else br=1;
    }
    printf("\n Duzina najduzeg podniza: %d", max);
    printf("\n Najduzi podniz: ");
    for(i=0; i<max; i++)
        printf ("%c", znak);
    getch();
    return 0;
}
```



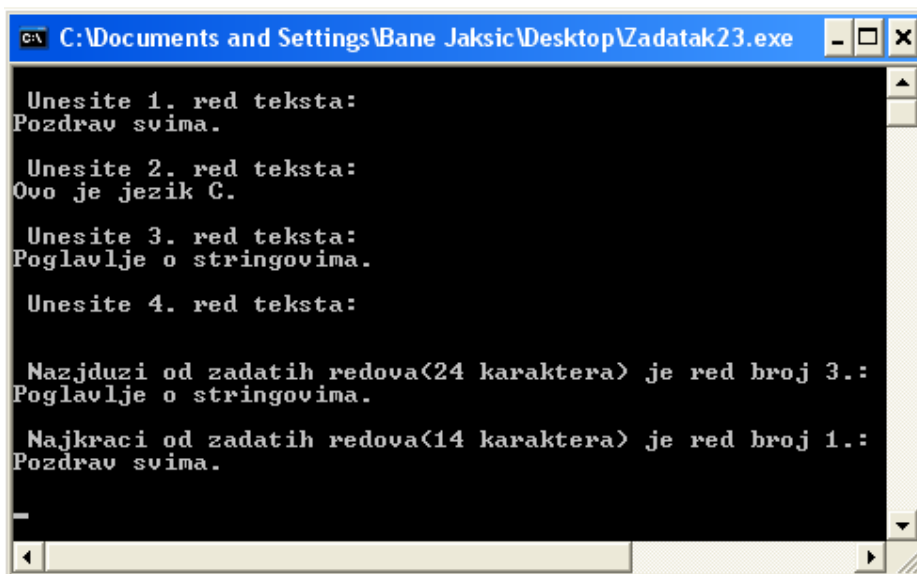
Испис на екрану

1.23. Саставити програм за одређивање најдужег и најкраћег реда од 20 задатих редова текста. STOP код при задавању редова текста може бити празан ред.

```
#include <stdio.h>
#include <string.h>
#define BROJ 20
#define DUZINA 100

main()
{
    char stringovi[BROJ][DUZINA+1];
    char stringMax[DUZINA+1], stringMin[DUZINA+1];
    int i, imax, imin, d, dmin, dmax;
    printf("\n Unesite 1. red teksta:\n");
    gets(stringovi[0]);
    imax=imin=0;
    dmax=dmin=strlen(stringovi[0]);
    strcpy(stringMax, stringovi[0]);
    strcpy(stringMin, stringovi[0]);
    for(i=1; i<BROJ; i++)
    {
        printf("\n Unesite %d. red teksta:\n", i+1);
```

```
    gets(stringovi[i]);
    if((d=strlen(stringovi[i]))==0)
        break;
    if(d>dmax)
    {
        imax=i;
        dmax=d;
        strcpy(stringMax, stringovi[i]);
    }
    if(d<dmin)
    {
        imin=i;
        dmin=d;
        strcpy(stringMin, stringovi[i]);
    }
}
printf("\n Nazjduzi od zadatih redova(%d karaktera)", dmax);
printf("je red broj %d.\n%s\n", imax+1, stringMax);
printf("\n Najkraci od zadatih redova(%d karaktera)", dmin);
printf("je red broj %d.\n%s\n\n", imin+1, stringMin);
getche();
return 0;
}
```



```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak23.exe
Unesite 1. red teksta:
Pozdrav svima.
Unesite 2. red teksta:
Ovo je jezik C.
Unesite 3. red teksta:
Poglavlje o stringovima.
Unesite 4. red teksta:

Nazjduzi od zadatih redova(24 karaktera) je red broj 3.:
Poglavlje o stringovima.
Najkraci od zadatih redova(14 karaktera) je red broj 1.:
Pozdrav svima.
```

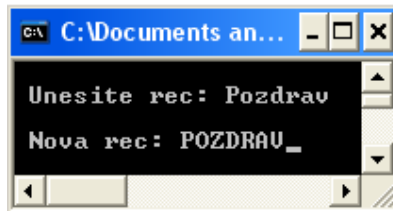
Испис на екрану

1.5 Уређивање стрингова

1.24. Саставити програм који у задатом стрингу (једна реч) врши конверзију свих малих слова у велика, а осатала не мења.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s[MAX+1];
    int i;
    printf("\n Unesite rec: ");
    scanf("%s", s);
    for(i=0; i<strlen(s);i++)
    {
        if(s[i]>='a' && s[i]<='z')
            s[i]-='a'-'A';
    }
    printf("\n Nova rec: %s", s);
    getche();
    return 0;
}
```



Испис на екрану

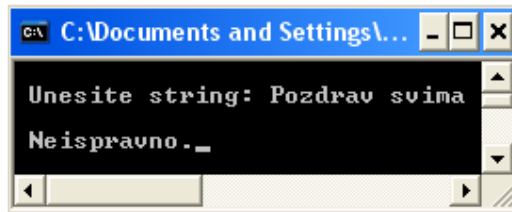
1.25. Саставити програм који учитава стринг, који представља реченицу, и који проверава да ли је реченица исправно унета. Исправно унета реченица задовољава следеће услове:

- реченица мора почети великим словом и завршити се тачком;
- речи су произвољни подстрингови који могу садржати само мала слова.
- речи могу бити раздвојени једним размаком (SPACE), зарезом или тачка-зарезом.

На излазу исписати обавештење да ли је реченица исправно унета.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int i, duz, ind=1;
    char s[MAX+1];
    printf("\n Unesite string: ");
    gets(s);
    duz=strlen(s);
    if((s[0]<'A' || s[0]>'Z') || s[duz-1]!='.')
        ind=0;
    if(ind==1)
    {
        for(i=0; i<duz; i++)
        {
            if((s[i]<'a' || s[i]>'z') && s[i]!=' ' &&
                s[i]!=',' && s[i]!=';')
                ind=0;
            if((s[i]==' ' || s[i]==',' || s[i]==';') &&
                (s[i+1]!=' ' || s[i+1]!=',' || s[i+1]!=';'))
                ind=0;
        }
    }
    if(ind) printf("\n Ispravno.");
    else printf("\n Neispravno.");
    getche();
    return 0;
}
```



Испис на екрану

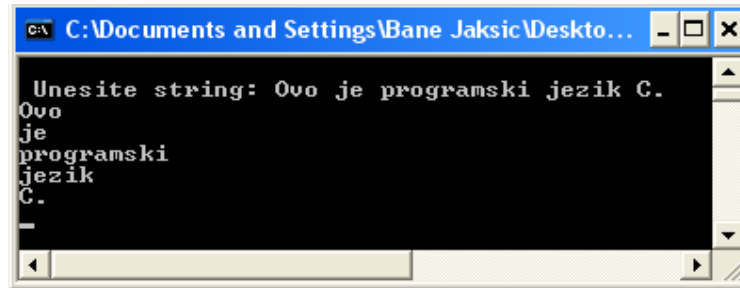
1.26. Саставити програм који ће речи задате реченице исписати једну испод друге.

```

#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s[MAX+1];
    int n, i, k, t=0;
    printf("\n Unesite string: ");
    gets(s);
    n=strlen(s);
    for(i=0; i<=n; i++)
    {
        if(s[i]==' ' || s[i]==',' || i==n)
        {
            for(k=t; k<=i; k++)
                printf("%c",s[k]);
            if(s[i]==',' && s[i+1]!=' ')
                t=i+2;
            else
            {
                t=i+1;
                printf("\n");
            }
        }
    }
    getche();
    return 0;
}

```



Испис на екрану

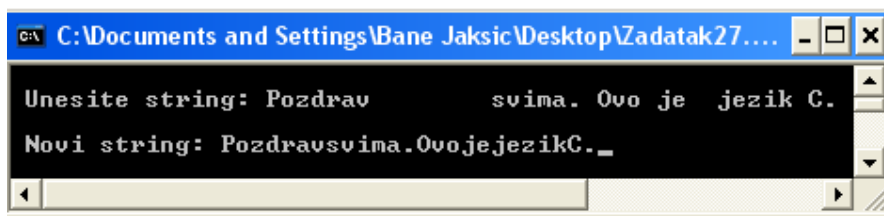
1.27. Саставити програм који за унети стринг уклања све размаке и табулаторе. Исписати нови стринг.

```

#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int n, i;
    char s[MAX+1];
    printf("\n Unesite string: ");
    gets(s);
    n=strlen(s);
    printf("\n Novi string: ");
    for(i=0; i<n; i++)
        if(s[i] != ' ' && s[i] != '\t')
            printf("%c", s[i]);
    getche();
    return 0;
}

```



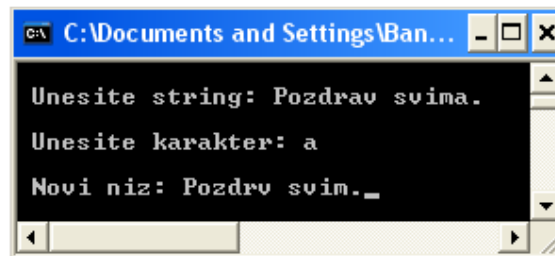
Испис на екрану

1.28. Саставити функцију која уклања задати карактер у унетом стрингу. Затим, тестирати функцију у главном програму за конкретан стринг и карактер.

```
#include <stdio.h>
#define MAX 100

void Ukloni(char s[], char c)
{
    int i,j=0;
    for(i=0; s[i]!='\0';i++)
        if(s[i]!=c)
        {
            s[j]=s[i];
            j++;
        }
    s[j]='\0';
}

main()
{
    char s[MAX+1];
    char c;
    printf("\n Unesite string: ");
    gets(s);
    printf("\n Unesite karakter: ");
    scanf("%c", &c);
    Ukloni(s, c);
    printf("\n Novi niz: %s", s);
    getch();
    return 0;
}
```

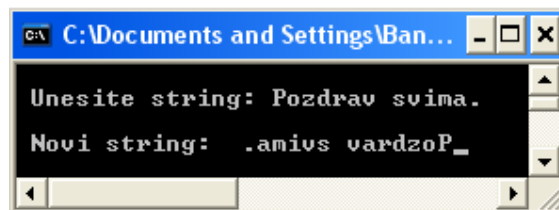


Испис на екрану

1.29. Саставити програм који уčitани стринг испишује уназад (с десна на лево).

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s[MAX];
    int n,i;
    printf("\n Unesite string: ");
    gets(s);
    n=strlen(s);
    printf("\n Novi string: ");
    for(i=n; i>=0; i--) printf("%c",s[i]);
    getch(); return 0;
}
```

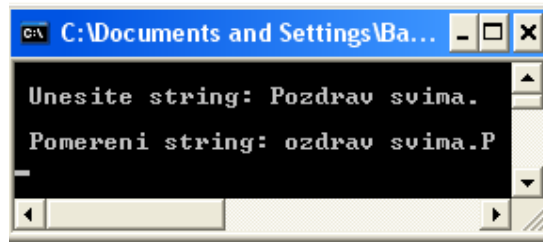


Испис на екрану

1.30. Саставити програм за циклично премештање знакова у задатом стрингу за једно место у лево. Исписати новокреирани стринг.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    int i, n;
    char pom, s[MAX+1];
    printf("\n Unesite string: ");
    gets(s);
    n=strlen(s);
    pom=s[0];
    for(i=0; i<n-1; i++)
        s[i]=s[i+1];
    s[n-1]=pom;
    printf("\n Pomereni string: ");
    puts(s);
    getche();
    return 0;
}
```



Испис на екрану

1.31. Саставити програм за циклично премештање стрингова задатог низа за једно место у лево. Стрингови се уносе један испод другог, тј. одвајају се тастером ENTER. Исписати новокреирани списак.

```
#include <stdio.h>
#include <string.h>
#define MAX 30
#define BROJ 10

main()
{
    int i, n;
    char pom[MAX+1], spisak[BROJ][MAX+1];
    printf("\n Unesite n: ");
    scanf("%d", &n);
    getchar(); /*Citanje koda ENTER*/
    printf("\n Spisak: \n");
    for(i=0; i<n; i++)
        gets(spisak[i]);
    strcpy(pom, spisak[0]);
    for(i=0; i<n-1; i++)
        strcpy(spisak[i], spisak[i+1]);
    strcpy(spisak[n-1], pom);
    printf("\n Novi spisak: \n");
    for(i=0; i<n; i++)
        puts(spisak[i]);
    getche();
    return 0;
}
```



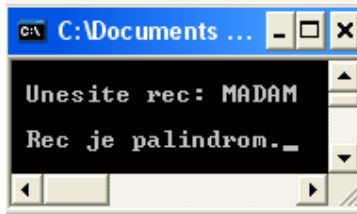
Испис на екрану

1.6 Сортирање стрингова

1.32. Саставити програм који за уčitану реч испишује поруку да ли је она палиндром. Реч је палиндром ако се исто чита као од почетка ка крају и од краја ка почетку.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s[MAX+1];
    int n, i, k=0;
    printf("\n Unesite rec: ");
    gets(s);
    n=strlen(s);
    for(i=0; i<n; i++)
        if(s[i]!=s[n-i-1])
            k=1;
    if(k==0)
        printf("\n Rec je palindrom.");
    else
        printf ("\n Rec nije palindrom.");
    getch();
    return 0;
}
```

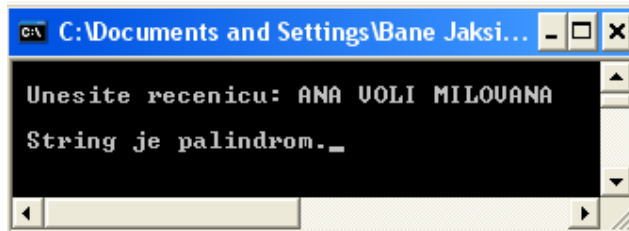


Испис на екрану

1.33. Саставити програм који за уčitану реченицу испишује поруку да ли је она палиндром. Реченица је палиндром ако се исто чита као од почетка ка крају и од краја ка почетку.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

main()
{
    char s[MAX+1];
    int n, i, k=0, br=0, j;
    printf("\n Unesite recenicu: ");
    gets(s);
    n=strlen(s);
    /*broji razmake*/
    for(i=0; i<n; i++)
        if (s[i]==' ')
            br++;
    /*izbacuje razmake*/
    for(i=0; i<br; i++)
        for(j=0; j<n; j++)
        {
            if(s[j]==' ')
            {
                s[j]=s[j+1];
                s[j+1]=' ';
            }
        }
}
```



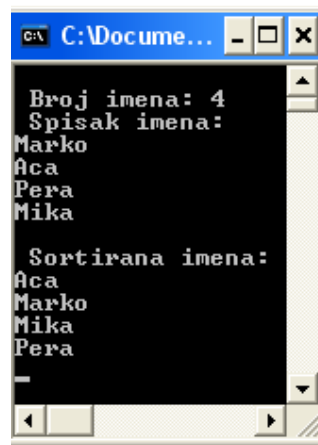
Испис на екрану

```
/*proverava da li je ucitani string palindrom*/
n=n-br-1;
for(i=0; i<n; i++)
    if(s[i]!=s[n-i])
        k=1;
if(k==0) printf("\n String je palindrom.");
else printf("\n String nije palindrom.");
getche();
return 0;
}
```

1.34. Саставити програм за сортирање задатог низа списка по абecedном реду. Исписати новокреирани списак.

```
#include <stdio.h>
#include <string.h>
#define BROJ 10
#define MAX 30

main()
{
    int i, j, n;
    char pom[MAX+1], ime[BROJ][MAX+1];
    printf("\n Broj imena: ");
    scanf("%d", &n);
    getchar(); /*Citanje koda ENTER*/
    printf(" Spisak imena: \n");
    /*Citanje spiska*/
    for(i=0; i<n; i++)
        gets(ime[i]);
    /*Sortiranje niza*/
    for(i=0; i<n-1; i++)
        for(j=i+1; j<n; j++)
        {
            if(strcmp(ime[i], ime[j]) > 0)
            {
                strcpy(pom, ime[i]);
                strcpy(ime[i], ime[j]);
                strcpy(ime[j], pom);
            }
        }
    /*Prikaz rezultujuceg stringa*/
    printf("\n Sortirana imena:\n");
    for(i=0; i<n; i++)
        puts(ime[i]);
    getche();
    return 0;
}
```



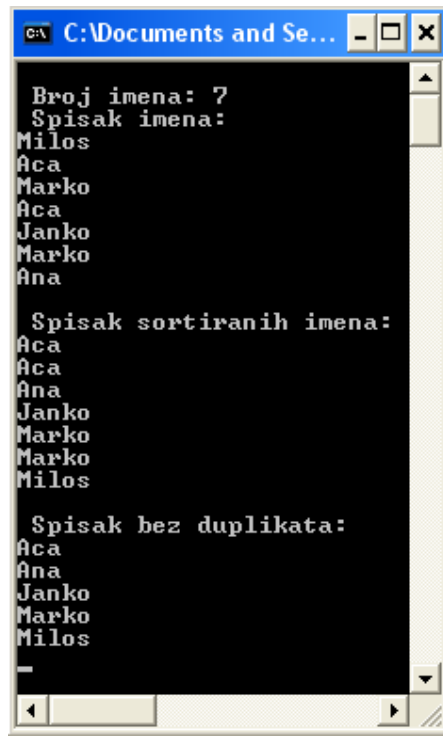
Испис на екрану

1.35. Саставити програм који уклања дубликате из датог списка имена и исписује новокреирани списак. Претходно сортирати лексикографски списак имена.

```
#include <stdio.h>
#include <string.h>
#define BROJ 100
#define MAX 30

void Sortiraj(char ime[][MAX], int n)
{
    int i, j, min;
    char pom[MAX];
    for(i=0; i<n-1; i++)
    {
        min = i;
        for(j=i+1; j<n; j++)
            if(strcmp(ime[j], ime[min])<0)
                min=j;
        if(min!=i)
        {
            strcpy(pom, ime[i]);
            strcpy(ime[i], ime[min]);
            strcpy(ime[min], pom);
        }
    }
}

main()
{
    char ime[BROJ][MAX];
    int i, n, tekuci;
    printf("\n Broj imena: ");
    scanf("%d", &n);
    getchar(); /*Citanje koda ENTER*/
    printf(" Spisak imena: \n");
    /*Citanje spiska*/
    for(i=0; i<n; i++)
        gets(ime[i]);
    Sortiraj(ime,n);
    /*Prikaz sortiranog spiska*/
    printf("\n Spisak sortiranih imena:\n");
    for(i=0; i<n; i++)
        puts(ime[i]);
    /*Uklanjanje duplikata*/
    for(i=1, tekuci=0; i<n; i++)
    {
        while(strcmp(ime[tekuci],ime[i]) == 0)
            i++;
        strcpy(ime[++tekuci],ime[i]);
    }
    n=tekuci;
    /*Prikaz spiska bez duplikata*/
    printf("\n Spisak bez duplikata:\n");
    for(i=0; i<n ;i++)
        puts(ime[i]);
    getch();
    return 0;
}
```



```
C:\Documents and Se...
Broj imena: 7
Spisak imena:
Milos
Aca
Marko
Aca
Janko
Marko
Ana

Spisak sortiranih imena:
Aca
Aca
Ana
Janko
Marko
Marko
Milos

Spisak bez duplikata:
Aca
Ana
Janko
Marko
Milos
```

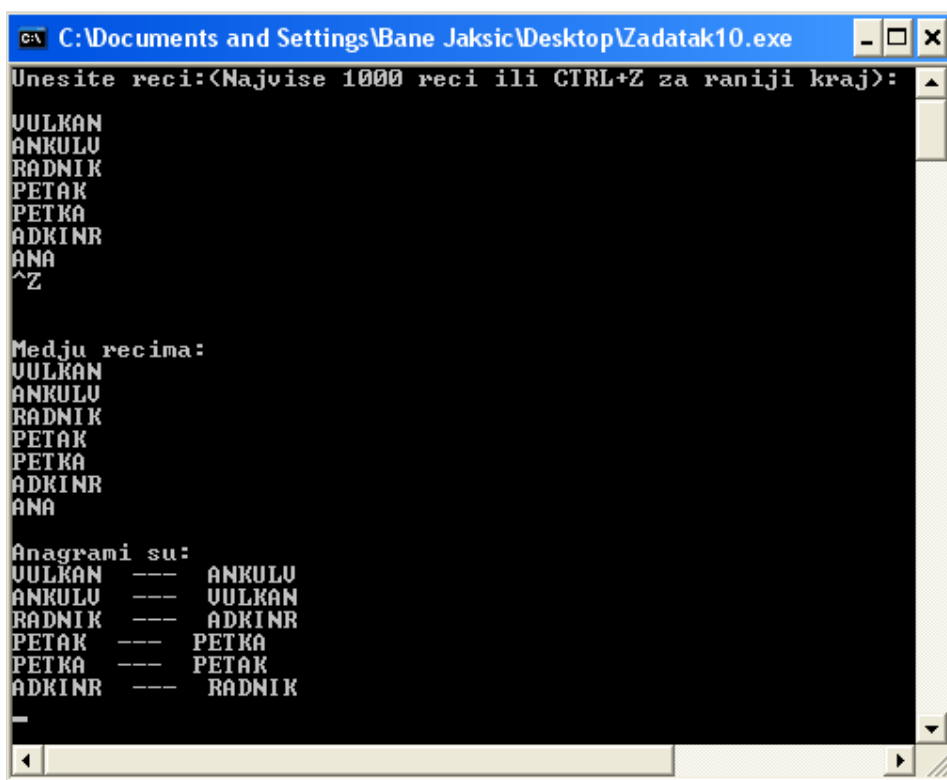
Испис на екрану

1.36. Саставити програм који проналази и исписује све анаграме у датом низу речи. За две речи кажемо да су анаграми ако се од једне речи може добити друга премештањем слова у речи.

```
#include <stdio.h>
#include <string.h>
#define MAX_DUZINA 20
#define MAX_RECI 1000

void Sortiraj(char a[], int n)
{
    int i, j, min;
    for(i=0; i<n-1; i++)
    {
        min=i;
        for(j=i+1; j<n; j++)
            if(a[j]<a[min])
                min=j;
        if(min!=i)
        {
            char pom=a[i];
            a[i]=a[min];
            a[min]=pom;
        }
    }
}

main()
{
    char reci[MAX_RECI][MAX_DUZINA];
    char kopije[MAX_RECI][MAX_DUZINA]; /*niz u kome cuvamo sortirane kopije*/
    int i=0, n, j;
    printf("Unesite reci:
           (Najvise %d reci ili CTRL+Z za raniji kraj):\n\n", MAX_RECI);
    while(scanf("%s", reci[i]) == 1)
    {
        strcpy(kopije[i], reci[i]);
        Sortiraj(kopije[i], strlen(kopije[i]));
        i++;
        if(i==MAX_RECI)
        {
            printf("Niz je popunjen! Prekidamo dalji unos!\n");
            break;
        }
    }
    n=i;
    printf("\nMedju recima:\n");
    for(i=0; i<n; i++)
        printf("%s\n", reci[i]);
    printf("\nAnagrami su:\n");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(i != j && strcmp(kopije[i], kopije[j])==0)
                printf("%s --- %s\n", reci[i], reci[j]);
    getch();
    return 0;
}
```



```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak10.exe
Unesite reci:<Najviše 1000 reci ili CTRL+Z za raniji kraj>:
VULKAN
ANKULU
RADNIK
PETAK
PETKA
ADKINR
ANA
^Z

Medju recima:
VULKAN
ANKULU
RADNIK
PETAK
PETKA
ADKINR
ANA

Anagrami su:
VULKAN --- ANKULU
ANKULU --- VULKAN
RADNIK --- ADKINR
PETAK --- PETKA
PETKA --- PETAK
ADKINR --- RADNIK
```

Испис на екрану

2 ПОКАЗИВАЧИ

2.1 Основне конструкције програма са показивачима

Табела 2.1: Декларације са показивачима

| декларација | значење имена x |
|--------------|---|
| T x | објекат типа T |
| T x[] | (отворени) низ објеката типа T |
| T x[n] | низ од n објеката типа T |
| T *x | показивач на објекат типа T |
| T **x | показивач на показивач типа T |
| T *x[] | низ показивача на објекат типа T |
| T *(x[]) | низ показивача на објекат типа T |
| T (*x)[] | показивач на низ објеката типа T |
| T x() | функција која враћа објекат типа T |
| T *x() | функција која враћа показивач на објекат типа T |
| T (*x()) | функција која враћа показивач на објекат типа T |
| T (*x)() | показивач на функцију која враћа објекат типа T |
| T (*x[n])() | низ од n показивача на функцију која враћа објекат типа T |

2.1. Објаснити сваку наредбу у делу програмског кода:

а)

```
int x=1, y=1;
int *ip;
ip = &x;
y = *ip;
*ip = 0;
*ip+=10;
++*ip;
(*ip)++;
```

б)

```
int x, y, *px, *py;
px = &x;
*px = 0;
py = px;
*py += 1;
y = (*px)++;
```

Решење а):

```
int x=1, y=1;
int *ip;   ip је показивач на int, односно *ip је типа int
ip = &x;   ip чува адресу променљиве x, тј. ip показује на x
```

```

y = *ip;    y добија вредност онога што се налази на адреси коју чува променљива ip,
            односно, пошто ip чува адресу променљиве x, а x има вредност 1 то је и y 1
*ip = 0;    преко показивача мења се садржај на адреси коју чува ip, према томе x је сада 0
*ip+=10;    x је сада 10
++*ip;      x је сада 11
(*ip)++;    x је сада 12, заграда неопходна због приоритета оператора

```

Решење б):

```

int x, y, *px, *py;
px = &x;      px садржи адресу од x
*px = 0;      вредност x постаје 0
py = px;      py сада показује на x
*py += 1;     увећава x за 1
y = (*px)++;  сада је y = 1, а x = 2

```

2.2. Које вредности ће имати променљиве x и y после извршавања следећег блока наредби:

```

x=0; y=0;
ip = &x;
*ip = *ip + 10;
y = *ip + 1;
*ip += 1;
(*ip)++;
int *iq;
iq = ip;
int **ir;
ir = &iq;
**ir += 3;

```

Решење

```

x=0; y=0;
ip = &x;
*ip = *ip + 10;  увећава *ip за 10
y = *ip + 1;     вредност променљиве на коју показује ip
                 се увећава за 1 и додељује y
*ip += 1;        повећава променљиву на коју показује ip за 1
**ir += 1;       повећава променљиву на коју показује ip за 1
(*ip)++;         повећава променљиву на коју показује ip за 1
int *iq;         декларација показивача iq
iq = ip;         iq сада показује на исту променљиву као ip
int **ir;        декларација показивача на показивач на int
ir = &iq;         ir сада показује на показивач iq
**ir += 3;       исто као x=x+3

```

x=16, y=11

2.3. Шта се исписује на екрану извршавањем следећих програмских кодова:

a)

```

#include <stdio.h>

main()
{
    int x, prom=5, *pokaz;
    pokaz = &prom;
    x = *pokaz;
    printf(" prom= %d\n x= %d\n", prom, x);
    getch();
    return 0;
}

```



Испис на екрану

6)

```
#include <stdio.h>

main()
{
    int x=5;
    /*Adresu promenljive x zapamticemo u novoj promeljivoj.
    Nova promenljiva je tipa pokazivaca na int (int*)*/
    int *px;
    printf(" Adresa promenljive x: %p\n", &x);
    printf(" Vrednost promenljive x: %d\n", x);
    px=&x;
    printf(" Vrednost promenljive px (tj. px): %p\n", px);
    printf(" Vrednost promenljive na koju ukazuje px (tj. *px): %d\n", *px);
    /*Menjamo vrednost promenljive na koju ukazuje px*/
    *px=6;
    printf(" Vrednost promenljive na koju ukazuje px (tj. *px): %d\n", *px);
    /*Posto px sadrzi adresu promenljive x, ona ukazuje na x
    tako da je posredno promenjena i vrednost promenljive x*/
    printf(" Vrednost promenljive x: %d\n", x);
    getch();
    return 0;
}
```

```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak3...
Adresa promenljive x: 0022FF74
Vrednost promenljive x: 5
Vrednost promenljive px (tj. px): 0022FF74
Vrednost promenljive na koju ukazuje px (tj. *px): 5
Vrednost promenljive na koju ukazuje px (tj. *px): 6
Vrednost promenljive x: 6
```

Испис на екрану

2.4. Шта се исписује на екрану након извршавања следећег програмског кода:

```
#include <stdio.h>

main()
{
    double x=2.1, *px=&x;
    double y=0.1, *py=&y;

    /*Prikazuju se vrednosti pokazivaca. Specifikator %p u printf()-u
    ispisuje vrednost pokazivaca u heksadekadnoj notaciji*/
    printf(" Vrednost pokazivaca px je %p\n", px);
    printf(" Vrednost pokazivaca py je %p\n", py);

    /*Ispisujemo vrednosti na koje pokazuju pokazivaci*/
    printf(" px pokazuje na vrednost: %f\n", *px);
    printf(" py pokazuje na vrednost: %f\n", *py);

    /*Ispitujemo da li su pokazivaci jednaki, tj. da li
    pokazuju na ti objekat u memoriji*/
    if(px == py)
        printf(" Pokazivaci px i py pokazuju na isti objekat\n");
    else
        printf(" Pokazivaci px i py ne pokazuju na isti objekat\n");
}
```

```

py = px;

/*Prikazuju se vrednosti pokazivaca. Specifikator %p u printf()-u
ispisuje vrednost pokazivaca u heksadekadnoj notaciji*/
printf(" Vrednost pokazivaca px je %p\n", px);
printf(" Vrednost pokazivaca py je %p\n", py);

/*Ispisujemo vrednosti na koje pokazuju pokazivaci*/
printf(" px pokazuje na vrednost: %f\n", *px);
printf(" py pokazuje na vrednost: %f\n", *py);

/*Ispitujemo da li su pokazivaci jednaki, tj. da li
pokazuju na isti objekat u memoriji */
if(px == py)
    printf(" Pokazivaci px i py pokazuju na isti objekat\n");
else
    printf(" Pokazivaci px i py ne pokazuju na isti objekat\n");

px = &y;

/*Prikazuju se vrednosti pokazivaca. Specifikator %p u printf()-u
ispisuje vrednost pokazivača u heksadekadnoj notaciji*/
printf(" Vrednost pokazivaca px je %p\n", px);
printf(" Vrednost pokazivaca py je %p\n", py);

/*Ispisujemo vrednosti na koje pokazuju pokazivaci*/
printf(" px pokazuje na vrednost: %f\n", *px);
printf(" py pokazuje na vrednost: %f\n", *py);

/*Ispitujemo da li su pokazivaci jednaki, tj. da li
pokazuju na isti objekat u memoriji*/
if(px == py)
    printf(" Pokazivaci px i py pokazuju na isti objekat\n");
else
    printf(" Pokazivaci px i py ne pokazuju na isti objekat\n");

getche();
return 0;
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop\Za...
Vrednost pokazivaca px je 0022FF70
Vrednost pokazivaca py je 0022FF60
px pokazuje na vrednost: 2.100000
py pokazuje na vrednost: 0.100000
Pokazivaci px i py ne pokazuju na isti objekat
Vrednost pokazivaca px je 0022FF70
Vrednost pokazivaca py je 0022FF70
px pokazuje na vrednost: 2.100000
py pokazuje na vrednost: 2.100000
Pokazivaci px i py pokazuju na isti objekat
Vrednost pokazivaca px je 0022FF60
Vrednost pokazivaca py je 0022FF70
px pokazuje na vrednost: 0.100000
py pokazuje na vrednost: 2.100000
Pokazivaci px i py ne pokazuju na isti objekat

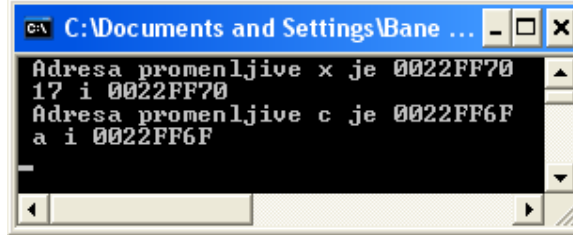
```

Испис на екрану

2.5. Пример употребе показивача на празан тип. Шта се исписује на екрану извршавањем следећег програмског кода:

```
#include <stdio.h>

main()
{
    void *pp; /*pp je pokazivac na prazan tip*/
    int x=2;
    char c='a';
    pp=&x; /*pp sada pokazuje na promenljivu x*/
    *(int *)pp = 17; /*x postaje 17*/
    /*Mora eksplicitno da se navede kastovanje u tip (int *)*/
    printf(" Adresa promenljive x je %p\n", &x);
    printf(" %d i %p\n", *(int*)pp, (int*)pp);
    pp = &c; /*pp sada pokazuje na promenljivu c*/
    printf(" Adresa promenljive c je %p\n", &c);
    printf(" %c i %p\n", *(char*)pp, (char*)pp);
    getch();
    return 0;
}
```



Испис на екрану

2.2 Показивачи као аргументи функција

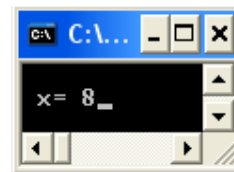
2.6. Који резултат извршења следећих програмских кодова:

a)

```
#include <stdio.h>

/*Funkcija inkrementira vrednost varijable, cija se adresa
prenosi u funkciju kao vrednost pokazivaca p.
Varijabli se pristupa pomocu indirekcije pokazivaca p.*/
void Increment(int *p)
{
    (*p)++;
}

main()
{
    int x=7;
    Increment(&x); /*argument je adresa varijable*/
    printf ("\n x= %d", x);
    getch();
    return 0;
}
```



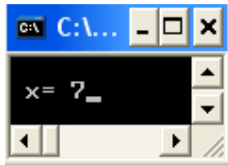
Испис на екрану а)

6)

```
#include <stdio.h>

void f(int *x)
{
    (*x) *= 2;
    (*x)++;
}

main()
{
    int x=3;
    f(&x);
    printf("\n x= %d", x);
    getche();
    return 0;
}
```



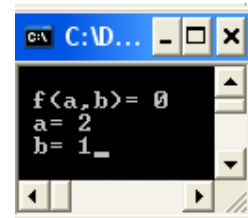
Испис на екрану б)

в)

```
#include <stdio.h>

int f(int a, int *b)
{
    a+=(*b);
    *b+=3;
    return a;
}

main()
{
    int a=2, b=-2;
    printf("\n f(a,b)= %d ", f(a,&b));
    printf("\n a= %d\n b= %d", a, b);
    getche();
    return 0;
}
```



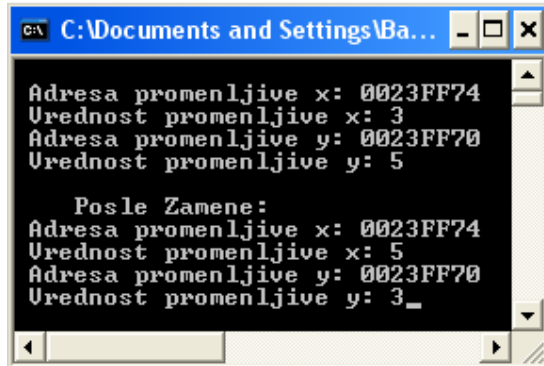
Испис на екрану в)

2.7. Саставити функцију која врши замену места две променљиве, а затим у главном програму исписати вредности и адресе променљивих пре и после замене. Пренос аргумената функције вршити преко показивача.

```
#include <stdio.h>

void Zamena(int *px, int *py)
{
    int tmp;
    tmp = *px;
    *px = *py;
    *py = tmp;
}

main()
{
    int x=3, y=5;
    printf("\n Adresa promenljive x: %p", &x);
    printf("\n Vrednost promenljive x: %d", x);
    printf("\n Adresa promenljive y: %p", &y);
    printf("\n Vrednost promenljive y: %d", y);
    Zamena(&x, &y);
    printf("\n\n Posle Zamene:");
    printf("\n Adresa promenljive x: %p", &x);
    printf("\n Vrednost promenljive x: %d", x);
    printf("\n Adresa promenljive y: %p", &y);
    printf("\n Vrednost promenljive y: %d", y);
    getche();
    return 0;
}
```



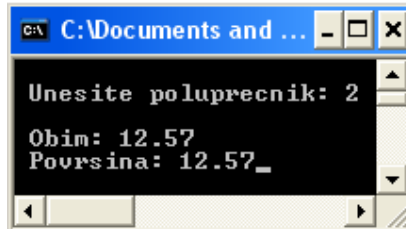
Испис на екрану

2.8. Саставити функцију за рачунање обима и површине круга, а затим у главном програму за унети полупречник круга **r** применити формирану функцију и исписати резултате. Пренос аргумената функције вршити преко показивача.

```
#include <stdio.h>
#define PI 3.141592

void ObimPovrsina(float r, float *O, float *P)
{
    *O=2*r*PI;
    *P=r*r*PI;
}

main()
{
    float r, ob, pov;
    printf("\n Unesite poluprecnik: ");
    scanf("%f",&r);
    ObimPovrsina(r, &ob, &pov);
    printf("\n Obim: %.2f\n Povrsina: %.2f", ob, pov);
    getche();
    return 0;
}
```



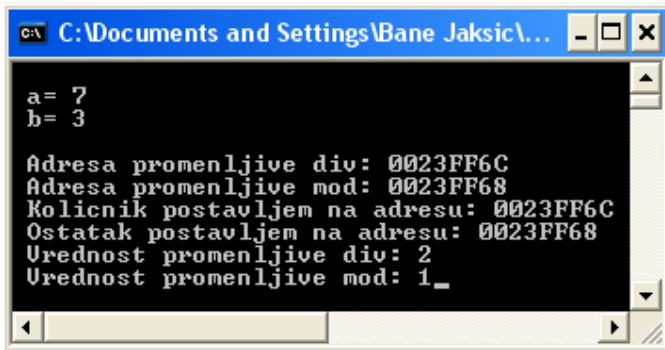
Испис на екрану

2.9. Саставити функцију за одређивање целобројног количника и остатка при целобројном дељењу два цела броја, а затим функцију тестирати у главном програму. Исписати адресе и вредности резултата. Пренос аргумената функције вршити преко показивача.

```
#include <stdio.h>

void DivIMod(int x, int y, int *div, int *mod)
{
    printf("\n Kolicnik postavljem na adresu: %p", div);
    printf("\n Ostatak postavljem na adresu: %p", mod);
    *div = x/y;
    *mod = x%y;
}

main()
{
    int a, b, div, mod;
    printf("\n a= ");
    scanf("%d", &a);
    printf(" b= ");
    scanf("%d", &b);
    printf("\n Adresa promenljive div: %p", &div);
    printf("\n Adresa promenljive mod: %p", &mod);
    DivIMod(5, 2, &div, &mod);
    printf("\n Vrednost promenljive div: %d", div);
    printf("\n Vrednost promenljive mod: %d", mod);
    getche();
    return 0;
}
```



```

C:\Documents and Settings\Bane Jaksic\...
a= 7
b= 3

Adresa promenljive div: 0023FF6C
Adresa promenljive mod: 0023FF68
Kolicnik postavljen na adresu: 0023FF6C
Ostatak postavljen na adresu: 0023FF68
Vrednost promenljive div: 2
Vrednost promenljive mod: 1_

```

Испис на екрану

2.10. Програм учитава текст са улаза. Унос текста се завршава сигналом EOF. Саставити функцију која броји мала и велика слова, цифре, белине и редове, а затим је тестирати у главном програму. Пренос аргумената функције вршити преко показивача.

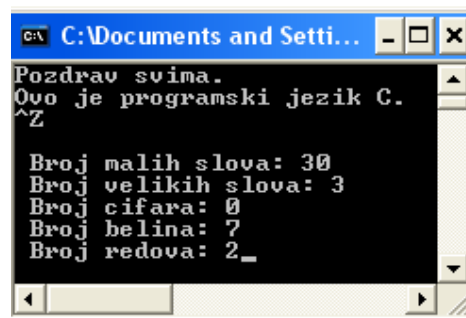
```

#include <stdio.h>

void Prebroj(int *mala, int *velika, int *cifre,
             int *beline, int *redovi)
{
    int c;
    while((c = getchar()) != EOF)
    {
        if(c >= 'a' && c <= 'z') (*mala)++;
        else if(c >= 'A' && c <= 'Z') (*velika)++;
        else if(c >= '0' && c <= '9') (*cifre)++;
        else if(c == '\n' || c == '\t' || c == ' ')
        {
            (*beline)++;
            if(c == '\n') (*redovi)++;
        }
    }
}

main()
{
    int mala=0, velika=0, cifre=0, beline=0, redovi=0;
    Prebroj(&mala, &velika, &cifre, &beline, &redovi);
    printf("\n Broj malih slova: %d", mala);
    printf("\n Broj velikih slova: %d", velika);
    printf("\n Broj cifara: %d", cifre);
    printf("\n Broj belina: %d", beline);
    printf("\n Broj redova: %d", redovi);
    getch();
    return 0;
}

```



```

C:\Documents and Setti...
Pozdrav svima.
Ovo je programski jezik C.
^Z

Broj malih slova: 30
Broj velikih slova: 3
Broj cifara: 0
Broj belina: 7
Broj redova: 2_

```

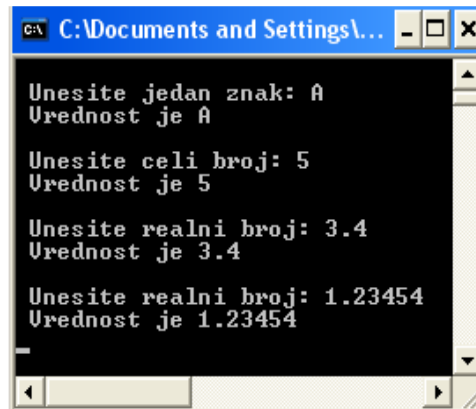
Испис на екрану

2.11. Пример аргумента функције који показивач типа **void**. Саставити функцију помоћу које се може извршити унос података типа **char**, **int**, **float** и **double**, а затим функцију тестирати у главном програму.

```
#include <stdio.h>
#define CHAR 0
#define INT 1
#define FLOAT 2
#define DOUBLE 3

void UnesiVrednost(void *p, int tip)
{
    switch (tip)
    {
        case CHAR:    printf("\n Unesite jedan znak: ");
                      scanf("%c", (char *)p);
                      break;
        case INT:     printf("\n Unesite celi broj: ");
                      scanf("%d", (int *)p);
                      break;
        case FLOAT:   printf("\n Unesite realni broj: ");
                      scanf("%g", (float *)p);
                      break;
        case DOUBLE:  printf("\n Unesite realni broj: ");
                      scanf("%lg", (double *)p);
                      break;
    }
}

main()
{
    int ceo;
    float realni;
    double realnid;
    char znak;
    UnesiVrednost(&znak, CHAR);
    printf(" Vrednost je %c\n" , znak);
    UnesiVrednost(&ceo, INT);
    printf(" Vrednost je %d\n" , ceo);
    UnesiVrednost(&realni, FLOAT);
    printf(" Vrednost je %g\n" , realni);
    UnesiVrednost(&realnid, DOUBLE);
    printf(" Vrednost je %lg\n" , realnid);
    getch();
    return 0;
}
```



Испис на екрану

Уочите како је употребљена функција **scanf()**. Испред имена аргумента није употребљен адресни оператор јер је вредност показивача **p** адреса. Испред аргумената је експлицитно означен тип. Овакав начин употребе **void** показивача је опасан, јер ако се при позиву функције не позову компатибилни аргументи, може доћи до непредвидивих последица.

2.3 Показивачи и низови

2.12. Описати сваку наредбу у следећем блоку наредби:

```
int a[10], *pa, *pb, x, y;
pa=&a[4];
x=*(pa+3);
y=*pa+3;
*pa++;
(*pa)++;
pb=a+2;
*--pa;
--*pa;
```

```
int a[10], *pa, *pb, x, y;
pa=&a[4];      исто као pa=a+4;  pa показује на a[4]
x=*(pa+3);    x=a[7];
y=*pa+3;      y=a[4]+3;
*pa++;        исто као *(pa++); повећава се показивач
(*pa)++;      повећава се показивани податак
pb=a+2;       исто као pb=&a[2]; pb показује на a[2]
*--pa;        смањује се показивач
--*pa;        смањује се показивани податак
```

2.13. Које вредности се исписују на екрану након извршавања следећих програмских кодова:

```
#include <stdio.h>

main()
{
    char s[] = "abcde";
    int t[] = {1, 2, 3, 4, 5};
    /*Inicijalizujmo pokazivace ps i pt na pocetke
    nizova s i t*/
    char *ps = &s[0];
    int *pt = &t[0];
    /*Pokazivace je moguće sabirati sa celim brojevima i
    od njih je moguće oduzimati cele brojeve*/
    /*Ispisimo vrednosti pokazivaca*/
    printf(" ps = %p\n", ps);
    printf(" ps+1 = %p\n", ps+1);
    printf(" ps+2 = %p\n", ps+2);
    printf(" ps-1 = %p\n", ps-1);
    printf(" ps-2 = %p\n", ps-2);
    printf(" pt = %p\n", pt);
    printf(" pt+1 = %p\n", pt+1);
    printf(" pt+2 = %p\n", pt+2);
    printf(" pt-1 = %p\n", pt-1);
    printf(" pt-2 = %p\n\n", pt-2);
    /*Na pokazivace je moguće primenjivati i operatore ++ i --*/
    for(ps=s; *ps; ps++)
        putchar(*ps);
    putchar('\n');
    /*Slicno, dva pokazivaca istog tipa se mogu oduzimati. Prilikom
    odredjivanja rezultata, uzima se u obzir velicina tipa.*/
    ps=&s[3];
    printf("\n s = %p\n", s);
    printf(" ps = %p\n", ps);
    printf(" ps - s = %d\n", ps - s);
    pt=&t[3];
    printf("\n t = %p\n", t);
    printf(" pt = %p\n", pt);
    printf(" pt - t = %d\n", pt - t);
    getch();
    return 0;
}
```

```

C:\Docume...
ps = 0023FF60
ps+1 = 0023FF61
ps+2 = 0023FF62
ps-1 = 0023FF5F
ps-2 = 0023FF5E
pt = 0023FF40
pt+1 = 0023FF44
pt+2 = 0023FF48
pt-1 = 0023FF3C
pt-2 = 0023FF38

abcde

s = 0023FF60
ps = 0023FF63
ps - s = 3

t = 0023FF40
pt = 0023FF4C
pt - t = 3
```

Испис на екрану

2.14. Који је резултат извршавања следећих програмских кодова:

a)

```
#include <stdio.h>

main()
{
    int x[5] = {1,2,3,4,5};
    int *p;
    p=x;
    printf(" %d %d %d\n", x[0], x[1], x[2]);
    printf(" %d %d %d\n", *p, *(p+1), *(p+2));
    getch();
    return 0;
}
```

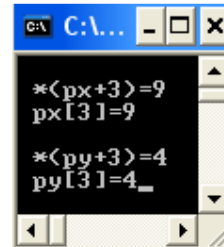


Испис на екрану

б)

```
#include <stdio.h>

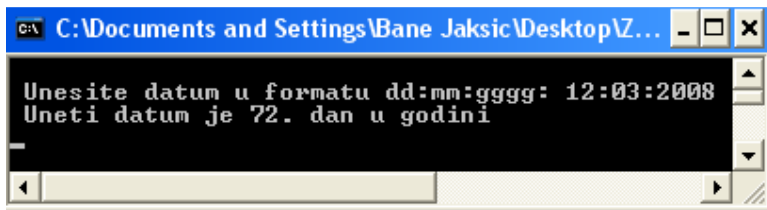
main()
{
    int a[10]={1,2,3,4,5,6,7,8,9,10}, i=3;
    int *px, *py;
    px=a+5;
    printf("\n *(px+%d)=%d", i, *(px+i));
    printf("\n px[%d]=%d \n", i, px[i]);
    py=a;
    printf("\n *(py+%d)=%d", i, *(py+i));
    printf("\n py[%d]=%d", i, py[i]);
    getch();
    return 0;
}
```



Испис на екрану

2.15. Саставити програм који за унети датум исписује редни број тог дана у датој години.

```
#include <stdio.h>
main()
{
    int obicna[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
    int prestupna[] = {31,29,31,30,31,30,31,31,30,31,30,31};
    int dan, mesec, godina, i, *tekuca, danUgodini=0;
    printf("\n Unesite datum u formatu dd:mm:gggg: ");
    scanf("%d:%d:%d", &dan, &mesec, &godina);
    /*Proveravamo da li je godina prestupna*/
    if(godina%400 == 0 || (godina%100 != 0 && godina%4 == 0))
        tekuca = prestupna;
    else
        tekuca = obicna;
    /*Sumiramo dane protekle prethodnih meseci. Indeks ide
    od 0 do mesec-2, zato sto indeksi u C-u pocinju od 0.
    Npr, za dan u mesecu martu (treci mesec) treba sabrati
    broj dana u januaru i februaru (tj. indeks u petlji treba
    ici od 0 do 1, ukljucujuci i jednicu).*/
    for(i=0; i<mesec-1; i++)
        danUgodini += tekuca[i];
    /*Nakon toga jos treba dodati broj proteklih dana u tekucem mesecu*/
    danUgodini += dan;
    printf(" Uneti datum je %d. dan u godini\n", danUgodini);
    getch(); return 0;
}
```

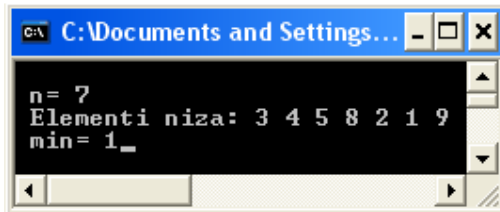


Испис на екрану

2.16. Саставити програм за исписивање вредности најмањег елемента у низу целих бројева дужине **n**. За приступ елементима користити показиваче.

```
#include <stdio.h>
#define MAX 100

main()
{
    int a[MAX], n, min, *p;
    printf("\n n= ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(p=a; p<a+n; p++)
        scanf ("%d", p);
    min=*a;
    for(p=a+1; p<a+n; p++)
        if(*p<min)
            min=*p;
    printf(" min= %d", min);
    getch();
    return 0;
}
```



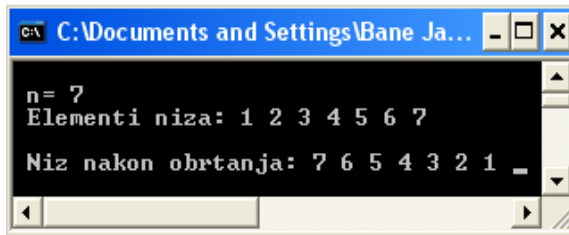
Испис на екрану

2.17. Саставити програм који за унети низ целих бројева дужине **n** врши обртање његових елемената у супротном смеру. Исписати новодобијени низ. За приступ елементима низа користити показиваче.

```
#include <stdio.h>
#define MAX 100

main()
{
    int a[MAX], n, *p, *q, t;
    printf("\n n= ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(p=a; p-a<n; p++)
        scanf("%d", p);
    printf("\n");
    for(p=a, q=a+n-1; p<q; p++, q--)
    {
        t=*p;
        *p=*q;
        *q=t;
    }
}
```

```
printf(" Niz nakon obrtanja: ");
for(p=a; p-a<n; p++)
    printf("%d ", *p);
getche();
return 0;
}
```



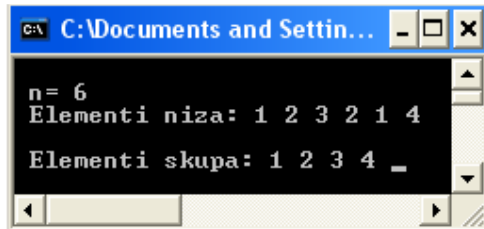
Испис на екрану

2.18. Саставити функцију за претварање низа целих бројева (међу чијим елементима могу да буду и једнаки) у скуп чији су сви елементи међусобно различити. Затим саставити програм који прочита низ бројева дужине **n**, претвори га у скуп и испише добијени резултат. Користити показивач на дужину низа као аргумент функције.

```
#include <stdio.h>
#define MAX 100

void Skup(int a[], int *n)
{
    int i, j=0, k;
    for(i=0; i<*n; i++)
    {
        for(k=0; k<j && a[k]!=a[i]; k++);
        if(k == j)
        {
            a[j] = a[i];
            j++;
        }
    }
    *n = j;
}

main ()
{
    int a[MAX], n, i;
    printf ("\n n= ");
    scanf ("%d", &n);
    printf (" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    if(n==0) printf("\n");
    Skup(a, &n);
    printf ("\n Elementi skupa: ");
    for(i=0; i<n; i++)
        printf ("%d ",a[i]);
    getch();
    return 0;
}
```



Испис на екрану

2.19. Саставити функцију којом се у задатом низу бројева сваки подниз међусобно једнаких бројева сведе на по један примерак тих бројева. На пример, низ 1, 2, 2, 3, 1, 1, 1, 4, 4 треба претворити у низ 1, 2, 3, 1, 4. Затим саставити главни програм који прочита низ бројева дужине **n**, позове претходну функцију и испише добијени резултат. Користити показивач на дужину низа као аргумент функције.

```
#include <stdio.h>
#define MAX 100

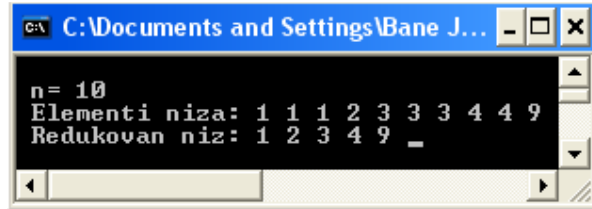
void Redukcija(int a[], int *n)
{
    int i, j=0;
    for(i=1; i<*n; i++)
        if(a[j] != a[i])
        {
            j++;
            a[j]=a[i];
        }
}
```

```

    }
    *n=j+1;
}

main ()
{
    int a[MAX], n, i;
    printf("\n n= ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%d", &a[i]);
    Redukcija(a, &n);
    printf(" Redukovan niz: ");
    for(i=0; i<n; i++)
        printf ("%d ", a[i]);
    getch();
    return 0;
}

```



Испис на екрану

2.4 Показивачи на низове као аргументи функције

2.20. Шта се исписује на екрану након извршавања следећег програмског кода:

```

#include <stdio.h>

void Stampaj(int *pa, int n)
{
    int i;
    for(i = 0; i<n; i++)
        printf(" %d", pa[i]);
}

main ()
{
    int a[]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int n=sizeof(a)/sizeof(int);
    int *pa;

    /*Niz je isto sto i adresa prvog elementa*/
    printf(" Niz a: %p\n", a);
    printf(" Adresa prvog elementa niza a (&a[0]): %p\n", &a[0]);

    /*Moguće je dodeliti niz pokazivacu odgovarajućeg tipa*/
    pa = a;

    printf(" Pokazivac pa ukazuje na adresu: %p\n", pa);

    /*Nizu nije moguće dodeliti pokazivacku promenljivu
    (nizove mozemo smatrati KONSTANTNIM pokazivacima na prvi element)*/

    /*Niz je moguće koristiti kao pokazivac tj. vaze pravila
    pokazivacke aritmetike */
    printf(" a+3 = %p\n", a + 3);
}

```

```

/*Vazi da je a+i=&a[i] odnosno *(a+i)=a[i] */
printf(" &a[3] = %p\n", &a[3]);

/*Identiteti a+i=&a[i] odnosno *(a+i)=a[i]
vazi i za pokazivace i za nizove */

/*Pokazivace je na osnovu prethodnog moguće indeksirati kao nizove */
printf(" pa[5] = %d\n", pa[5]);
printf(" *(pa + 5) = %d\n", *(pa+5));

/*Medjutim, sizeof(pa) je samo velicina pokazivaca, a ne niza*/
printf(" sizeof(a) = %d\n", sizeof(a));
printf(" sizeof(pa) = %d\n", sizeof(pa));

/*Pozivamo funkciju za stampanje niza i saljemo joj niz*/
Stampaj(a, n);
printf("\n");
/*Pozivamo funkciju za stampanje niza i saljemo joj
pokazivac na pocetak niza*/
Stampaj(pa, n);
getche();
return 0;
}

```

Испис на екрану

2.21. Саставити функције за учитавање и испис нiza користећи као аргумент функције показивач на низ. Затим саставити програм који учита два нiza целих бројева исте дужине **n**, одређује који низ има већу суму елемената и исписује онау суму која има већу вредност.

```

#include <stdio.h>
#define MAX 100

void Ucitaj(int *a, int n)
{
    int i;
    for(i=0; i<n; i++)
        scanf("%d", (a+i));
}

void Stampaj(int *a, int n)
{
    int i;
    for(i=0; i<n; i++)
        printf(" %d", *(a+i));
}

```

```

main ()
{
    int a[MAX], b[MAX], s1=0, s2=0;
    int i, j, n;
    printf("\n n= ");
    scanf("%d",&n);
    printf(" Elementi niza A: ");
    Ucitaj(&a[0], n);
    printf(" Elementi niza B: ");
    Ucitaj(&b[0],n);
    printf(" Elementi niza sa vecom sumom: ");
    for(i=0; i<n; i++)
    {
        s1+=a[i];
        s2+=b[i];
    }
    if(s1>s2) Stampaaj(a,n);
    else Stampaaj(b,n);
    getche();
    return 0;
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop...
n= 5
Elementi niza A: 1 1 2 3 1
Elementi niza B: 0 0 0 1 2
Elementi niza sa vecom sumom: 1 1 2 3 1

```

Испис на екрану

2.22. Саставити функције за одређивање суме елемената низа користећи као аргумент функције показивач на низ. Затим саставити програм који учита низ целих бројева дужине **n** и користећи претходну функцију израчунава и исписује суму елемената низа.

```

#include <stdio.h>
#define MAX 100

int Suma(int *a, int n)
{
    int s=0, i;
    for(i=0; i<n; i++)
        s += *(a+i);
    return(s);
}

main ()
{
    int a[MAX], n, i, s;
    printf("\n n= ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%d", &a[i]);
    s=Suma(a, n);
    printf (" suma= %d ", s);
    getche();
    return 0;
}

```

```

C:\Documents and Se...
n= 5
Elementi niza: 1 2 3 2 7
suma= 15

```

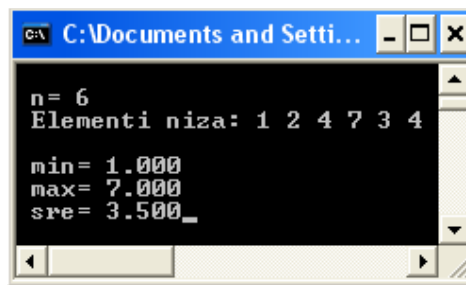
Испис на екрану

2.23. Саставити функције за одређивање минималног и максималног елемента, као и аритметичку средину елемената низа користећи као аргумент функције показивач на низ. Затим саставити програм који учита низ реалних бројева дужине **n** и користећи претходну функцију испишује минимални и максимални елемент, као и аритметичку средину.

```
#include <stdio.h>
#define MAX 100

void MinMaxSre(double *niz, int n, double *pMin,
               double *pMax, double *pSre)
{
    int i;
    double suma=0;
    *pMin=*pMax=niz[0];
    for(i=0; i<n; i++)
    {
        if(niz[i]>*pMax) *pMax=niz[i];
        if(niz[i]<*pMin) *pMin=niz[i];
        suma+=niz[i];
    }
    *pSre=suma/n;
}

main ()
{
    int i, n;
    double min, max, sre, a[MAX];
    printf("\n n= ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%lg", &a[i]);
    MinMaxSre(a, n, &min, &max, &sre);
    printf("\n min= %.3lf\n max= %.3lf", min, max);
    printf("\n sre= %.3lf", sre);
    getch();
    return 0;
}
```



Испис на екрану

2.24. Саставити функцију која методом *Бинарне претраге* проналази позицију траженог елемента у низу користећи као аргумент функције показивач на низ. Претпоставља се да је низ уређен у растућем поретку. Затим саставити програм који за унети низ целих бројева, дужине **n**, проналази и испишује позицију траженог елемента или испишује обавештење да тражени елемент не постоји у низу. Користити:

- а) итеративну верзију функције
- б) рекурзивну верзију функције.

Метода Бинарне претраге: Нека је дат низ **a[0], a[1], ..., a[n-1]** и вредност елемента који се тражи **b**. Најпре се **b** са средњим елементом низа (или елементом око средине). Ако су једнаки, претраживање је завршено. Ако је **b** мање од средњег елемента, тада се претраживање наставља у левој половини низа, а супротно у десну. У изабраној половини се примењује исти алгоритам.

a)

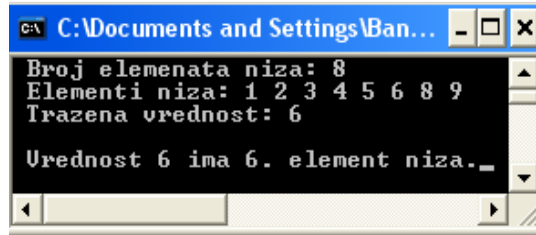
```

#include <stdio.h>
#define MAX 100

int BinPretraga(int *niz, int n, int broj)
{
    int levi=0, desni=n-1, m;
    while(levi <= desni)
    {
        m=(levi+desni)/2;
        if(broj == niz[m])
            return m;
        else if(broj < niz[m])
            desni = m-1;
        else
            levi = m+1;
    }
    return -1;
}

main ()
{
    int i, n, broj, niz[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    printf(" Trazena vrednost: ");
    scanf("%d", &broj);
    i=BinPretraga(niz, n, broj);
    if(i== -1)
        printf("\n Vrednost %d nije nadjena u nizu.", broj);
    else
        printf("\n Vrednost %d ima %d. element niza.", broj, i+1);
    getch();
    return 0;
}

```

*Изпис на екрану*

6)

```

#include <stdio.h>
#define MAX 100

int BinPretragaR(int *niz, int levi, int desni, int broj)
{
    int m;
    if(levi >= desni)
        return -1;
    m=(levi+desni)/2;
    if(broj < niz[m])
        return BinPretragaR(niz, levi, m, broj);
    else if(broj > niz[m])
        return BinPretragaR(niz, m, desni, broj);
    else
        return m;
}

main ()
{
    int i, n, broj, niz[MAX];

```

```

printf(" Broj elemenata niza: ");
scanf("%d", &n);
printf(" Elementi niza: ");
for(i=0; i<n; i++)
    scanf("%d", &niz[i]);
printf(" Trazena vrednost: ");
scanf("%d", &broj);
i=BinPretragaR(niz, 0, n-1, broj);
if(i==-1)
    printf("\n Vrednost %d nije nadjena u nizu.", broj);
else
    printf("\n Vrednost %d ima %d. element niza.", broj, i+1);
getche();
return 0;
}

```

2.25. Саставити функцију за сортирање низа целих бројева дужине **n** у неоппадајући поредак методом мехурића (*Bubble Sort*) користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сортира низ и исписује новодобијени низ.

Bubble Sort: Пролазимо кроз низ редом поредећи суседне елементе, и при том их замењујући ако су у погрешном поретку. Овим се највећи елемент попут мехурића истискује на "површину", тј. на крајњу десну позицију. Након тога је потребно овај поступак поновити над низом **a[0],...,a[n-2]**, тј. над првих **n-1** елемената низа без последњег који је постављен на праву позицију. Након тога се исти поступак понавља над све краћим и краћим префиксима низа, чиме се један по један истискују елементи на своје праве позиције.

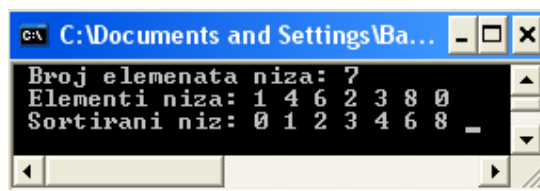
```

#include <stdio.h>
#define MAX 100

void BubbleSort(int *niz, int n)
{
    int i, j, pom;
    for(i=0; i<n; i++)
        for(j=i; j<n; j++)
            if(niz[i] > niz[j])
            {
                pom = niz[i];
                niz[i] = niz[j];
                niz[j] = pom;
            }
}

main ()
{
    int i, j, n, niz[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    BubbleSort(niz, n);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf ("%d ", niz[i]);
    getche();
    return 0;
}

```



Испис на екрану

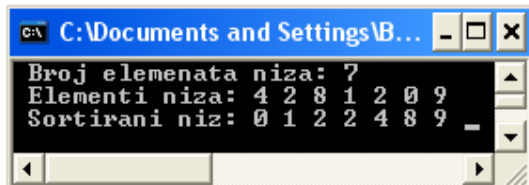
2.26. Саставити функцију за сортирање низа целих бројева дужине **n** у неоппадајући поредак методом уметања (*Insert Sort*) користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сортира низ и исписује новодобијени низ.

Insert Sort: Нека је првих **k** елемената већ уређено у неоппадајућем поретку, тада се узима (**k+1**)-ви елемент и умеће на одговарајуће место међу првих **k** елемената тако да првих **k+1** елемената буде уређено. Овај се метод примењује за **k** од **0** до **n-2**.

```
#include <stdio.h>
#define MAX 100

void InsertSort(int *niz, int n)
{
    int i, j, pom;
    for(i=1; i<n; i++)
    {
        pom=niz[i];
        j=i-1;
        for(j=i-1; j>=0; j--)
            if (niz[j] > pom)
                niz[j+1]=niz[j];
            else break;
        niz[j+1]=pom;
    }
}

main ()
{
    int i, j, n, niz[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    InsertSort(niz, n);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf ("%d ", niz[i]);
    getch();
    return 0;
}
```



Испис на екрану

2.27. Саставити функцију за сортирање низа целих бројева дужине **n** у неоппадајући поредак методом избора (*Selection Sort*) користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сортира низ и исписује новодобијени низ.

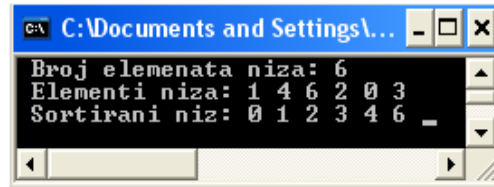
Selection Sort подразумева да минимални елемент низа размени са **a[0]**, минимални елемент одсечка **a[1], a[2], ..., a[n-1]** разменити са **a[1]**, минимални елемент одсечка **a[2], a[3], ..., a[n-1]** разменити са **a[2]**; исти поступак применити на преостале елементе осим последњег који се налази на свом месту.

```
#include <stdio.h>
#define MAX 100

/*Prva verzija funkcije*/
void SelectionSort1(int *niz, int n)
{
    int i, j, najmanji, indexNajmanjeg;
    for(i=0; i<n-1; i++)
    {
        indexNajmanjeg = i;
        najmanji = niz[i];
        for(j=i+1; j<n; j++)
        {
            if(niz[j] < najmanji)
            {
                indexNajmanjeg = j;
                najmanji = niz[j];
            }
        }
        niz[indexNajmanjeg] = niz[i];
        niz[i] = najmanji;
    }
}

/*Druga verzija funkcije*/
void SelectionSort2(int *niz, int n)
{
    int i, j, najmanji, indexNajmanjeg;
    for(i=0; i<n-1; i++)
    {
        indexNajmanjeg = i;
        for(j=i+1; j<n; j++)
            if(niz[j] < niz[indexNajmanjeg])
                indexNajmanjeg = j;
        najmanji = niz[indexNajmanjeg];
        niz[indexNajmanjeg] = niz[i];
        niz[i] = najmanji;
    }
}

main ()
{
    int i, j, n, niz[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    SelectionSort2(niz, n);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf ("%d ", niz[i]);
    getch();
    return 0;
}
```



Испис на екрану

2.28. Саставити рекурзивну функцију за сортирање низа целих бројева дужине **n** у неоппадајући поредак методом *Quick Sort* користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сортира низ и исписује новодобијени низ.

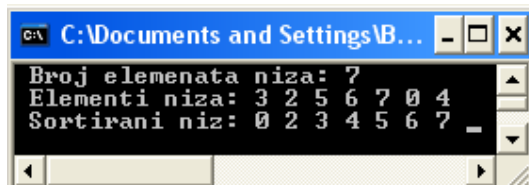
Quick Sort: Прво се цео низ премештањем елемената дели на две групе: леву и десну, тако да су сви елементи леве групе мањи од **d** (**d** је делитељ на групе и може бити произвољан елемент низа, нпр. последњи), а елементи десне групе већи или једнаки **d**. Затим се размењују вредности делитеља и елемената који је на граничној позицији између група (лево од граничне позиције су све вредности мање, а десне веће или једнаке од делитеља). Пошто је делитељ на одговарајућем месту које треба да има у сортираном низу, исти поступак применити на елемент лево и десно од делитеља рекурзивним позивом функције.

```
#include <stdio.h>
#define MAX 100

void Razmeni(int *niz, int i, int j)
{
    int pom;
    pom=niz[i];
    niz[i]=niz[j];
    niz[j]=pom;
}

void QuickSort(int *niz, int levi, int desni)
{
    int i, zadnji;
    if(levi >= desni)
        return;
    Razmeni(niz, levi, (levi+desni)/2);
    zadnji=levi;
    for(i=levi+1; i<=desni; i++)
        if(niz[i] < niz[levi])
            Razmeni(niz, ++zadnji, i);
    Razmeni(niz, levi, zadnji);
    QuickSort(niz, levi, zadnji-1);
    QuickSort(niz, zadnji+1, desni);
}

main ()
{
    int i, j, n, niz[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    QuickSort(niz, 0, n);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf ("%d ", niz[i]);
    getch();
    return 0;
}
```



Испис на екрану

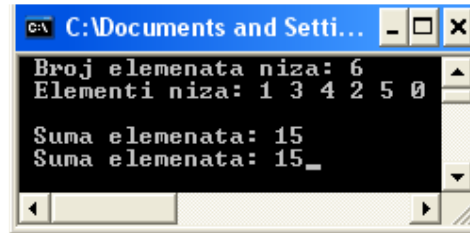
2.29. Саставити рекурзивну функцију за сабирање елемената низ целих бројева користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сабира елементе и исписује њихову суму.

```
#include <stdio.h>
#define MAX 100

/*Prva verzija funkcije*/
int sumaNiza1(int *a, int n)
{
    if(n<=0)
        return 0;
    return (a[n-1]+sumaNiza1(a,n-1));
}

/*Druga verzija funkcije*/
int sumaNiza2(int *a, int n)
{
    if(n<=0)
        return 0;
    return (a[0]+sumaNiza2(a+1,n-1));
}

main()
{
    int i, n, a[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%d", &a[i]);
    printf("\n Suma elemenata: %d",sumaNiza1(a, n));
    printf("\n Suma elemenata: %d",sumaNiza2(a, n));
    getch();
    return 0;
}
```



Испис на екрану

Урађене су две верзије рекурзивних функција у зависности од тога да ли се низ **a[0],a[1],...,a[n-1]** посматра као:

- **a[0]** и остатак низа или као
- почетни део низа + **a[n-1]**.

2.30. Саставити рекурзивну функцију за налажење елемента максималне вредности из низа целих бројева користећи као аргумент функције показивач на низ. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције исписује највећи елемент.

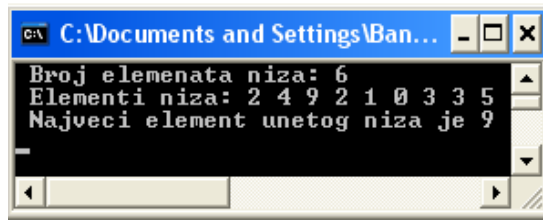
```
#include <stdio.h>
#define MAX 100

int Maksimum(int *a, int n)
{
    int m;
    /*m nam prihvata povratnu vrednost rekurzivnog poziva,
    da ne bismo imali dva rekurzivna poziva.*/
    /*Pre poziva funkcije treba proveriti da li je n>0
    inace nema smisla traziti maksimum.*/
    if(n==1)
        return a[0];
    return a[n-1] > (m=Maksimum(a,n-1)) ? a[n-1]: m;
}
```

```

main()
{
    int i, n, a[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%d", &a[i]);
    printf(" Najveci element unetog niza je %d\n", Maksimum(a, n));
    getch();
    return 0;
}

```



Испис на екрану

2.31. Саставити рекурзивну функцију која израчунава скаларни производ два дата вектора целих бројева користећи као аргумент функције показивач на низ. Саставити и функцију којом се учитавају димензије и координате вектора. Затим саставити програм који употребом претходних функција учитава два вектора и исписује њихов скаларни производ.

```

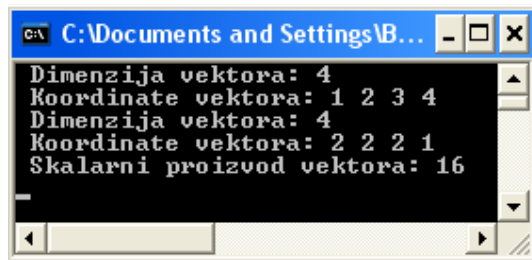
#include <stdio.h>
#define MAX 100

int SkalarniProizvod(int *a, int *b, int n)
{
    if(n<=0) return 0;
    return a[n-1] * b[n-1] + SkalarniProizvod(a, b, n-1);
}

void UcitajVektor(int a[], int *n)
{
    int i;
    printf(" Dimenzija vektora: ");
    scanf("%d", n);
    if(*n<=0 || *n>MAX)
        printf("Vektor mora imati vise od 0 i manje od %d koordinata.\n", MAX);
    printf(" Koordinate vektora: ");
    for(i=0; i<*n; i++)
        scanf("%d",&a[i]);
}

main()
{
    int a[MAX], b[MAX], n, m ;
    UcitajVektor(a,&n);
    UcitajVektor(b,&m);
    if(n != m)
        printf(" Vektori nisu istih dimenzija!\n");
    printf(" Skalarni proizvod vektora: %d\n", SkalarniProizvod(a, b, n));
    getch();
    return 0;
}

```



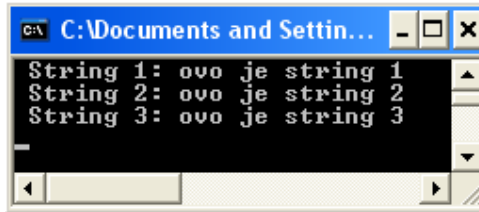
Испис на екрану

2.5 Показивачи и низови

2.32. Који је резултат извршавања следећег програмског кода:

```
#include <stdio.h>

main()
{
    char s1[] = {'o', 'v', 'o', ' ', 'j', 'e', ' ', 's',
                't', 'r', 'i', 'n', 'g', ' ', 'l', '\0'};
    char s2[] = "ovo je string 2";
    char *s3 = "ovo je string 3";
    int i;
    char *p;
    /*Ispis prvog niza karaktera*/
    printf(" String 1: ");
    i=0;
    while(s1[i] != '\0')
        putchar(s1[i++]);
    putchar('\n');
    /* Ispis drugog niza */
    printf(" String 2: ");
    p=s2;
    while(*p != '\0')
        putchar (*p++);
    putchar ('\n');
    /* Ispis treceg niza */
    printf(" String 3: ");
    while(*s3 != '\0')
        putchar (*s3++);
    putchar('\n');
    getche(); return 0;
}
```

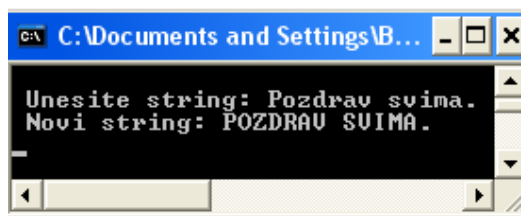


Испис на екрану

2.33. Саставити програм који учитава стринг, а затим свако мало слово у унетом стрингу замењује великим словом. Приступ елементима стринга остварити показивачима.

```
#include <stdio.h>
#define MAX 100

main()
{
    char s[MAX], *p;
    printf("\n Unesite string: ");
    gets(s);
    p=s;
    while(*p)
    {
        if(*p >='a' && *p <='z')
            *p = *p - ('a' - 'A');
        p++;
    }
    printf(" Novi string: ");
    puts(s);
    getche(); return 0;
}
```

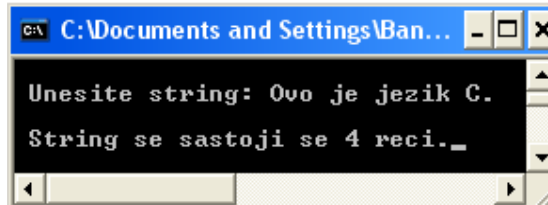


Испис на екрану

2.34. Саставити програм који учитава стринг, а затим исписује број речи у том стрингу. Приступ елементима стринга остварити показивачима.

```
#include <stdio.h>
#define MAX 100

main()
{
    char s[MAX], *p;
    int br=1;
    printf("\n Unesite string: ");
    gets(s);
    p=s;
    while(*p==' ')
        p++;
    while(*p)
    {
        if(*p==' ')
        {
            while(*p==' ')
                p++;
            br++;
        }
        p++;
    }
    printf("\n String se sastoji se %d reci.", br);
    getch();
    return 0;
}
```

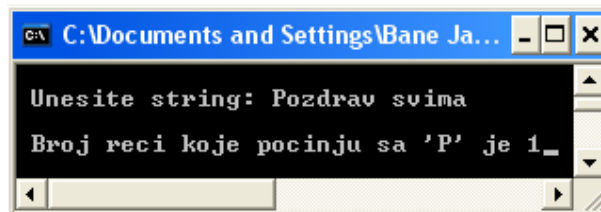


Испис на екрану

2.35. Саставити програм који учитава стринг, а затим исписује број речи у том стрингу које почињу са "P". Приступ елементима стринга остварити показивачима.

```
#include <stdio.h>
#define MAX 100

main()
{
    char s[MAX], *p;
    int br=0;
    printf("\n Unesite string: ");
    gets(s);
    p=s;
    while(*p==' ')
        p++;
    if(*p=='P')
        br++;
    while(*p)
    {
        if(*s==' ')
        {
            while(*p==' ')
                p++;
            if(*p=='B')
                br++;
        }
        p++;
    }
}
```



Испис на екрану

```

printf("\n Broj reci koje pocinju sa 'B' je %d", br);
getche();
return 0;
}

```

2.36. Саставити функцију која:

- а) израчунава дужину стринга;
- б) копира стринг **s2** у стринг **s1**, претпоставља да у **s1** има довољно простора;
- в) надовезује стринг **s2** на крај стринга **s1**, претпоставља да у **s1** има довољно простора
- г) врши лексикографско поређење два стринга, враћа: 0 - уколико су стрингови једнаки, <0 - уколико је **s1** лексикографски испред **s2**, >0 - уколико је **s1** лексикографски иза **s2**;
- д) проналази прву позицију карактера **c** у стрингу **s**, враћа позицију на којој је **c**, односно NULL уколико **s** не садржи **c**;
- ђ) проналази последњу позицију карактера **c** у стрингу **s**, враћа позицију на којој је **c**, односно NULL уколико **s** не садржи **c**;
- е) проверава да ли стринг **s1** садржи стринг **s2**, враћа позицију на којој **s2** почиње, односно NULL уколико га нема.

Приликом формирања функција користити показиваче. Тестирати формиране функције.

```

#include <stdio.h>
#include <stdlib.h> /*zbog NULL*/
#define MAX 100

/*Izracunava duzinu stringa*/
int Duzina(char *s)
{
    char *t;
    for(t=s; *t; t++)
        ;
    return t - s;
}

/*Kopira string s2 u string s1. Pretpostavlja da u s1 ima dovoljno prostora.*/
void Kopiraj(char *s1, char *s2)
{
    /*Kopira karakter po karakter, sve dok nije iskopiran karakter '\0'*/
    while(*s1++ = *s2++)
        ;
}

/*Nadovezuje string s2 na kraj stringa s1.
Pretpostavlja da u s1 ima dovoljno prostora.*/
void Nadovezi(char *s1, char *s2)
{
    /*Pronalazimo kraj stringa s*/
    while(*s1)
        s1++;
    /*Vrsi se kopiranje, slicno funkciji Kopiraj*/
    while(*s1++ = *s2++)
        ;
}

/* Vrsi leksikografsko poredjenje dva stringa.
Vraca :
0 - ukoliko su stringovi jednaki
<0 - ukoliko je s1 leksikografski ispred s2
>0 - ukoliko je s1 leksikografski iza s2*/
int Poredi(char *s1, char *s2)

```

```

{
    /*Petlja tece sve dok ne naidjemo na prvi razliciti karakter*/
    for(; *s1==*s2; s1++, s2++)
        if(*s1 == '\0') /*Naisli smo na kraj oba stringa,a nismo nasli razliku*/
            return 0;
    /* *s1 i *s2 su prvi karakteri u kojima se niske razlikuju.
       Na osnovu njihovog odnosa, odredjuje se odnos stringova*/
    return *s1 - *s2;
}

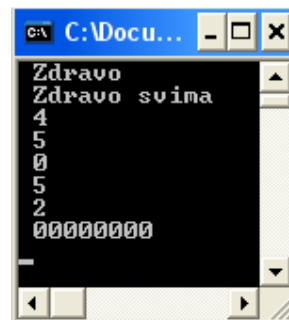
/*Pronalazi prvu poziciju karaktera c u stringu s,
i vraca pokazivac na nju, odnosno NULL ukoliko s ne sadrzi c */
char *PrvaPozicija(char *s, char c)
{
    for(; *s; s++)
        if(*s == c)
            return s;
    /*Nije nadjeno*/
    return NULL;
}

/*Pronalazi poslednju poziciju karaktera c u stringu s.
Vraca poziciju na kojoj je c, odnosno NULL ukoliko s ne sadrzi c*/
char *PoslednjaPozicija(char *s, char c)
{
    char *t=s;
    /*Pronalazimo kraj stringa s*/
    while(*t++)
        ;
    /*Krecemo od kraja i trazimo c unazad*/
    for(t--; t >= s; t--)
        if(*t == c)
            return t;
    /*Nije nadjeno*/
    return NULL;
}

/*Proverava da li string s1 sadrzi string s2.
Vraca poziciju na kojoj s2 pocinje, odnosno NULL ukoliko ga nema*/
char *StringUString(char *s1, char *s2)
{
    char *s, *t;
    /*Proveravamo da li s2 pocinje na svakoj poziciji i */
    for(; *s1; s1++)
        /*Poredimo s2 sa s1 pocevsi od poziciji i sve dok ne naidjemo na razliku*/
        for(s=s1, t=s2; *s == *t; s++, t++)
            /*Nismo naisli na razliku a ispitali smo sve karaktere niske s2 */
            if(*(t+1) == '\0')
                return s1;
    /*Nije nadjeno*/
    return NULL;
}

main()
{
    char s[MAX];
    char t[] = "Zdravo";
    char u[] = " svima";
    char r[] = "racunari";
    Kopiraj(s, t);
    printf(" %s\n", s);
    Nadovezi(s, u);
    printf(" %s\n", s);
}

```



Испис на екрану

```
printf(" %d\n", PrvaPozicija(r, 'n') - r);
printf(" %d\n", PoslednjaPozicija(r, 'a') - r);
printf(" %d\n", StringUString(r, "rac") - r);
printf(" %d\n", StringUString(r, "ari") - r);
printf(" %d\n", StringUString(r, "cun") - r);
printf(" %p\n", StringUString(r, "cna"));
getche();
return 0;
}
```

2.37. Саставити функцију која сортира учитани низ стрингове у алфабетском редоследу. Затим саставити програм који сортира низ учитаних стрингова помоћу претходне функције и исписује сортирани низ стрингова као и дужину сваког стринга.

```
#include <stdio.h>
#include <string.h>
#define DUZINA 100
#define MAX 50
#define STOP ""

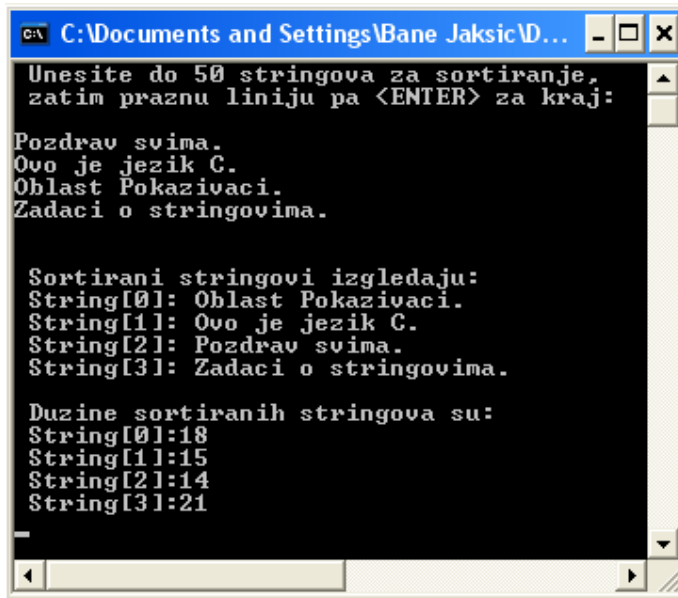
void Sortiraj(char *st[], int d[], int n)
{
    char *pom;
    int i, j, m;
    for(i=0; i<n-1; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(strcmp(st[i], st[j])>0)
            {
                /*zamena mesta*/
                pom=st[i];
                st[i]=st[j];
                st[j]=pom;
                /*zamena duzine*/
                m=d[i];
                d[i]=d[j];
                d[j]=m;
            }
        }
    }
}

main()
{
    char str[MAX][DUZINA];
    char *pokstr[MAX]; /*niz pokazivaca*/
    int duz[MAX];
    int i=0, j;
    printf(" Unesite do %d stringova za sortiranje,", MAX);
    printf("\n zatim praznu liniju pa <ENTER> za kraj:\n\n");
    while(gets(str[i])!= NULL && strcmp(str[i], STOP)!=0 && i<MAX)
    {
        pokstr[i] = str[i];
        duz[i] = strlen(str[i]);
        i++;
    }
    Sortiraj(pokstr, duz, i);
    printf("\n Sortirani stringovi izgledaju:\n");
}
```

```

for(j=0;j<i; j++)
{
    printf(" String[%d]: ",j);
    puts(pokstr[j]);
}
printf("\n Duzine sortiranih stringova su:\n");
for(j=0;j<i; j++)
{
    printf(" String[%d]:",j);
    printf("%d\n", duz[j]);
}
getche();
return 0;
}

```



Испис на екрану

2.38. Саставити функцију која из датог стринга издваја само слова. Функција враћа показивач на новокреирани низ. Затим саставити програм који за унети стринг користећи претходну функцију формира нови стринг само од слова. Исписати нови стринг.

```

#include <stdio.h>
#define MAX 100

char *SamoSlova(char *slpok)
{
    static char a[MAX],*p;
    p=a; /*p pokazuje na prvi clan novog niza*/
    while(*slpok)
    {
        if((*slpok>='a' && *slpok<='z') || (*slpok>='A' && *slpok<='Z'))
        {
            *p = *slpok;
            p++;
        }
        slpok++;
    }
}

```

```

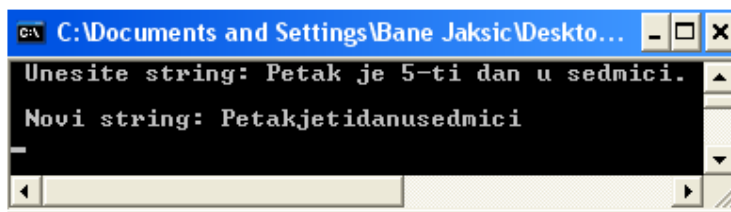
    }
    *p = '\0';
    return a;
}

```

```

main()
{
    char s1[MAX];
    char *s2;
    printf(" Unesite string: ");
    gets(s1);
    s2=SamoSlova(s1);
    printf ("\n Novi string: ");
    puts(s2);
    getch();
    return 0;
}

```



Испис на екрану

2.39. Саставити функцију која утврђује да ли су две речи анаграми, а затим саставити програм који испишује обавештење да ли су две речи анаграми. За две речи кажемо да су анаграми ако се од једне речи може добити друга премештањем слова у речи.

```

#include <stdio.h>
#include <string.h>
#define MAX 100

/*Funkcija sortira karaktere stringa s*/
void Sortiraj(char s[])
{
    int i, j, min, n;
    char pom;
    for(n=0; s[n]!='\0'; n++);
    for(i=0; i<n-1; i++)
    {

```

```

        min=i;
        for(j=i+1; j<n; j++)
            if(s[j] < s[min])
                min = j;
        if(min != i)
        {
            pom = s[i];
            s[i] = s[min];
            s[min] = pom;
        }
    }
}

```

```

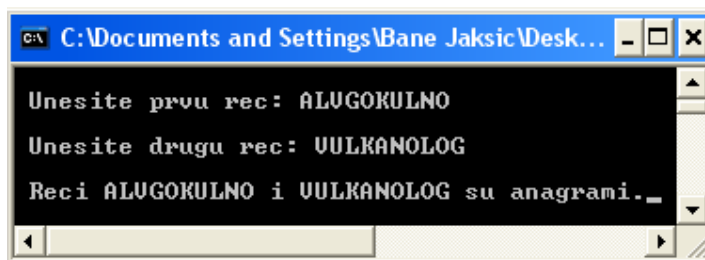
/* Funkcija utvrdjuje da li su reci s i t anagrami.*/

```

```

int Anagrami(char *s, char *t)
{
    char sp[MAX], tp[MAX];
    /*Kopiramo stringove (reci) u pomocne nizove sp i tp zato sto
    ne zelimo da nasa funkcija promeni originalne nizove.*/
    strcpy(sp, s);
    strcpy(tp, t);
    /*Sortiramo karaktere stringova u pomocnim nizovima*/
    Sortiraj(sp);

```



Испис на екрану

```

Sortiraj(tp);
/*Ako su stringovi nakon sortiranja jednaki,
tada su polazne reci bile anagrami */
return (strcmp(sp, tp) == 0);
}

main ()
{
    char s[MAX], t[MAX];
    printf("\n Unesite prvu rec: ");
    scanf("%s", s);
    printf("\n Unesite drugu rec: ");
    scanf("%s", t);
    if(Anagrami(s, t))
        printf("\n Rec i %s i %s su anagrami.", s, t);
    else
        printf("\n Rec i %s i %s nisu anagrami.", s, t);
    getche();
    return 0;
}

```

2.40. Саставити рекурзивну функцију која испитује да ли је дата ниска палиндром. Ниска је палиндром ако се исто чита као од почетка ка крају и од краја ка почетку

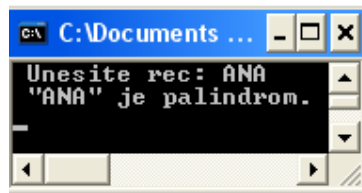
```

#include <stdio.h>
#include <string.h>
#define MAX 20

int Palindrom(char *a, int n)
{
    if( n==0 || n==1)
        return 1; /*Prazna rec ili rec od samo jednog karaktera jeste palindrom.*/
    return a[0]== a[n-1] ? Palindrom(a+1,n-2) : 0;
}

main()
{
    char rec[MAX];
    printf(" Unesite rec: ");
    scanf("%s", rec);
    printf(" \"%s\" je palindrom.\n",
        rec, Palindrom(rec,strlen(rec)) ? "" : "ni" );
    getche();
    return 0;
}

```



Испис на екрану

2.6 Показивачи на функције

2.41. Шта се исписује на екрану након извршавања следећег програмског кода:

```
#include <stdio.h>

int Kvadrat(int n)
{
    return (n*n);
}

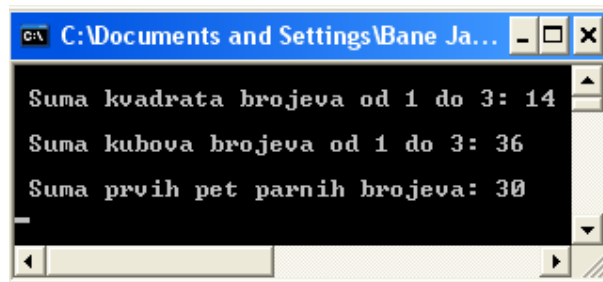
int Kub(int n)
{
    return (n*n*n);
}

int ParniBroj(int n)
{
    return (2*n);
}

int Sumiraj(int(*f)(int), int n)
{
    int i, suma=0;
    for(i=1; i<=n; i++)
        suma += (*f)(i);
    return suma;
}

main()
{
    printf("\n Suma kvadrata brojeva od 1 do 3: %d\n", Sumiraj(Kvadrat,3));
    printf("\n Suma kubova brojeva od 1 do 3: %d\n", Sumiraj(Kub,3));
    printf("\n Suma prvih pet parnih brojeva: %d\n", Sumiraj(ParniBroj,5));
    getch();
    return 0;
}
```

Функција Sumiraj израчунава суму бројева $f(i)$ за i из интервала $[1, n]$.
int (*f)(int) у аргументу функције Sumiraj је показивач на функцију са именом f , која као аргумент прима променљиву типа **int** и враћа као резултат вредност типа **int**.
 У позиву функције Sumiraj, имена функција Kvadrat, Kub и ParniBroj су заправо истовремено и адресе функција, па оператор & није неопходан али није грешка уколико се се оператор & ипак користи испред имена функције.



Испис на екрану

2.42. Саставити програм који за унети реални број рачуна и исписује вредност једне од функција: 1-квадрат, 2-синус или 3-косинус. Корисник бира једну од понуђених функција. Користити показиваче на функције.

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

double Kvadrat(double x)
{
    return x*x;
}

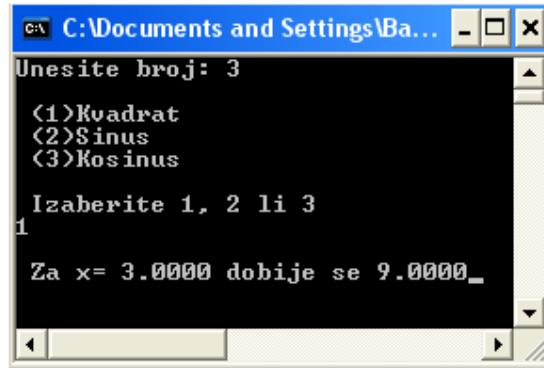
void Stampaj(double(*pF)(), double x)
{
    printf("\n Za x= %.4lf dobije se %.4lf", x, pF(x));
}
```

```

}

main()
{
    double n;
    int izbor;
    double (*pF)(double);
    printf("Unesite broj: ");
    scanf("%lf", &n);
    fflush(stdin); /*odstrani višak znakova s ulaza*/
    printf("\n (1)Kvadrat\n (2)Sinus\n (3)Kosinus\n");
    printf("\n Izaberite 1, 2 li 3\n");
    izbor = getchar();
    switch(izbor)
    {
        case '1': pF=Kvadrat; break;
        case '2': pF=sin; break;
        case '3': pF=cos; break;
        default: return 0;
    }
    Stampaj(pF, n);
    getche();
    return 0;
}

```



Испис на екрану

2.43. Саставити програм који за унети реални број рачуна и исписује вредност једне од функција: 1-квадрат, 2-синус или 3-косинус. Корисник бира једну од понуђених функција. Користити низ показивача на функције.

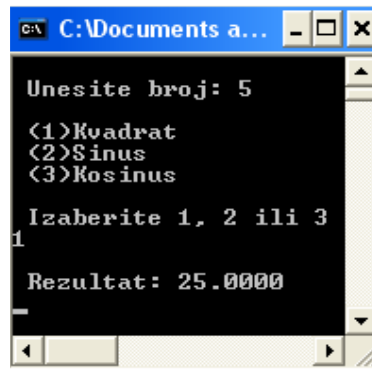
```

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

double Kvadrat(double x)
{
    return x*x;
}

main()
{
    double n;
    int izbor;
    double (*pF[3])(double)={Kvadrat, sin, cos};
    printf("\n Unesite broj: ");
    scanf("%lf", &n);
    fflush(stdin); /*odstrani višak znakova s ulaza*/
    printf("\n (1)Kvadrat\n (2)Sinus\n (3)Kosinus \n");
    printf("\n Izaberite 1, 2 ili 3\n");
    scanf("%d", &izbor);
    if(izbor >=1 && izbor <=3)
        printf("\n Rezultat: %.4lf\n", (*pF[izbor-1])(n));
    getche();
    return 0;
}

```



Испис на екрану

2.44. Саставити функцију која табелира вредности за функције синус, косинус и експонент користећи показивач на функцију. Са тастатуре се уносе границе интервала и корак за табелирање. Затим саставити програм који тестира претходну функцију и исписује вредности.

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

void Tabeliraj(double a, double b, double h, double(*f)(double))
{
    double x;
    printf(" -----\n");
    for(x=a; x<=b; x+=h)
        printf(" | %8.3f | %8.3f |\n", x, (*f)(x));
    printf(" -----\n");
}

main()
{
    double a, b, h;
    printf("\n Unesite granice intervala: ");
    scanf("%lf%lf", &a, &b);
    printf(" Unesite korak: ");
    scanf("%lf", &h);
    printf("\n sin(x)\n");
    Tabeliraj(a, b, h, sin);
    printf(" cos(x)\n");
    Tabeliraj(a, b, h, cos);
    printf(" exp(x)\n");
    Tabeliraj(a, b, h, exp);
    getch();
    return 0;
}
```

Функција `Tabeliraj()` прихвата границе интервала `a` и `b`, корак `h`, као и показивач `f` који показује на функцију која прихвата **double** аргумент, и враћа **double** резултат. За тако дату функцију исписује њене вредности у интервалу `[a,b]` са кораком `h`.

```
C:\Documents and Settings\B...
Unesite granice intervala: 2 3
Unesite korak: 0.2

sin(x)
| 2.000 | 0.909 |
| 2.200 | 0.808 |
| 2.400 | 0.675 |
| 2.600 | 0.516 |
| 2.800 | 0.335 |

cos(x)
| 2.000 | -0.416 |
| 2.200 | -0.589 |
| 2.400 | -0.737 |
| 2.600 | -0.857 |
| 2.800 | -0.942 |

exp(x)
| 2.000 | 7.389 |
| 2.200 | 9.025 |
| 2.400 | 11.023 |
| 2.600 | 13.464 |
| 2.800 | 16.445 |
```

Испис на екрану

2.45. Саставити функцију која табелира вредности реалне функције пригушених осцилација $f(x)=e^{-0.1x}\sin x$ користећи показивач на функцију. Са тастатуре се уносе границе интервала и корак за табелирање. Затим саставити програм који тестира претходну функцију и исписује вредности.

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

void Tabeliraj(double a, double b, double h, double (*f)(double))
{
    double x;
    printf ("\n          x          f(x)\n");
    printf (" ----- \n");
    for(x=a; x<=b; x+=h)
        printf (" | %8.3f | %8.3f | \n", x, (*f)(x));
    printf (" ----- \n");
}

double Oscil(double x)
{
    return(exp(-0.1*x)*sin(x));
}

main()
{
    double a, b, h;
    printf ("\n Unesite granice intervala: ");
    scanf("%lf%lf", &a, &b);
    printf (" Unesite korak: ");
    scanf("%lf", &h);
    Tabeliraj(a, b, h, Oscil);
    getch();
    return 0;
}
```

```
C:\Documents and Settings\B...
Unesite granice intervala: 1 5
Unesite korak: 0.5

      x          f(x)
-----
|  1.000 |  0.761 |
|  1.500 |  0.859 |
|  2.000 |  0.744 |
|  2.500 |  0.466 |
|  3.000 |  0.105 |
|  3.500 | -0.247 |
|  4.000 | -0.507 |
|  4.500 | -0.623 |
|  5.000 | -0.582 |
-----
```

Испис на екрану

2.46. Саставити функцију која приближно израчунава интеграл функције **f(x)** на интервалу **[a,b]**. Функција **f** се прослеђује као параметар, а интеграл се процењује по Симпсоновој формули. Затим у главном програму за унети интервал и број тачака **n**, употребом формиране функције одредити интеграл косинусне функције и исписати добијену вредност.

Симпсонова формула: $\int_a^b f(x)dx = \frac{h}{3} (f(a) + f(b) + 4s_1 + 4s_2),$

где је $h = \frac{b-a}{n}, \quad s_1 = \sum_{i=1}^n f(a + (2i-1)*h), \quad s_2 = \sum_{i=1}^n f(a + 2i*h),$

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

double Simpson(double a, double b, double (*f)(double), int n)
{
    double s1=0, s2=0, h;
    int i;
    h=(b-a)/n;
    for(i=1; i<=n; i++)
    {
        s1 += f(a+(2*i-1)*h);
        s2 += f(a+2*i*h);
    }
    return h/3 * (f(a)+f(b)+4*s1+2*s2);
}

main()
{
    double a, b;
    int n;
    printf(" Unesite granice intervala na kom se funkcija integri: ");
    scanf("%lf %lf", &a,&b);
    printf(" Unesite broj tacaka za Simpsonovu formulu: ");
    scanf("%d",&n);
    printf(" Integral funkcije cos() na [%.2f, %.2f] je %.2f\n",
           a, b, Simpson(a, b, cos,n));
    getch();
    return 0;
}
```

```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak14.exe
Unesite granice intervala na kom se funkcija integri: 1 5
Unesite broj tacaka za Simpsonovu formulu: 6
Integral funkcije cos() na [1.00, 5.00] je -0.57
```

Испис на екрану

2.7 Полиморфне функције

(функције које се могу прилагодити различитим типовима аргумената)

Табела 2.2: Полиморфне функције дефинисане у библиотеци <stdlib.h>

| функција | значење |
|-----------|---|
| qsort() | <p>Функција qsort() врши сортирање низа методом сортирања раздвајањем. Декларација ове функције је следећа:</p> <pre>void qsort(void *b, int n, int s, int (*comp)(const void *, const void *));</pre> <p>Први аргумент је адреса почетка низа који се сортира. С обзиром да се не зна тачан тип елемената низа, користи се генерички показивач (void *, погледати напомену доле). Други аргумент је број елемената низа, а трећи величина сваког од елемената низа. Последњи аргумент је показивач на функцију поређења. Ова функција треба да прихвата адресе елемената низа који се пореде, и да враћа >0 ако је први елемент већи, <0 ако је први елемент мањи, а враћа 0 ако су елементи који се пореде једнаки. На овај начин се може сортирати било који низ, довољно је да је на неки начин функцијом поређења дефинисан потпуни поредак међу елементима низа. Аргументи функције поређења су такође генерички показивачи, опет зато што не знамо тачно ког су типа елементи низа. Ови показивачи су још квалификовани кључном речју const.</p> |
| bsearch() | <p>Функција bsearch() врши претраживање сортираног низа методом бинарне претраге. Функција има следећу декларацију:</p> <pre>void *bsearch(const void *x, const void *b, int n, int s, int (*comp)(const void *, const void *));</pre> <p>Први аргумент је показивач на податак који се тражи у низу. Други аргумент је адреса почетка низа, трећи величина низа, а четврти величина елемента низа. Последњи аргумент је показивач на функцију поређења која дефинише поредак у складу са којим је сортиран низ. Функција враћа адресу елемента у низу који је једнак тразеном елементу, или NULL уколико елемент није пронађен.</p> |

*НАПОМЕНА: Показивач на **void** је показивач који може садржати адресу било ког податка у меморији. Сваки показивач се може без конверзије доделити овом показивачу, као и обрнуто. Може се користити за чување адресе податка за који унапред не знамо ког ће бити типа. Његова главна карактеристика је да се не може дереференцирати, зато што се не зна ког је типа оно на шта он показује. Програмер мора на други начин да утврди ког је типа податак на тој адреси, па да најпре конвертује **void** показивач у одговарајући тип показивача, а затим да га тако конвертованог дереференцира.

2.47. Саставити програм који употребом функције **qsort()** врши сортирање унетог низа реалних бројева. Исписати сортирани низ.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

/*Funkcija poredi dva broja po velicini.*/
/*const znaci da ono na sta pokazuje a (odnosno b) nece biti menjano u funkciji.*/
int Poredi(const void* a, const void* b)
{
    float br_a = *(float*)a;
    float br_b = *(float*)b;
    if(br_a > br_b) return 1;
}
```

```

    else if(br_a < br_b) return -1;
    else return 0;
}

main()
{
    int i, n;
    float a[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%f", &a[i]);
    /*Pocetak niza, broj elemenata, velicina jednog elementa,
    adresa funkcije poredjenja */
    qsort((void*)a, n, sizeof(float), Poredi);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf("%.2f ", a[i]);
    getch();
    return 0;
}

```



Испис на екрану

2.48. Саставити програм који за дати низ целих бројева употребом функције **qsort()** сортира тако да прво иду негативни бројеви (у произвољном редоследу), а затим нуле и на крају позитивни бројеви. Исписати сортирани низ.

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 100

/*Funkcija poredi dva cela broja i vraca:
-1 ako je prvi negativan a drugi 0 ili je >0
0 ako su istog znaka ili oba 0 i
1 ako je prvi pozitivan drugi negativan ili 0*/
int Poredi(const void* m, const void* n)
{
    int a=*(int*)m;
    int b=*(int*)n;
    if(a<0)
    {
        if(b>=0) return -1;
        else return 0;
    }
    if(a==0)
    {
        if(b>0) return -1;
        if(b==0) return 0;
        if(b<0) return 1;
    }
    if(b<=0) return 1;
    else return 0;
}

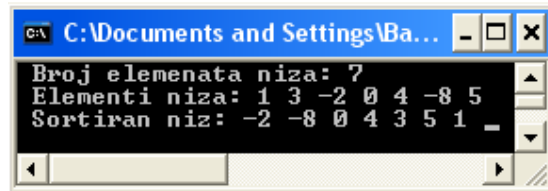
main()
{
    int a[MAX], i, n;

```

```

printf(" Broj elemenata niza: ");
scanf("%d",&n);
if(n<0 || n>MAX)
    printf(" Broj elemenata niza mora biti u opsegu od 0 do %d!\n", MAX);
printf(" Elementi niza: ");
for(i=0; i<n; i++)
    scanf("%d", &a[i]);
qsort(a, n , sizeof(int), &Poredi);
printf(" Sortiran niz: ");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
getche();
return 0;
}

```



Испис на екрану

2.49. Саставити програм који сортира лексикографски и дужински низ речи употребом функције `qsort()`.

```

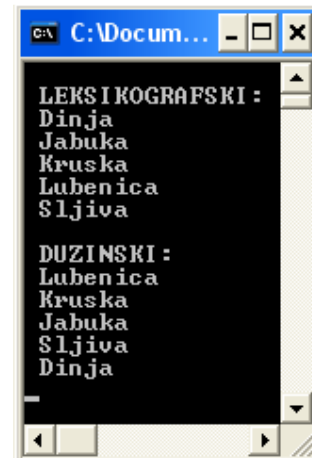
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*Funkcija vrshi leksikografsko poredjenje dve reci. Vraca kao rezultat
>0 ukoliko je prva rec veca,
0 ako su jednake i
<0 ako je prva rec manja.
Sortiranje ce biti u rastucem poretku.*/
int Leksikografski(const void* a, const void* b)
{
    return strcmp(*(char**)a,*(char**)b);
}

/*Funkcija koja poredjenje po duzini dve reci.
Sortiranje koje koristi ovu funkciju ce biti opadajuce!*/
int Duzinski(const void* a, const void* b)
{
    return strlen(*(char**)b)-strlen(*(char**)a);
}

main()
{
    int i;
    char* a[] = {"Jabuka", "Kruska", "Sljiva", "Dinja", "Lubenica"};
    int n = sizeof(a)/sizeof(int);
    qsort((void*)a, n, sizeof(char*), &Leksikografski);
    printf("\n LEKSIKOGRAFSKI:\n");
    for(i=0; i<n; i++)
        printf(" %s\n", a[i]);
    printf("\n DUZINSKI:\n");
    qsort((void*)a, n, sizeof(char*), &Duzinski);
    for(i=0; i<n; i++)
        printf(" %s\n", a[i]);
    getche();
    return 0;
}

```



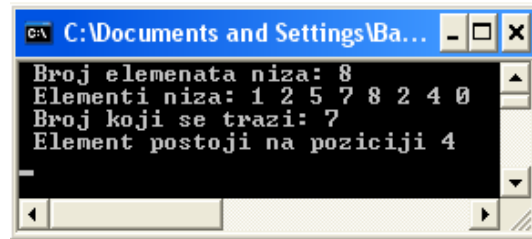
Испис на екрану

2.50. Саставити програм који за унети низ целих бројева дужине **n**, исписује позицију задатог броја **x** уз употребу уграђене функције **bsearch()**. Уколико тражени број не постоји у низу исписати одговарајућу поруку.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

int Poredi(const void* a, const void *b)
{
    return *(int*)a-*(int*)b;
}

main()
{
    int i, n, a[MAX], x;
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf ("%d", &a[i]);
    printf(" Broj koji se trazi: ");
    scanf("%d", &x);
    /*Adresa elementa koji trazimo, pocetak niza, broj elemenata,
       velicina jednog elementa, adresa funkcije poredjenja. */
    int* elem=(int*)bsearch((void*)&x,(void*)a, n, sizeof(int), Poredi);
    if(elem==NULL)
        printf(" Element nije pronadjen\n");
    else
        printf(" Element postoji na poziciji %d\n",elem-a+1);
    getche();
    return 0;
}
```



Испис на екрану

3 ДИНАМИЧКА ЗОНА МЕМОРИЈЕ

3.1 Основне конструкције програма са динамичком зоном меморије

Табела 3.1: Функције за рад са динамичком зоном меморије дефинисане у библиотеци <stdlib.h>

| функција | значење |
|---------------------------|---|
| <code>malloc(k)</code> | Функција додељује меморију од <code>k</code> бајтова. Вредност функције је генерички показивач (тип void*) на додељени простор, или <code>NULL</code> ако захтев не може да буде задовољен. Садржај додељеног простоја је недефинисан. |
| <code>calloc(n,k)</code> | Функција додељује меморију за низ од <code>n</code> елемената од којих сваки има по <code>k</code> бајтова. Вредност функције је генерички показивач (тип void*) на додељени простор, или <code>NULL</code> ако захтев не може да буде задовољен. Додељени простор се попуњава нулама. |
| <code>realloc(p,k)</code> | Функција мења величину додељене меморије на коју показује показивач <code>p</code> у <code>k</code> бајтова. Нова величина може да буде већа или мања од старе. У случају смањивања величине додељене меморије, скраћивање је од краја, а садржај задржаних бајтова се сачува. У случају повећавања додељене меморије, нови бајтови недефинисаног садржаја додају се на крај. Вредност функције је генерички показивач (тип void*) на ново место додељене меморије, или <code>NULL</code> ако захтев не може да буде задовољен. У случају неуспеха сачува се почетно место и садржај додељене меморије. |
| <code>free(p)</code> | Функција ослобађа простор на који показује показивач <code>p</code> . Показивач <code>p</code> мора да садржи вредност која је раније добијена као вредност неке од функција <code>malloc</code> , <code>calloc</code> или <code>realloc</code> . Важно је да се простор додељен подацима који више нису потребни, ослободи да би тај простор, по потреби, могао да буде додељен другим подацима у току истог програма. |

3.1. Шта реализују успешно извршене следеће наредбе:

а)

```
int *p;
p=malloc(5*sizeof(int));
```

б)

```
int *p;
p=calloc(5,100*sizeof(int));
```

в)

```
int *p;
p=malloc(10*sizeof(int));
...
p=realloc(p,50*sizeof(int));
```

г)

```
int *p;
p=malloc(5*sizeof(int));
...
free(p);
```

- а) Извршена је алокација меморије, тј. резервише меморијски блок за пет променљивих типа `int` и адресу овог блока додељује показивачу `p`, али не иницијализује овај меморијски блок.
- б) Резервише и иницијализује (на вредност 0) пет меморијских блокова, сваки од 50 променљивих типа `int` и адресу првог од ових блокова додељује показивачу `p`.
- в) Мења величину додељеног простора од адресе `p`, са 10 на 50 променљивих типа `int` и адресу овог блока додељује показивачу `p` (не мења се постојећи садржај од адресе `p`).
- г) Ослобађа динамички алоцирану меморију (ослобађа меморију од адресе `p`).

3.2. Саставити програм који учита један цео број и смешта га у динамички алоцирану меморију, а затим тај број исписује на стандардни излаз.

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

main()
{
    /*p je pokazivac na int, koji trenutno ne pokazuje ni na sta.*/
    int *p;

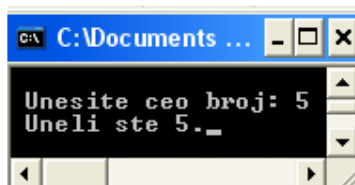
    /*Dinamicki alociramo prostor za jedan ceo broj.*/
    p=(int *)malloc(sizeof(int));

    /*Nakon alociranja prostora obavezno se mora proveriti
    da li je uspela alokacija! Ako je malloc vratila
    vrednost NULL, to nam je indikator neuspesne alokacije. */
    if(p==NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }

    printf("\n Unesite ceo broj: ");
    /*Ucitavamo ceo broj i smestamo ga u dinamicki rezervisan prostor.*/
    scanf("%d", p);

    /*Ispisujemo podatak na standardni izlaz.*/
    printf(" Uneli ste %d.", *p);

    /*Oslobadjamo dinamicki alociran prostor.*/
    free(p);
    getche();
    return 0;
}
```

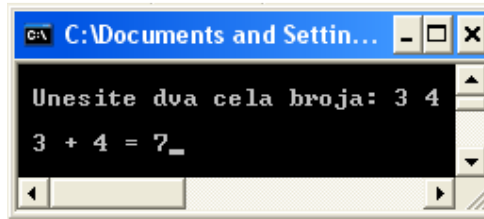


Испис на екрану

3.3. Саставити програм који чита са тастатуре два цела броја и исписује њихов збир на екрану. Меморију за бројеве алоцирати динамички.

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

main()
{
    int *p1, *p2;
    p1=(int*)malloc(sizeof(int));
    if(p1==NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }
    p2=(int*)malloc(sizeof(int));
    if(p2==NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }
    printf("\n Unesite dva cela broja: ");
    scanf("%d",p1);
    scanf("%d",p2);
    printf("\n %d + %d = %d", *p1, *p2, *p1+*p2);
    free(p1);
    free(p2);
    getch();
    return 0;
}
```



Испис на екрану

3.2 Низови и динамичка зона меморије (malloc())

3.4. Саставити програм који учита низ целих бројева дужине **n** и проналази и исписује највећи елемент. Елементе сместити у динамички алоцирану меморију.

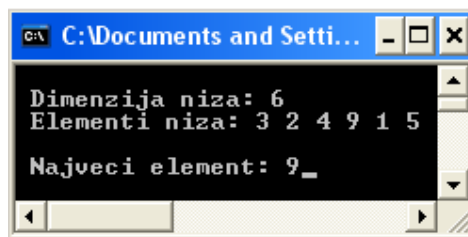
```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int i, n, max;
    int *a;
    printf("\n Dimenzija niza: ");
    scanf("%d", &n);
    a=(int*)malloc(n*sizeof(int));
    if(a==NULL)
    {
        printf("\n Nema dovoljno memorije!");
        return 1;
    }
}
```

```

printf(" Elementi niza: ");
for(i=0; i<n; i++)
    scanf("%d", &a[i]);
max=a[0];
for(i=1; i<n; i++)
    if(a[i]>max)
        max = a[i];
printf("\n Najveci element: %d", max);
free(a);
getche();
return 0;
}

```



Испис на екрану

Декларација `int a[n];` није дозвољена јер компилатор не може у време превођења да одреди потребну количину меморије. Уместо овога, меморију ћемо алоцирати динамички тј. у фази извршавања програма када буде позната вредност броја `n`. Због тога је потребно упамтити само адресу почетка алоцираног блока меморије што ћемо урадити коришћењем следећег показивача `int *a`.

У овом тренутку знамо колико нам је меморије потребно и позивамо функцију `malloc` за динамичку алокацију. `malloc`-у се прослеђује количина потребне меморије у бајтовима, а он враћа генерички показивач (`void*`) који пре коришћења треба конвертовати у одговарајући реални показивачки тип.

Надаље `a` користимо као обичан низ.

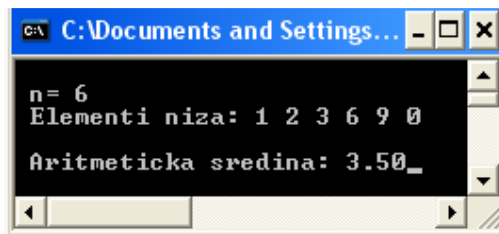
3.5. Саставити програм који учита низ целих бројева дужине `n` и проналази и исписује аритметичку средину елемената. Елементе низа сместити у динамички алоцирану меморију.

```

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

main()
{
    int n, i, *niz;
    double zbir=0.0;
    printf("\n n= ");
    scanf("%d", &n);
    niz=(int*)malloc(sizeof(int)*n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &niz[i]);
    for(i=0; i<n; i++)
        zbir += niz[i];
    printf("\n Aritmeticka sredina: %.2f", zbir/n);
    getche();
    return 0;
}

```

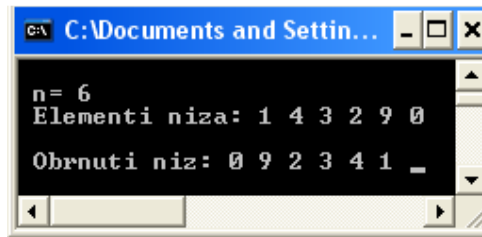


Испис на екрану

3.6. Саставити програм који учита низ целих бројева дужине **n** и формира нови низ са обрнутим редоследом елемената унетог низа. Исписати новодобијени низ. Елементе низа сместити у динамички алоцирану меморију.

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

main()
{
    int n, i, *a;
    printf("\n n= ");
    scanf("%d", &n);
    a=(int*)malloc(n*sizeof(int));
    if(a==NULL)
    {
        printf("\n Nema slobodne memorije!");
        return 1;
    }
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d",&a[i]);
    printf("\n Obrnuti niz: ");
    for(i=n-1; i>=0; i--)
        printf("%d ",a[i]);
    free(a);
    getch();
    return 0;
}
```

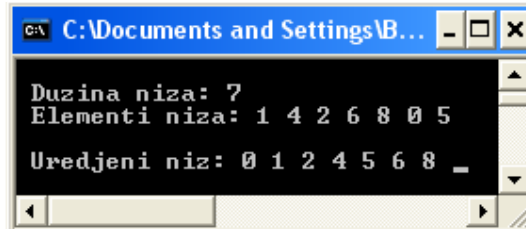


Испис на екрану

3.7. Саставити програм који учита низ целих бројева дужине **n** и уређује их у неоппадајућем редоследу методом уметања. Исписати уређени низ. Елементе низа сместити у динамички алоцирану меморију.

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

main()
{
    int n, *a, b, i, j;
    printf("\n Duzina niza: ");
    scanf("%d", &n);
    a=(int*)malloc (n*sizeof(int));
    if(a==NULL)
    {
        printf("\n Nema slobodne memorije!");
        return 1;
    }
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d",&a[i]);
    for(i=1; i<n; i++)
    {
        b=a[i];
        for(j=i-1; j>=0 && a[j]>b; j--)
            a[j+1]=a[j];
        a[j+1]=b;
    }
}
```



Испис на екрану

```

printf ("\n Uredjeni niz: ");
for(i=0; i<n; i++)
    printf ("%d ", a[i]);
free(a);
getche();
return 0;
}

```

3.8. Саставити програм који учита динамички низ целих бројева **A** дужине **n**, а затим од њега формира динамичке низове **B** и **C** који се састоје од негативних, односно позитивних бројева низа **A**. Исписати новокреиране низове.

```

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

main()
{
    int *a, *b, *c, i, j=0, k=0, n, n1=0;
    printf("\n Duzina niza: ");
    scanf("%d",&n);
    a=(int *)malloc(n*sizeof(int));
    if(a==NULL)
    {
        printf("\n Nema dovoljno memorije!");
        return 1;
    }
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
        if(a[i]>=0)
            n1++;
    }
    b=(int *)malloc(n1*sizeof(int));
    c=(int *)malloc((n-n1)*sizeof(int));
    if((b==NULL)|| (c==NULL))
    {
        printf("\n Nema dovoljno memorije!");
        return 1;
    }
    for(i=0; i<n; i++)
        if(a[i]>=0)
            b[j++]=a[i];
        else
            c[k++]=a[i];
    free(a);
    printf("\n Pozitivni niz: ");
    for(i=0; i<n1; i++)
        printf("%d ",b[i]);
    printf("\n Negativni niz: ");
    for(i=0; i<n-n1; i++)
        printf("%d ",c[i]);
    free(b);
    free(c);
    getche();
    return 0;
}

```

```

C:\Documents and Settings\Ban...
Duzina niza: 8
Elementi niza: 1 5 -2 7 3 2 -1 9
Pozitivni niz: 1 5 7 3 2 9
Negativni niz: -2 -1 _

```

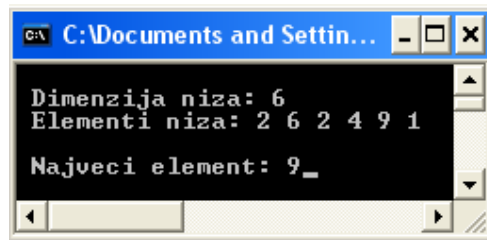
Испис на екрану

3.9. Саставити функцију која врши алокацију меморије за динамички низ целих бројева дужине **n**, а затим саставити главни програм који учита низ дужине **n** и исписује елемент који има максималну вредност.

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

int *Kreiraj(int n)
{
    return (int*)malloc(n*sizeof(int));
}

main()
{
    int n, *a, i, max;
    printf("\n Dimenzija niza: ");
    scanf("%d", &n);
    a=Kreiraj(n);
    if(a==NULL)
    {
        printf("\n Nema dovoljno memorije!");
        return 1;
    }
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    max=a[0];
    for(i=1; i<n; i++)
        if(a[i] > max)
            max=a[i];
    printf("\n Najveci element: %d", max);
    free(a);
    getche();
    return 0;
}
```



Испис на екрану

3.10. Саставити функцију која креира динамички стринг надовезивањем два стринга **s1** и **s2**. Адреса низа се враћа као повратна вредност. Затим саставити програм који помоћу претходно формиране функције надовезује два унета стринга и исписује резултујући стринг.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 1000

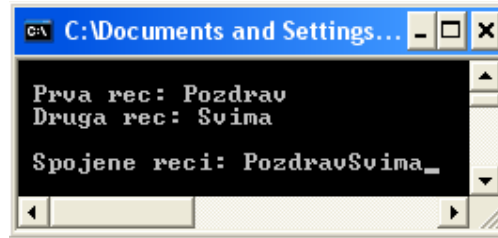
char *Nadovezi(char *s1, char *s2)
{
    int i, j;
    char *p;
    p=(char *)malloc((strlen(s1)+strlen(s2)+1)*sizeof(char));
    if(p==NULL)
        printf("\n Nema dovoljno memorije!");
    strcpy(p, s1);
    strcat(p, s2);
    return p;
}
```



```

main()
{
    char *s;
    char s1[MAX], s2[MAX];
    printf("\n Prva rec: ");
    scanf("%s", &s1);
    printf(" Druga rec: ");
    scanf("%s", &s2);
    s=Nadovezi(s1, s2);
    printf("\n Spojene reci: %s", s);
    free(s);
    getch();
    return 0;
}

```

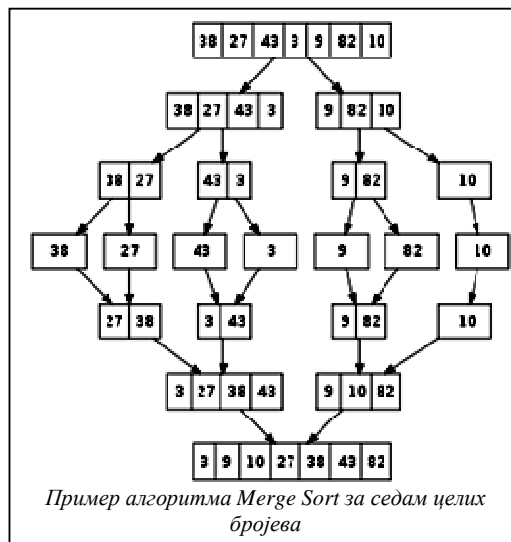


Испис на екрану

3.1:

Саставити функцију за сортирање низа целих бројева дужине **n** у неоппадајући поредак методом *Merge Sort* користећи као аргумент функције показивач на низ. За формирање функције користити динамичку меморију. Затим саставити програм који за унети низ целих бројева дужине **n** употребом претходне функције сортира низ и испишује новодобијени низ.

Merge Sort: Уколико имамо низ дужине 1, он је већ сортиран, у супротном поделити низ у два подниза на пола (или приближно пола). Рекурзивно сортирати оба подниза. На крају, спојити два сортирана подниза у један.



Пример алгоритма Merge Sort за седам целих бројева

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 100

/*Funkcija koja spaja dva podniza tako da se dobije sortirani niz*/
void Merge(int *a, int n1, int n2)
{
    int i=0, j1=0, j2=0;
    int *novi=(int*)malloc((n1+n2)*sizeof(int));
    while(j1<n1 && j2<n2)
        novi[i++]=(a[j1]<=a[n1+j2]) ? a[j1++] : a[n1+j2++];
    while(j1<n1)
        novi[i++] = a[j1++];
    while(j2<n2)
        novi[i++] = a[n1+j2++];
    for(i=0; i<n1+n2; i++)
        a[i]=novi[i];
    free(novi);
}

/*Funkcija koja sortira niz*/
void MergeSort(int *a, int n)
{
    int n1, n2;

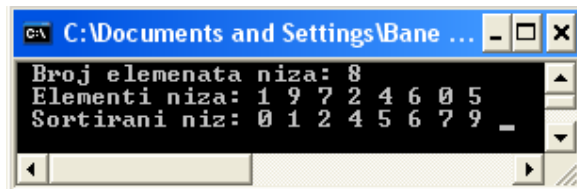
```

```

    if(n>1)
    {
        int n1=n/2;
        int n2=n-n1;
        MergeSort(a, n1);
        MergeSort(a+n1, n2);
        Merge(a, n1, n2);
    }
}

main ()
{
    int i, j, n, a[MAX];
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    MergeSort(a, n);
    printf(" Sortirani niz: ");
    for(i=0; i<n; i++)
        printf ("%d ",a[i]);
    getch();
    return 0;
}

```



Испис на екрану

3.3 Низови и динамичка зона меморије (calloc() и realloc())

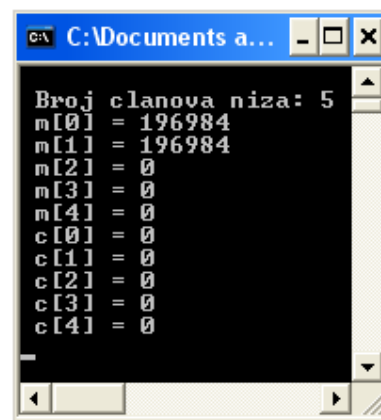
3.12. Које вредности ће бити исписане на екрану након извршавања следећег програмског кода:

```

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

main()
{
    int *m, *c, i, n;
    printf("\n Broj clanova niza: ");
    scanf("%d", &n);
    /*Niz m NE MORA garantovano da ima sve nule*/
    m=(int*)malloc(n*sizeof(int));
    if(m==NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }
    /*Niz c MORA garantovano da ima sve nule*/
    c=(int*)calloc(n, sizeof(int));
    if(c == NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }
    /*Odstampace se sadrzaj neinicijalizovane memorije*/
    for(i=0; i<n; i++)
        printf(" m[%d] = %d\n", i, m[i]);
}

```



Испис на екрану

```

/*Odstampace se nule*/
for(i=0; i<n; i++)
    printf(" c[%d] = %d\n", i, c[i]);
free(m);
free(c);
getche();
return 0;
}

```

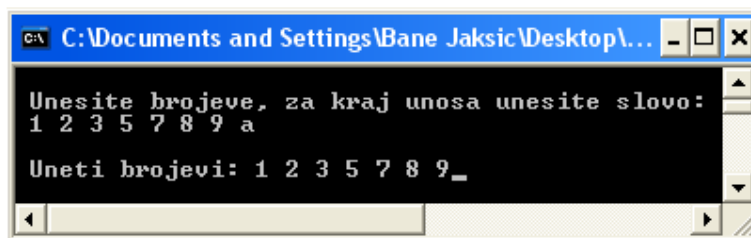
3.13. Саставити програм који алоцира низ од **max=5** целобројних елемената. Корисник затим уноси низ бројева. Ако број унетих бројева премаши **max**, тада се повећава алоцирана меморија за додатних 5 елемената. Унос се прекида када корисник укуца слово. На крају се исписује низ и деалоцира меморија.

```

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

main()
{
    int *a; /*pokazivac na niz celih brojeva*/
    int max=5; /*pocetna maksimalna velicina niza*/
    int br=0; /*pocetno je u nizu 0 elemenata*/
    int i, podatak;
    a=malloc(max*sizeof(int));
    if(a==NULL)
    {
        printf("\n Nema slobodne memorije!");
        return 1;
    }
    printf("\n Unesite brojeve, za kraj unosa unesite slovo:\n");
    while(scanf("%d",&podatak)==1)
    {
        a[br++]=podatak;
        if(br>=max)
        {
            max+=5;
            a=realloc(a, max*sizeof(int));
            if(a==NULL)
            {
                printf("\n Nema slobodne memorije!");
                return 1;
            }
        }
    }
    printf("\n Uneti brojeva:\n");
    for(i=0; i<br; i++)
        printf(" %d\n", a[i]);
    free(a);
    getche();
    return 0;
}

```



Испис на екрану

3.14. Саставити програм који са стандардног улаза чита целе бројеве све док се не унесе 0. Бројеви се смештају у низ. Не правити никакве претпоставке о димензији низа целих бројева. Унети низ исписати у обрнутом редоследу. Меморију проширивати:

- а) употребом **realloc()** функције;
- б) без употребе **realloc()** функције.

а)

```
#include <stdio.h>
#include <stdlib.h>
#define KORAK 10

main()
{
    int *a=NULL , duzina=0, alocirano=0, n, i;
    printf("\n Unosite brojeve. Za kraj unosa unesite 0:\n");
    do
    {
        scanf("%d", &n);
        /*Ukoliko nema vise slobodnih mesta, vrsi se prosirivanje.*/
        if(duzina==alocirano)
        {
            /*Niz se prosiruje na 10 elemenata vise.*/
            alocirano+=KORAK;
            a=realloc(a, alocirano*sizeof(int));
        }
        a[duzina]=n;
        duzina++;
    } while(n!=0);
    printf("\n Obrnuti niz: ");
    for(i=duzina-2; i>=0; i--)
        printf(" %d",a[i]);
    free(a);
    getche();
    return 0;
}
```

б)

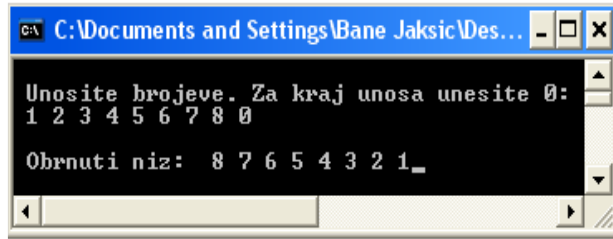
```
#include <stdio.h>
#include <stdlib.h>
#define KORAK 10

main()
{
    int *a=NULL, duzina=0, alocirano=0, n, i;
    printf("\n Unosite brojeve. Za kraj unosa unesite 0:\n");
    do
    {
        scanf("%d", &n);
        /*Ukoliko nema vise slobodnih mesta, vrsi se prosirivanje.*/
        if(duzina==alocirano)
        {
            /*Kreira se novi niz*/
            int *novi;
            /*Za njega se alocira 10 elemenata vise od prethodnog.*/
            alocirano+=KORAK;
            novi=malloc(alocirano*sizeof(int));
            /*Kopira se sadrzaj starog niza u novi.*/
            for(i=0; i<duzina; i++)
                novi[i]=a[i];
            /*Oslobadja se stari niz.*/
            free(a);
        }
    }
}
```

```

        /*Stari niz postaje novi.*/
        a=novi;
    }
    /* Ucitani broj se upisuje u niz.*/
    a[duzina]=n;
    duzina++;
} while(n!=0);
printf("\n Obrnuti niz: ");
for(i=duzina-2; i>=0; i--)
    printf(" %d",a[i]);
free(a);
getche();
return 0;
}

```



Испис на екрану

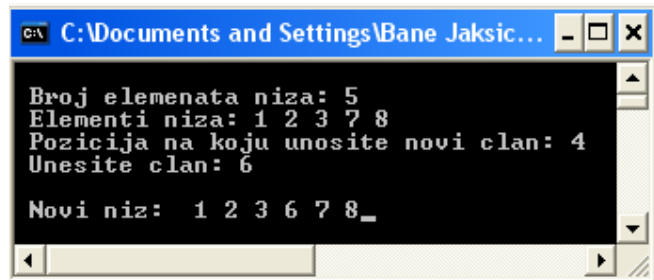
3.15. Саставити програм за унети низ целих бројева дужине **n** врши уметање новог елемента између **j**-тог и **j+1**-ог елемента датог низа. Исписати новоформиранни низ. Низ сместити у динамичку зону меморије.

```

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

main()
{
    int *a;
    int n, i, j, k;
    printf("\n Broj elemenata niza: ");
    scanf("%d", &n);
    a=malloc(n*sizeof(int));
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    printf(" Pozicija na koju unosite novi clan: ");
    scanf("%d", &j);
    if(j<=0 || j>n)
    {
        printf("\n Unesete pozitivan ceo broj koji je manji od broja %d.",
            n+1);
        return 1;
    }
    printf(" Unesite clan: ");
    scanf("%d", &k);
    a=realloc(a, (n+1)*sizeof(int));
    for(i=n; i>j-1; i--)
        a[i]=a[i-1];
    a[j-1]=k;
    printf("\n Novi niz: ");
    for(i=0; i<n+1; i++)
        printf(" %d", a[i]);
    free(a);
    getche();
    return 0;
}

```



Испис на екрану

3.16. Саставити програм за налажење уније, пресека и разлике два скупа реалних бројева. Скупове сместити у динамичку зону меморије.

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

main()
{
    int *s1, *s2, *s3;
    int n1, n2, n3, i, j;
    printf ("\n Broj elemenata niza S1: ");
    scanf ("%d", &n1);
    s1=malloc(n1*sizeof(int));
    printf(" Elementi niza S1= ");
    for(i=0; i<n1; i++)
        scanf("%d", &s1[i]);
    if(n1==0) printf("\n");
    printf("\n Broj elemenata niza S2= ");
    scanf ("%d", &n2);
    s2=malloc(n2*sizeof(int));
    printf(" Elementi niza S2: ");
    for(i=0; i<n2; i++)
        scanf("%d", &s2[i]);
    if(n2==0) printf ("\n");

    /*Unija dva skupa*/
    s3=malloc((n1+n2)*sizeof(int));
    for(n3=0; n3<n1; n3++)
        s3[n3] = s1[n3];
    for(j=0; j<n2; j++)
    {
        for(i=0; i<n1 && s1[i]!=s2[j]; i++);
        if(i == n1)
            s3[n3++] = s2[j];
    }
    s3=realloc(s3, n3*sizeof(int));
    printf("\n S1+S2= ");
    for(i=0; i<n3; i++)
        printf("%d ", s3[i]);
    free(s3);

    /*Presek dva skupa*/
    s3=malloc((n1<n2 ? n1 : n2)*sizeof(int));
    for(n3=i=0; i<n1; i++)
    {
        for(j=0; j<n2 && s1[i]!=s2[j]; j++);
        if(j < n2)
            s3[n3++] = s1[i];
    }
    s3=realloc(s3, n3*sizeof(int));
    printf("\n S1*S2= ");
    for(i=0; i<n3; i++)
        printf("%d ", s3[i]);
    free(s3);

    /*Razlika dva skupa*/
    s3=malloc(n1*sizeof(int));
    for(n3=i=0; i<n1; i++)
    {
        for(j=0; j<n2 && s1[i]!=s2[j]; j++);
```

```

        if(j == n2)
            s3[n3++] = s1[i];
    }
    s3=realloc(s3, n3*sizeof(int));
    printf ("\n S1-S2= ");
    for(i=0; i<n3; i++)
        printf("%d ", s3[i]);
    free(s3);

    free(s1);
    free(s2);
    getch();
    return 0;
}

```

```

C:\Documents and Settings...
Broj elemenata niza S1: 5
Elementi niza S1: 1 2 3 4 5

Broj elemenata niza S2: 3
Elementi niza S2: 4 5 6

S1+S2= 1 2 3 4 5 6
S1*S2= 4 5
S1-S2= 1 2 3 _

```

Испис на екрану

3.4 Низ показивача и динамичка алокација меморије

3.17. Пример употребе низа показивача. Који је резултат извршавања следећег програмског кода:

```

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

main()
{
    /*Niz od tri elemenata tipa int.*/
    int nizi[3];
    /*Niz od tri elemenata tipa int*, dakle
    niz od tri pokazivaca na int.*/
    int *nizip[3];
    /*Alociramo memoriju za prvi element niza.*/
    nizip[0]=(int*) malloc(sizeof(int));
    if(nizip[0]==NULL)
    {
        printf("\n Nema slobodne memorije!");
        return 1;
    }
    /*Upisujemo u prvi element niza broj 5.*/
    *nizip[0]=5;
    printf("\n %d", *nizip[0]);
    /*Alociramo memoriju za drugi element niza.
    Drugi element niza pokazuje na niz od dva elementa.*/
    nizip[1]=(int*) malloc(2*sizeof(int));
    if(nizip[1] == NULL)
    {
        printf("\n Nema slobodne memorije!");
        free(nizip[0]);
        return 1;
    }
    /*Pristupamo prvom elementu na koji pokazuje pokazivac nizip[1].*/
    *(nizip[1]) = 1;
    /*Pristupamo sledecem elementu u nizu na koji pokazuje nizip[1].*/
    *(nizip[1] + 1) = 2;
}

```

```

printf(" %d", nizip[1][1]);
/*Alociramo memoriju za treci element niza nizip.*/
nizip[2]=(int*) malloc(sizeof(int));
if(nizip[2] == NULL)
{
    printf("\n Nema slobodne memorije!");
    free(nizip[0]);
    free(nizip[1]);
    return 1;
}
*(nizip[2])=2;
printf(" %d", *(nizip[2]));
free(nizip[0]);
free(nizip[1]);
free(nizip[2]);
getche();
return 0;
}

```



Испис на екрану

3.18. Саставити програм који омогућује да корисник уноси произвољан број линија текста. Почетно се алоцира меморија за 5 линија текста. Ако корисник унесе више линија, тада се помоћу функције **realloc()** повећава потребна меморија за додатних 5 линија. Унос се завршава када се откуца празна линија. Након тога се врши сортирање текста помоћу функције **qsort()**. На крају програма исписује се сортирани текст и деалочира меморија.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 1000

int CmpString(const void *pstr1, const void *pstr2 )
{
    return strcmp(*(char **)pstr1, *(char **)pstr2 );
}

main()
{
    char **tekst; /*pokazivac na pokazivac stringa*/
    int max=5; /*pocetna maksimalna velicina niza */
    int n=0; /*pocetno je u nizu 0 stringova*/
    char str[MAX]={"\0"};
    int i;
    tekst=malloc(max*sizeof(char *));
    if(tekst==NULL )
    {
        printf("\n Nema slobodne memorije!");
        return 1;
    }
    printf("\n Unesite tekst:\n\n");
    while(gets(str) != NULL)
    {
        /*stvora kopiju stringa s, i vraca pokazivac
        na novoformirani string*/
        char *s = strdup(str);
        if(s== NULL)
        {
            printf("\n Nema slobodne memorije!");
            return 1;
        }
    }
}

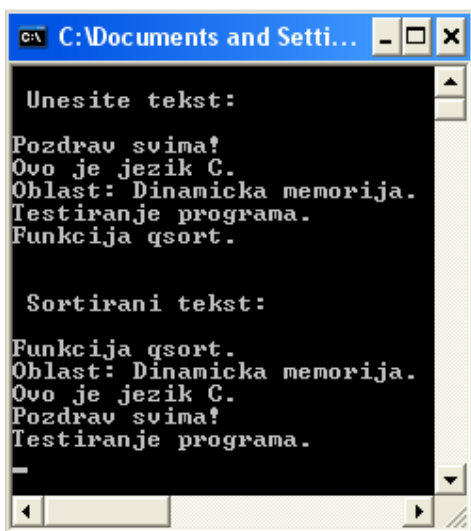
```



```

    if(strlen(s)==0) break;
    tekst[n++]=s;
    if(n>=max)
    {
        max+=5;
        tekst=realloc(tekst,max*sizeof(int));
        if(tekst== NULL)
        {
            printf("\n Nema slobodne memorije!");
            return 1;
        }
    }
    if(n>1)
        qsort((void *)tekst, n, sizeof(char*), CmpString);
    printf("\n Sortirani tekst:\n\n");
    for(i=0; i<n; i++)
    {
        puts(tekst[i]);
        free(tekst[i]);
    }
    free(tekst);
    getch();
    return 0;
}

```



Испис на екрану

Упоредна функција `CmpString` се користи у функцији `qsort()`. Према договору, у упоредну функцију се шаљу показивачи на елемент низа, у овом случају показивач на стринг, односно `char**`. За поређење стрингова користи се функција `strcmp()`. Пошто аргумент функције `strcmp` мора бити стринг (`char *`) то значи да јој се мора послати `*pstr1` и `*pstr2`, односно садржај аргумената `pstr1` и `pstr2`, који су показивачи типа `char**`.

3.5 Матрице и динамичка зона меморије

3.19. Саставити програм који формира матрицу реланих бројева од **m** врста и **n** колона у динамичкој зони меморије помоћу показивача на показивач. Исписати елементе матрице и одредити и исписати траг матрице (сума елемената на главној дијагонали), еуклидску норму матрице (корен суме квадрата свих елемената) и горњу вандијагоналну норму (сума апсолутних вредности елемената изнад главне дијагонале).

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

main()
{
    int i, j, m, n;
    double **a;
    double trag, norma, vdnorma;
    printf("\n Broj vrsta: ");
    scanf("%d", &m);
    printf(" Broj kolona: ");
    scanf("%d", &n);
    /*dinamicki alociramo prostor za n pokazivaca na double-ove*/
    a=(double **)malloc(m*sizeof(double*));
    if(a == NULL)
    {
        printf("\n GRESKA!!!");
        exit(1);
    }
    /*dinamicki alociramo prostore za elemente u vrstama*/
    for(i=0; i<m; i++)
    {
        a[i]=(double *)malloc(n*sizeof(double));
        if(a[i]==NULL)
        {
            /*Dalje ne mozemo da nastavimo. Trebalo bi da napustimo program,
            ali pre toga moramo da oslobodimo svih i-1 prethodno alociranih
            vrsta,i alociran niz pokazivaca.*/
            for( j=0; j<i; j++)
                free(a[j]);
            free(a);
            exit(1);
        }
    }
    printf("\n Unesite elemente matrice:\n");
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
        {
            printf(" a[%d][%d] = ", i, j);
            scanf("%lf", &a[i][j]);
        }
    printf("\n Uneli ste matricu:\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            printf(" %.2f", a[i][j]);
        printf("\n");
    }

    trag=0.0;
    for(i=0; i<n; i++)
        trag += a[i][i];
    printf(" Trag matrice: %.2f\n",trag);

    norma=0.0;
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            norma += a[i][j]*a[i][j];
    norma=sqrt(norma);
    printf(" Euklidska norma matrice: %.2f\n", norma);
```

```

vdnorma = 0.0;
for(i=0; i<m; i++)
    for(j=i+1; j<n; j++)
        vdnorma+=fabs(a[i][j]);
printf(" Gornja vandijagonalna norma matrice: %.2f\n", vdnorma);

for( j=0; j<m; j++)
    free(a[j]);
free(a);
getche();
return 0;
}

```

За динамичку алокацију матрице применили смо следећи поступак:

1. Декларишемо показивач-на-показивач на **double**:

```
double **a;
```

Овај показивач ће нам служити да чува адресу првог у низу динамички алоцираних показивача на **double** елемент.

2. Динамички алоцирамо низ од m показивача на **double**-ове, и адресу првог у низу смештамо у променљиву a :

```
a=(double **) malloc(m*sizeof(double*));
```

Приметимо да је име типа **double*** показивач на **double**. Ових m показивача ће нам служити да показују на прве елементе низова **double**-ова који представљају динамички алоциране врсте.

3. Сваку од m врста динамички алоцирамо посебним позивом **malloc** функције. Повратну адресу смештамо редом у мало пре алоцирани низ показивача. За i -ту врсту повратну вредност смештамо у $a[i]$:

```
for(i=0; i<m; i++)
    a[i]=malloc(n*sizeof(double));
```

Приметимо да низови **double**-ова који се алоцирају нису морали бити сви једнаких димензија. Нису морали сви имати n елемената као што стоји у примеру. Ово може бити корисно у ситуацијама када знамо да ће у некој врсти бити коришћено само неколико првих елемената. Тада можемо алоцирати мање простора за ту врсту и тако уштедети простор.

4. Елементима динамички алоциране матрице приступамо исто као и код аутоматски алоциране матрице, са $a[i][j]$. $a[i]$ је i -ти показивач у низу, а како он показује на почетни елемент у i -тој врсти, тада је $a[i][j]$ управо j -ти елементи у i -тој врсти.

5. Деалокација се састоји у ослобађању динамички алоцираних врста, након чега се ослобађа низ показивача. Дакле, деалокација иде у супротном редоследу од редоследа алокације:

```
for(i=0; i<m; i++)
    free(a[i]);
free(a);
```

```

C:\Documents and Settings\Bane Jaksic\Des...
Broj vrsta: 3
Broj kolona: 2

Unesite elemente matrice:
a[0][0] = 1
a[0][1] = 2
a[1][0] = 4
a[1][1] = 5
a[2][0] = 2
a[2][1] = 0

Uneli ste matricu:
1.00 2.00
4.00 5.00
2.00 0.00
Trag matrice: 6.00
Euklidska norma matrice: 7.07
Gornja vandijagonalna norma matrice: 2.00

```

Испис на екрану

3.20. Саставити програм који формира квадратну матрицу целих бројева димензија $n \times n$, а затим исписује вредности елемената матрице и збир елемената испод споредне дијагонале. Матрицу сместити у динамичку зону меморије.

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

main()
{
    int **a;
    int n, i, j, k, suma=0, x=0;
    printf("\n Broj vrsta i kolona: ");
    scanf("%d", &n);
    a=(int **)malloc(n*sizeof(int*));
    if(a==NULL)
    {
        printf("\n GRESKA!");
        return 1;
    }
    for(i=0; i<n; i++)
    {
        a[i]=(int *)malloc(n*sizeof(int));
        if(a[i]==NULL)
        {
            printf("\n GRESKA!");
            for(k=0; k<i; k++)
                free(a[k]);
            free(a);
            return 1;
        }
    }
    printf("\n Unesite elemente matrice:\n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            printf(" a[%d][%d] = ", i, j);
            scanf("%d", &a[i][j]);
        }
    }
    printf("\n Izgled matrice:\n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf(" %d", a[i][j]);
        printf("\n");
    }
    for(i=n-1; i>=0; i--)
    {
        x++;
        for(j=n-1; j>=x; j--)
            suma+=a[i][j];
    }
    printf("\n Zbir= %d", suma);
    for(i=0; i<n; i++)
        free(a[i]);
    free(a);
    getch();
    return 0;
}
```

```
C:\Documents and Sett...
Broj vrsta i kolona: 3

Unesite elemente matrice:
a[0][0] = 1
a[0][1] = 2
a[0][2] = 3
a[1][0] = 4
a[1][1] = 0
a[1][2] = 1
a[2][0] = 0
a[2][1] = 2
a[2][2] = 2

Izgled matrice:
1 2 3
4 0 1
0 2 2

Zbir= 5
```

Испис на екрану

3.21. Саставити програм за транспоновање правоугаоне матрице целих бројева димензија **mxn** и њено исписивање. Матрицу сместити у динамичку зону меморије.

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

main()
{
    int **a, **b, m, n, i, j;
    printf("\n Broj vrsta: ");
    scanf("%d", &m);
    printf(" Broj kolona: ");
    scanf("%d", &n);

    /*Alokacija memorije za unetu matricu:*/
    a=(int **)malloc(m*sizeof(int*));
    if(a == NULL)
    {
        printf("\n GRESKA!!!");
        return 1;
    }
    for(i=0; i<m; i++)
    {
        a[i]=(int *)malloc(n*sizeof(int));
        if(a[i]==NULL)
        {
            for(j=0; j<i; j++)
                free(a[j]);
            free(a);
            return 1;
        }
    }

    /*Formiranje i stampanje matrice:*/
    printf("\n Unesite elemente matrice:\n");
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
        {
            printf(" a[%d][%d] = ", i, j);
            scanf("%d", &a[i][j]);
        }
    printf("\n Izgled matrice:\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            printf(" %3d", a[i][j]);
        printf("\n");
    }

    /*Alokacija memorije za transponovanu matricu:*/
    b=(int **)malloc(n*sizeof(int*));
    if(b == NULL)
    {
        printf("\n GRESKA!!!");
        return 1;
    }
    for(i=0; i<n; i++)
    {
        b[i]=(int *)malloc(m*sizeof(int));
        if(a[i]==NULL)
```

```

    {
        for(j=0; j<i; j++)
            free(b[j]);
        free(b);
        return 1;
    }

    /*Obrazovanje transponovane matrice:*/
    for(i=0; i<n; i++)
    {
        for(j=0; j<m; j++)
            b[i][j]=a[j][i];
    }

    /*Zamena stare matrice novom matricom:*/
    for(i=0; i<m; i++)
        free(a[i]);
    free(a);
    a=b;
    i=m;
    m=n;
    n=i;

    /*Ispisivanje transponovane matrice: */
    printf ("\n Transponovana matrica:\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            printf(" %3d", a[i][j]);
        printf ("\n");
    }

    /*Unistavanje matrice: */
    for(i=0; i<m; i++)
        free(a[i]);
    free (a);
    getche();
    return 0;
}

```

```

C:\Documents and Sett...
Broj vrsta: 3
Broj kolona: 2

Unesite elemente matrice:
a[0][0] = 1
a[0][1] = 2
a[1][0] = 4
a[1][1] = 6
a[2][0] = 7
a[2][1] = 8

Izgled matrice:
1 2
4 6
7 8

Transponovana matrica:
1 4 7
2 6 8

```

Испис на екрану

3.22. Саставити програм који формира матрицу реалних бројева димензија **mxn**. Програм треба да израчуна колико постоји елемената матрице који су једнаки аритметичкој средини својих суседа и испише њихове позиције у матрици. Матрицу сместити у динамичку зону меморије. Програм треба да обради произвољан број улазних матрица.

```

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

/*funkcija za unos svih elemenata matrice*/
float **UnosElementa(float **a, int m, int n)
{
    int i, j, k;
    float temp = 0;
    a=(float **)malloc(m*sizeof(float*));
    if(a==NULL)
    {
        printf("GRESKA!.\n");
        return a;
    }
    printf(" Unesite elemente matrice:\n");

```

```
for(i=0; i<m; i++)
{
    a[i]=(float *)malloc(n*sizeof(float));
    if(a[i]==NULL)
    {
        printf(" GRESKA!\n");
        for(k=0; k<i; k++)
            free(a[k]);
        free(a);
        a=NULL;
        return a;
    }
    for(j=0; j<n; j++)
    {
        printf(" [%d, %d] = ", i, j);
        scanf("%f", &a[i][j]);
    }
}
return a;
}

/*funkcija za stampanje svih elemenata matrice*/
void IspisMatrice(float **a, int m, int n)
{
    int i, j;
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            printf("%5.2f ", a[i][j]);
        printf("\n");
    }
}

/*funkcija za izracunavanje broja elemenata,
koji su jednaki aritmetickoj sredini svojih suseda*/
int Izracunaj(float **a, int m, int n)
{
    int i, j;
    int temp=0; /*broj trazених elemenata*/
    int broj=0; /*broj suseda datog elementa*/
    float templ = 0; /*za racunanje sume suseda datog elementa*/
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            if((i-1)>=0) /*da li postoji red iznad*/
            {
                templ+=a[i-1][j];
                broj += 1;
                if((j-1)>=0) /*da li postoji kolona levo*/
                {
                    templ += a[i-1][j-1];
                    broj += 1;
                }
                if((j+1)<n) /*da li postoji kolona desno*/
                {
                    templ += a[i-1][j+1];
                    broj += 1;
                }
            }
            if((i+1)<m) /*da li postoji red ispod*/
            {

```

```

        temp1 += a[i+1][j];
        broj += 1;
        if((j-1)>=0) /*da li postoji kolona levo*/
        {
            temp1 += a[i+1][j-1];
            broj += 1;
        }
        if((j+1)<n) /*da li postoji kolona desno*/
        {
            temp1 += a[i+1][j+1];
            broj += 1;
        }
    }
    if((j-1)>=0) /*da li postoji kolona levo*/
    {
        temp1 += a[i][j-1];
        broj += 1;
    }
    if((j+1)<n) /*da li postoji kolona desno*/
    {
        temp1 += a[i][j+1];
        broj += 1;
    }
    if(broj>0)
        temp1 /= broj; /*izracunavanje aritmeticke sredine
                        suseda datog elementa*/
    if(a[i][j] == temp1)
        temp += 1; /*izracunavanje broja elemenata, koji su jednaki
                    aritmetickoj sredini svojih suseda*/
    temp1=0;
    broj=0;
}
}
return temp;
}

/*funkcija za stampanje pozicija elemenata,
koji su jednaki aritmetickoj sredini svojih suseda*/
void IspisRezultata(float **a, int m, int n)
{
    int i, j, temp=0, broj=0;
    float temp1=0;
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            if((i-1)>=0) /*da li postoji red iznad*/
            {
                temp1 += a[i-1][j];
                broj += 1;
                if((j-1)>=0) /*da li postoji kolona levo*/
                {
                    temp1 += a[i-1][j-1];
                    broj += 1;
                }
                if((j+1)<n) /*da li postoji kolona desno*/
                {
                    temp1 += a[i-1][j+1];
                    broj += 1;
                }
            }
            if((i+1)<m) /*da li postoji red ispod*/

```



```

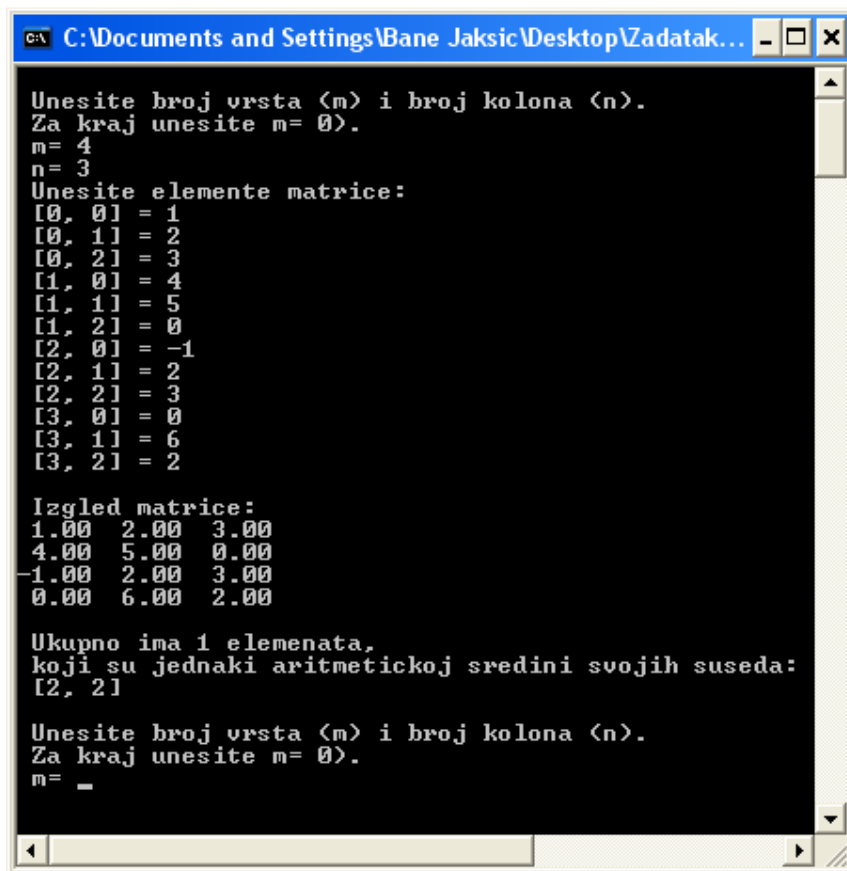
    {
        templ += a[i+1][j];
        broj += 1;
        if((j-1)>=0) /*da li postoji kolona levo*/
        {
            templ += a[i+1][j-1];
            broj += 1;
        }
        if((j+1)<n) /*da li postoji kolona desno*/
        {
            templ += a[i+1][j+1];
            broj += 1;
        }
    }
    if((j-1)>=0) /*da li postoji kolona levo*/
    {
        templ += a[i][j-1];
        broj += 1;
    }
    if((j+1)<n) /*da li postoji kolona desno*/
    {
        templ += a[i][j+1];
        broj += 1;
    }
    if(broj>0)
        templ /= broj; /*izracunavanje aritmeticke sredine
                        suseda datog elementa*/
    if(a[i][j] == templ)
        printf(" [%d, %d]\n", i, j); /*stampanje pozicije elementa, koji je
                                    jednak aritmetickoj sredini svojih suseda*/
    templ=0;
    broj=0;
}
}

/*funkcija za oslobadjanje dinamicke alocirane memorije*/
void OslobadjanjeMemorije(float **a, int x)
{
    int i;
    if(a != NULL)
    {
        for(i=0; i<x; i++)
        {
            if(a[i] != NULL)
                free(a[i]);
        }
        free(a);
    }
}

main()
{
    while(1)
    {
        float **a;
        int m, n;
        printf("\n Unesite broj vrsta (m) i broj kolona (n).");
        printf("\n Za kraj unesite m= 0.");
        printf("\n m= ");
        scanf("%d", &m);
        if(m==0) break;
    }
}

```

```
    if(m<0)
    {
        printf("\n Broj vrsta mora biti veci od 0.");
        break;
    }
    printf(" n= ");
    scanf("%d", &n);
    if(n<0)
    {
        printf("\n Broj kolona mora biti veci od 0.");
        break;
    }
    a=UnosElemenata(a, m, n);
    if(a==NULL) break;
    printf("\n Izgled matrice:\n");
    IspisMatrice(a, m, n);
    printf("\n Ukupno ima %d elemenata,", Izracunaj(a, m, n));
    printf("\n koji su jednaki aritmetickoj sredini svojih suseda:\n");
    IspisRezultata(a, m, n);
    OslobadjanjeMemorije(a, m);
}
getche();
return 0;
}
```



```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak...
Unesite broj vrsta <m> i broj kolona <n>.
Za kraj unesite m= 0).
m= 4
n= 3
Unesite elemente matrice:
[0, 0] = 1
[0, 1] = 2
[0, 2] = 3
[1, 0] = 4
[1, 1] = 5
[1, 2] = 0
[2, 0] = -1
[2, 1] = 2
[2, 2] = 3
[3, 0] = 0
[3, 1] = 6
[3, 2] = 2

Izgled matrice:
1.00 2.00 3.00
4.00 5.00 0.00
-1.00 2.00 3.00
0.00 6.00 2.00

Ukupno ima 1 elemenata,
koji su jednaki aritmetickoj sredini svojih suseda:
[2, 2]

Unesite broj vrsta <m> i broj kolona <n>.
Za kraj unesite m= 0).
m= _
```

Испис на екрану

3.23. Релација је дата квадратном матрицом целих бројева димензија **$n \times n$** . Саставити програм који формира матрицу и проверава и исписује да ли је релација представљена матрицом рефлексивна, симетрична и транзитивна. Матрицу сместити у динамичку зону меморије.

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

/*alokacija kvadratne matrice nxn*/
int **Alociraj(int n)
{
    int **a, i, k;
    a=(int**)malloc(n*sizeof(int*));
    if(a==NULL)
    {
        printf(" GRESKA!\n");
        exit(1);
    }
    for(i=0; i<n; i++)
    {
        a[i]=(int*)malloc(n*sizeof(int));
        if(a[i] == NULL)
        {
            printf(" GRESKA!\n");
            for(k=0; k<i; k++)
                free(a[k]);
            free(a);
            exit(1);
        }
    }
    return a;
}

/*dealokacija matrice dimenzije nxn*/
void Dealociraj(int **a, int n)
{
    int i;
    for(i=0; i<n; i++)
        free(a[i]);
    free(a);
}

/*ispis matrice*/
void IspisiMatricu(int **a, int n)
{
    int i, j;
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%3d ",a[i][j]);
        printf("\n");
    }
}

/*provera da li je relacija predstavljena matricom refleksivna*/
int Refleksivna(int **a, int n)
{
    int i;
    for(i=0; i<n; i++)
        if(a[i][i]==0)
            return 0;
}
```

```

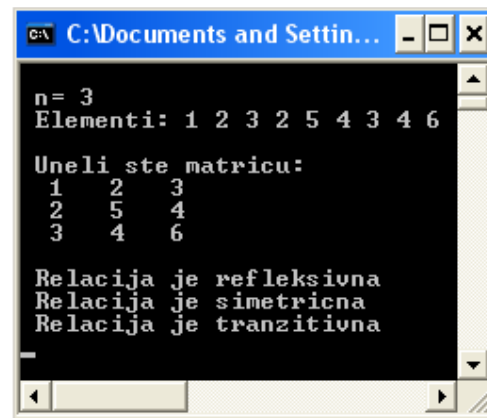
    return 1;
}

/*Provera da li je relacija predstavljena matricom simetricna*/
int Simetricna(int **a, int n)
{
    int i,j;
    for(i=0; i<n; i++)
        for(j=i+1; j<n; j++)
            if(a[i][j]!=a[j][i])
                return 0;
    return 1;
}

/*Provera da li je relacija predstavljena matricom tranzitivna*/
int Tranzitivna(int **a, int n)
{
    int i, j, k;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            for(k=0; k<n; k++)
                if((a[i][j]==1) && (a[j][k]==1) && (a[i][k]!=1))
                    return 0;
    return 1;
}

main()
{
    int **a, n, i, j;
    printf("\n n= ");
    scanf("%d",&n);
    a=Alociraj(n);
    printf(" Elementi: ");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d",&a[i][j]);
    printf("\n Uneli ste matricu:\n");
    IspisiMatricu(a,n);
    printf("\n");
    if(Refleksivna(a,n))
        printf(" Relacija je refleksivna\n");
    if(Simetricna(a,n))
        printf(" Relacija je simetricna\n");
    if(Tranzitivna(a,n))
        printf(" Relacija je tranzitivna\n");
    Deallociraj(a,n);
    getche();
    return 0;
}

```



Испис на екрану

3.24. Саставити програм који за квадратну матрицу целих бројева димензија **$n \times n$** одређује детерминанту преко Лапласовог развоја. Исписати матрицу и вредност детерминанте. Матрицу сместити у динамичку зону меморије.

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

/*funkcija alocira matricu dimenzije nxn*/
int **Alociraj(int n)
{
    int **a, i, k;
    a=(int**)malloc(n*sizeof(int*));
    if(a == NULL)
    {
        printf("GRESKA!\n");
        exit(1);
    }
    for(i=0; i<n; i++)
    {
        a[i]=malloc(n*sizeof(int));
        if(a[i] == NULL)
        {
            for(k=0; k<i; k++)
                free(a[k]);
            printf("GRESKA!\n");
            free(a);
            exit(1);
        }
    }
    return a;
}

/*funkcija dealocira matricu dimenzije nxn*/
void Dealociraj(int **a, int n)
{
    int i;
    for(i=0; i<n; i++)
        free(a[i]);
    free(a);
}

/*funkcija ucitava datu alociranu matricu sa standardnog ulaza*/
void UcitajMatricu(int **a, int n)
{
    int i,j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d",&a[i][j]);
}

/*funkcija vrsi ispis matrice*/
void IspisiMatricu(int **a, int n)
{
    int i,j;
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%3d",a[i][j]);
        printf("\n");
    }
}
```

```

}

/*rekurzivna funkcija koja vrsi Laplasov razvoj */
int Determinanta(int **matrica, int n)
{
    int **podmatrica, i, det=0, znak=1, vrsta, kolona;
    /*izlaz iz rekurzije je matrica 1x1*/
    if(n==1)
        return matrica[0][0];
    /*podmatrica ce da sadrzi minore polazne matrice*/
    podmatrica=Alociraj(n-1);
    for(i=0; i<n; i++)
    {
        for(kolona=0; kolona<i; kolona++)
            for(vrsta=1; vrsta<n; vrsta++)
                podmatrica[vrsta-1][kolona] = matrica[vrsta][kolona];
        for(kolona=i+1; kolona<n; kolona++)
            for(vrsta=1; vrsta<n; vrsta++)
                podmatrica[vrsta-1][kolona-1] = matrica[vrsta][kolona];
        det += znak*matrica[0][i]*Determinanta(podmatrica,n-1);
        znak*=-1;
    }
    Deallociraj(podmatrica,n-1);
    return det;
}

main()
{
    int **a, n;
    printf("\n n= ");
    scanf("%d", &n);
    a=Alociraj(n);
    printf(" Elementi: ");
    UcitajMatricu(a, n);
    printf(" Uneli ste matricu:\n");
    IspisiMatricu(a, n);
    printf("\n Determinanta= %d\n", Determinanta(a, n));
    Deallociraj(a, n);
    getch();
    return 0;
}

```

```

C:\Documents and Settings...
n= 3
Elementi: 1 2 3 4 5 6 7 8 0
Uneli ste matricu:
1 2 3
4 5 6
7 8 0

Determinanta= 27

```

Испис на екрану

4 СТРУКТУРЕ

4.1 Увод у структуре

4.1. Описати следеће блокове наредби:

а)

```
struct tacka
{
    int x;
    int y;
};
....
struct tacka
```

б)

```
struct tacka
{
    int x;
    int y;
} p;
```

в)

```
struct tacka
{
    int x;
    int y;
}p = {0, 0};
```

г)

```
typedef struct tacka
{
    int x;
    int y;
}TACKA;
....
TACKA p;
```

а) Извршена је дефиниција структуре имена (типа) `tacka` која има чланове `x` и `y`, оба типа `int`. Затим је у главном извршена декларација у којој променљива `p` постаје структура типа `tacka`.

б) Извршена је декларација унутар дефиниција структуре имена (типа) `tacka` која има чланове `x` и `y`, оба типа `int`. У оквиру дефиниције декларацијом променљива `p` постаје структура типа `tacka`.

в) Извршена је декларација и иницијализација унутар дефиниција структуре имена (типа) `tacka` која има чланове `x` и `y`, оба типа `int`. У оквиру дефиниције декларацијом променљива `p` постаје структура типа `tacka`., а иницијализацијом су додељене вредности 0 за чланове структуре.

г) Употребом **typedef** омогућено нам је да даље у програму уместо **struct tacka** користимо само `TACKA` приликом декларација променљивих која је типа `tacka`.

4.2. Ако је дата следећа декларација и дефиниција структуре:

```
struct osoba
{
    char prezime[15];
    char ime[15];
    int visina;
} student,
```

доделити у главном програму вредности члановима структуре.

```
student.prezime = "Markovic";
student.ime = "Marko";
student.visina = 190;
```

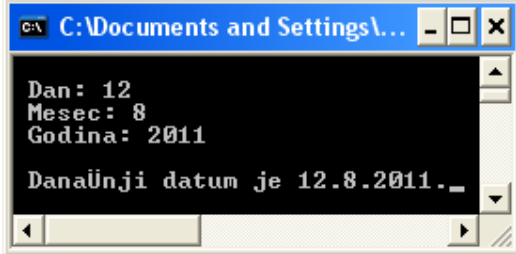
- 4.3. а) Саставити структуру која садржи дан, месец и годину, затим у главном програму користећи ову структуру унети податке о дану, месецу и години и исписати датум.
- б) Саставити структуру која садржи податке о раднику (име, презиме и месечна плата), затим у главном програму користећи ову структуру унети податке о раднику и исписати исте.

а)

```
#include <stdio.h>

struct datum
{
    int dan;
    int mesec;
    int godina;
};

main()
{
    int d, m, g;
    struct datum danas;
    printf("\n Dan: ");
    scanf("%d",&d);
    printf(" Mesec: ");
    scanf("%d",&m);
    printf(" Godina: ");
    scanf("%d",&g);
    danas.dan = d;
    danas.mesec = m;
    danas.godina = g;
    printf("\n Današnji datum je %d.%d.%d.",
           danas.dan, danas.mesec, danas.godina);
    getche();
    return 0;
}
```



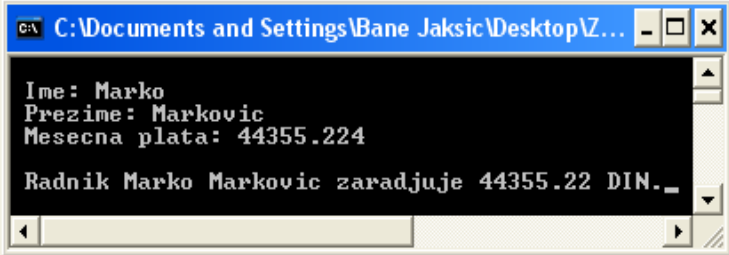
Испис на екрану

б)

```
#include <stdio.h>

struct licnost
{
    char ime[30];
    char prezime[30];
    float plata;
};

main()
{
    struct licnost radnik;
    printf("\n Ime: ");
    scanf("%s", &radnik.ime);
    printf(" Prezime: ");
    scanf("%s", &radnik.prezime);
    printf(" Mesecna plata: ");
    scanf("%f", &radnik.plata);
    printf("\n Radnik %s %s zaradjuje %.2f DIN.",
           radnik.ime, radnik.prezime, radnik.plata);
    getche();
    return 0;
}
```



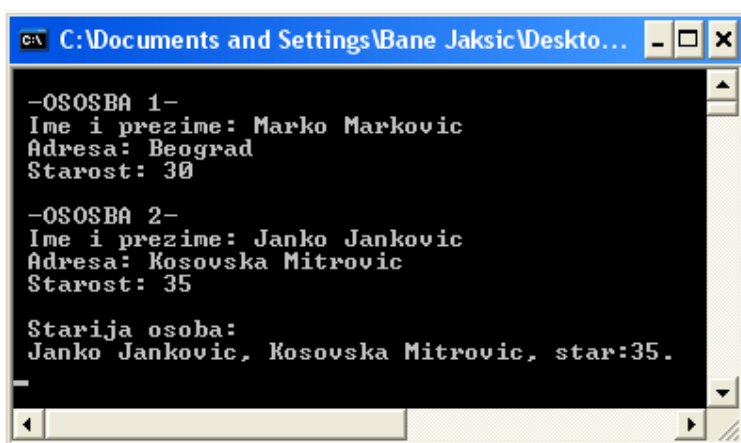
Испис на екрану

4.4. Саставити програм који учитава податке за две особе (име и презиме, адреса и старост у годинама), а затим исписује податке о старијој особи.

```
#include <stdio.h>

struct licnost
{
    char ime[30];
    char adresa[50];
    int starost;
};

main()
{
    struct licnost osobal, osoba2, s;
    printf("\n -OSOSBA 1-");
    printf("\n Ime i prezime: ");
    gets(osobal.ime);
    printf(" Adresa: ");
    gets(osobal.adresa);
    printf(" Starost: ");
    scanf("%d",&osobal.starost);
    while (getchar()!='\n'); /*ocistiti ulazni string iz scanf
                             jer je "nova linija" separator*/
    printf("\n -OSOSBA 2-");
    printf("\n Ime i prezime: ");
    gets(osoba2.ime);
    printf(" Adresa: ");
    gets(osoba2.adresa);
    printf(" Starost: ");
    scanf("%d",&osoba2.starost);
    if(osobal.starost > osoba2.starost)
        s=osobal;
    else s=osoba2;
    printf ("\n Starija osoba:\n" );
    printf(" %s, %s, star:%d.\n",s.ime,s.adresa,s.starost);
    getche();
    return 0;
}
```



Испис на екрану

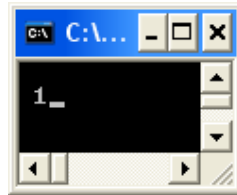
4.5. Који је резултат извршавања следећег програмског кода:

```
#include <stdio.h>

struct broj
{
    int x;
};

void fun(struct broj y)
{
    y.x=2;
}

main()
{
    struct broj z;
    z.x=1;
    fun(z);
    printf("\n %d", z.x);
    getch();
    return 0;
}
```



Испис на екрану

4.6. Саставити структуру за чување података о тачкама у равни, функцију за штампање тачке и функцију за израчунавање растојања између две тачке. Функције тестирати у главном програму.

први начин:

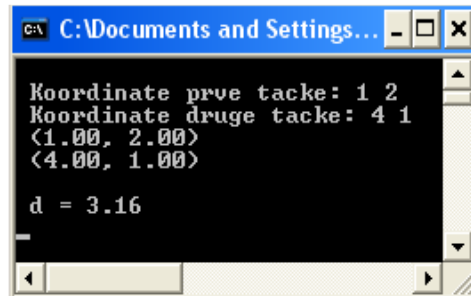
```
#include <stdio.h>
#include <math.h>

struct tacka
{
    float x;
    float y;
};

double Rastojanje(struct tacka t1, struct tacka t2)
{
    return sqrt((t1.x-t2.x)*(t1.x-t2.x) + (t1.y-t2.y)*(t1.y-t2.y));
}

void Odstampaj(struct tacka t)
{
    printf(" (%.2f, %.2f)\n", t.x, t.y);
}

main()
{
    struct tacka t1, t2;
    printf("\n Koordinate prve tacke: ");
    scanf("%f", &t1.x);
    scanf("%f", &t1.y);
    printf(" Koordinate druge tacke: ");
    scanf("%f", &t2.x);
    scanf("%f", &t2.y);
    Odstampaj(t1);
    Odstampaj(t2);
```



Испис на екрану

```

    printf("\n d = %.2f\n", Rastojanje(t1, t2));
    getch();
    return 0;
}

```

други начин (дефинисање типа структуре):

```

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

typedef struct tacka
{
    float x;
    float y;
} TACKA;

double Rastojanje(TACKA t1, TACKA t2)
{
    return sqrt((t1.x-t2.x)*(t1.x-t2.x) + (t1.y-t2.y)*(t1.y-t2.y));
}

void Odstampaj(TACKA t)
{
    printf(" (%.2f, %.2f)\n", t.x, t.y);
}

main()
{
    TACKA t1, t2;
    printf("\n Koordinate prve tacke: ");
    scanf("%f", &t1.x);
    scanf("%f", &t1.y);
    printf(" Koordinate druge tacke: ");
    scanf("%f", &t2.x);
    scanf("%f", &t2.y);
    Odstampaj(t1);
    Odstampaj(t2);
    printf("\n d = %.2f\n", Rastojanje(t1, t2));
    getch();
    return 0;
}

```

4.7. Датум је представљен са структуром од три члана: дан, месец и година. Саставити програм који користећи структуру за унети датум исписује сутрашњи датум.

```

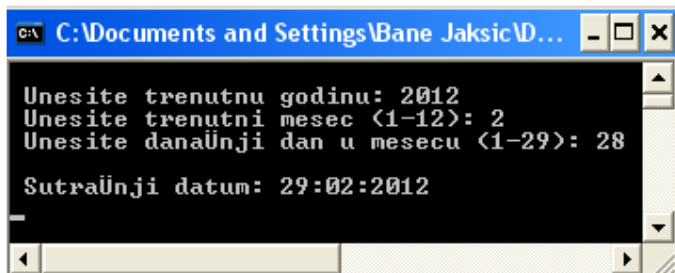
#include <stdio.h>

struct datum
{
    int dan, mesec, godina;
};

main()
{
    int brDana[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    struct datum danas, sutra;
    int ok=0;
    while(ok==0)
    {

```

```
printf("\n Unesite trenutnu godinu: ");
scanf("%d", &danaskodina);
if((danaskodina<=0))
    printf(" Pogrešna godina\n");
else
    ok = 1;
}
if((sutra.godina%4==0 && sutra.godina%100!=0 ||
    sutra.godina%400==0)==1)
    brDana[1]=29;
ok=0;
while(ok==0)
{
    printf(" Unesite trenutni mesec (1-12): ");
    scanf("%d", & danaskmesec);
    if((danaskmesec<1) || (danaskmesec>12))
        printf(" Pogrešan mesec\n");
    else
        ok=1;
}
ok=0;
while(ok==0)
{
    printf(" Unesite današnji dan u mesecu (1-%d): ",
        brDana[danaskmesec-1]);
    scanf("%d", &danaskdan);
    if((danaskdan<1) || (danaskdan>brDana[danaskmesec-1]))
        printf(" Pogrešan dan\n");
    else
        ok = 1;
}
sutra=danask;
sutra.dan++;
if(sutra.dan > brDana[sutra.mesec-1])
{
    sutra.dan=1;
    sutra.mesec++;
    if(sutra.mesec>12)
    {
        sutra.godina++;
        sutra.mesec=1;
    }
}
printf("\n Sutrašnji datum: %02d:%02d:%04d\n",
    sutra.dan, sutra.mesec, sutra.godina );
getche();
return 0;
}
```



Изпис на екрану

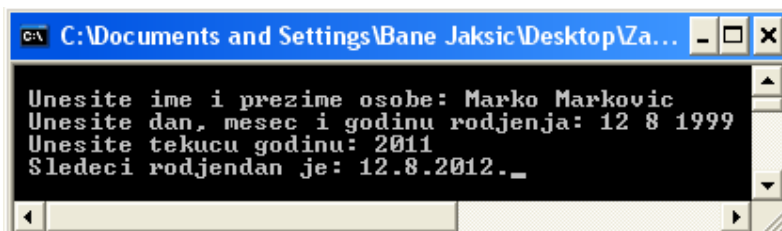
4.8. Шта се испишује на екрану након извршавања следећег програмског кода:

```
#include <stdio.h>

struct datum
{
    int dan;
    int mesec;
    int godina;
};

struct licnost
{
    char ime[30];
    struct datum datRodj;
};

main()
{
    struct licnost osoba;
    struct datum sledeciRodj;
    int tekGod;
    printf("\n Unesite ime i prezime osobe: ");
    gets(osoba.ime);
    printf(" Unesite dan, mesec i godinu rodjenja: ");
    scanf("%d %d %d", &osoba.datRodj.dan,
        &osoba.datRodj.mesec, &osoba.datRodj.godina);
    printf(" Unesite tekucu godinu: ");
    scanf("%u",&tekGod);
    sledeciRodj=osoba.datRodj;
    sledeciRodj.godina=tekGod+1;
    printf(" Sledeci rodjendan je: " );
    printf("%d.%d.%d.",
        sledeciRodj.dan, sledeciRodj.mesec, sledeciRodj.godina);
    getch();
    return 0;
}
```

*Испис на екрану***4.9.** Комплексни број је представљен структуром која садржи реални и имагинарни део броја. Саставити функције за израчунавање збира, разлике, производа и количника два комплексна броја. Затим саставити програм којим се тестирају претходне функције.

```

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

typedef struct kompl
{
    double re, im;
} KOMPL;

KOMPL Zbir (KOMPL a, KOMPL b)
{
    a.re += b.re;
    a.im += b.im;
    return a;
}

KOMPL Razlika (KOMPL a, KOMPL b)
{
    a.re -= b.re;
    a.im -= b.im;
    return a;
}

KOMPL Proizvod (KOMPL a, KOMPL b)
{
    KOMPL c;
    c.re = a.re * b.re - a.im * b.im;
    c.im = a.im * b.re + a.re * b.im;
    return c;
}

KOMPL Kolicnik (KOMPL a, KOMPL b)
{
    KOMPL c;
    double d;
    d = pow(b.re, 2) + pow(b.im, 2);
    c.re = (a.re*b.re + a.im*b.im)/d;
    c.im = (a.im*b.re - a.re*b.im)/d;
    return c;
}

main()
{
    KOMPL x, y, z;
    printf("\n Prvi broj (re,im): ");
    scanf("%lf%lf", &x.re, &x.im);
    printf(" Drugi broj (re,im): ");
    scanf("%lf%lf", &y.re, &y.im);
    printf("\n x    = (%.2f, %.2f)\n", x.re, x.im);
    printf(" y    = (%.2f, %.2f)\n", y.re, y.im);
    z=Zbir(x, y);
    printf(" x+y = (%.2f, %.2f)\n", z.re, z.im);
    z=Razlika(x, y);
    printf(" x-y = (%.2f, %.2f)\n", z.re, z.im);
    z=Proizvod(x, y);
    printf(" x*y = (%.2f, %.2f)\n", z.re, z.im);
    z=Kolicnik(x, y);
    printf(" x/y = (%.2f, %.2f)\n", z.re, z.im);
    getch();
    return 0;
}

```

```

C:\Documents and Se...
Prvi broj (re,im): 2 5
Drugi broj (re,im): 1 4

x    = (2.00, 5.00)
y    = (1.00, 4.00)
x+y  = (3.00, 9.00)
x-y  = (1.00, 1.00)
x*y  = (-18.00, 13.00)
x/y  = (1.29, -0.18)

```

Испис на екрану

4.2 Структуре и показивачи

4.10. Ако је дефиниција и декларација структуре дата на следећи начин:

```
struct datum
{
    int dan;
    int mesec;
    int godina;
};
...
struct datum *vreme;
```

како је могуће приступити елементима структуре и доделити им редом вредности: 12, 8 и 2011.

Решење:

```
(*vreme).dan=12;
(*vreme).mesec=8;
(*vreme).godina=2011;
```

или

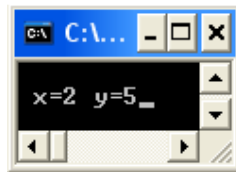
```
vreme->dan=12;
vreme->mesec=8;
vreme->godina=2011;
```

4.11. Које вредности се исписују након извршавања следећих програмских кодова:

а) показивач у структури

```
#include <stdio.h>
struct tacka
{
    int *p1;
    int *p2;
};

main()
{
    struct tacka A, *pok;
    int x, y;
    pok=&A;
    A.p1=&x;
    *A.p1=2;
    pok->p2=&y;
    *pok->p2 = 5;
    printf("\n x=%d y=%d", x,y);
    getch();
    return 0;
}
```



Испис на екрану

б) показивач на структуру

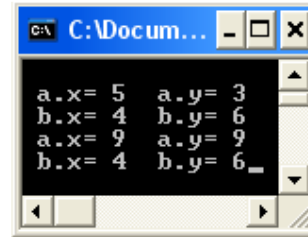
```
#include <stdio.h>
struct tacka
{
    int x;
    int y;
};

main()
{
    struct tacka a, b, *pa, *pb;
    pa=&a;
```

```

pb=&b;
(*pa).x = 5;
(*pa).y = 3;
pb->x = 4;
pb->y = 6;
printf("\n a.x= %d  a.y= %d", a.x, a.y);
printf("\n b.x= %d  b.y= %d", b.x, b.y);
a.x = (*pa).x + pb->x;
a.y = a.y + b.y;
printf("\n a.x= %d  a.y= %d", a.x, a.y);
printf("\n b.x= %d  b.y= %d", b.x, b.y);
getche();
return 0;
}

```



Испис на екрану

в) показивач на структуру

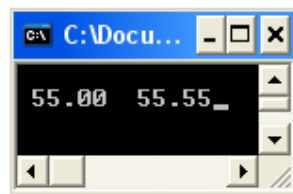
```

#include <stdio.h>

struct struktura
{
    float i;
    float f;
};

main()
{
    struct struktura a, *p;
    p=&a;
    a.i=55; a.f=55.55;
    printf("\n %.2f  %.2f", (*p).i, p->f);
    getche();
    return 0;
}

```



Испис на екрану

4.12. Која вредност се исписује након извршавања следећег програмског кода:

```

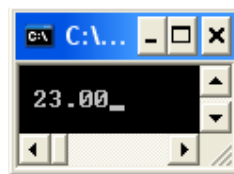
#include <stdio.h>

struct struktura
{
    char slovo;
    float z;
} S, *p;

void fun(struct struktura *y)
{
    y->slovo='A';
    y->z=23;
}

main()
{
    p=&S;
    S.z=11;
    p->slovo='B';
    fun (&S);
    printf("\n %.2f", S.z);
    getche(); return 0;
}

```

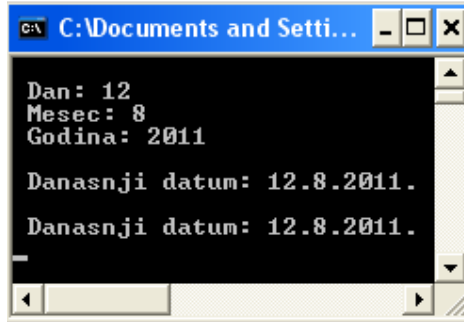


Испис на екрану

4.13. Саставити структуру која садржи податке о датуму (дан, месец и година), а затим користећи формирану структуру саставити главни програм који за унете податке о дану, месецу и години исписује датум. Приступ члановима структуре остварити преко показивача.

```
#include <stdio.h>
struct datum
{
    int dan, mesec, godina;
};

main()
{
    int d, m, g;
    struct datum Danas, *danas;
    printf("\n Dan: ");
    scanf("%d",&d);
    printf(" Mesec: ");
    scanf("%d",&m);
    printf(" Godina: ");
    scanf("%d",&g);
    danas = &Danas;
    danas->dan = d;
    danas->mesec = m;
    danas->godina = g;
    printf("\n Danasnji datum: %d.%d.%d.\n",
           danas->dan, danas->mesec, danas->godina);
    /*drugi nacin*/
    printf("\n Danasnji datum: %d.%d.%d.\n",
           Danas.dan, Danas.mesec, Danas.godina);
    getch();
    return 0;
}
```



Испис на екрану

4.14. Саставити функцију којим се учитавају подаци за особу (име и презиме, адреса и старост у годинама), а затим у главном програму користећи претходну функцију учитати податке за две особе и исписати податке о старијој особи.

```
#include <stdio.h>

struct licnost
{
    char ime[30];
    char adresa[50];
    unsigned starost;
};

void citaj(struct licnost *o)
{
    printf("\n Unesite ime: ");
    gets(o->ime);
    printf(" Unesite adresu: ");
    gets(o->adresa);
    printf(" Unesite starost: ");
    scanf("%u",&o->starost);
    while(getchar()!='\n');
}
```

```

main()
{
    struct licnost osobal, osoba2, *s;
    printf("\n -OSOBA 1-");
    citaj(&osobal);
    printf("\n - OSOBA 2-");
    citaj(&osoba2);
    if(osobal.starost > osoba2.starost)
        s=&osobal;
    else s=&osoba2;
    printf("\n Starija osoba:");
    printf("\n %s, %s, star: %u godina.\n",
        s->ime, s->adresa, s->starost );
    getche();
    return 0;
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak1...
-OSOBA 1-
Unesite ime: Marko Markovic
Unesite adresu: Beograd
Unesite starost: 20

- OSOBA 2-
Unesite ime: Janko Jankovic
Unesite adresu: Kosovska Mitrovica
Unesite starost: 26

Starija osoba:
Janko Jankovic, Kosovska Mitrovica, star: 26 godina.

```

Испис на екрану

4.15. Саставити програм који учитава структуру која садржи три координате вектора, а затим у главном програму одредити и исписати његов интензитет:

- без употребе показивача;
- употребом показивача.

a)

```

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

struct Vektor3D
{
    float x;
    float y;
    float z;
};

float Intenzitet(struct Vektor3D v)
{
    return sqrt(v.x*v.x+v.y*v.y+v.z*v.z);
}

main()
{

```

```

C:\Documents and Settings\Ban...
Unesite x koordiantu vektora: 3
Unesite y koordinatu vektora: 2
Unesite z koordinatu vektora: 1

Inetnizitet vekitra je 3.741657_

```

Испис на екрану

```

    struct Vektor3D vektor;
    printf("\n Unesite x koordiantu vektora: ");
    scanf("%f",&(vektor.x));
    printf(" Unesite y koordinatu vektora: ");
    scanf("%f",&(vektor.y));
    printf(" Unesite z koordinatu vektora: ");
    scanf("%f",&(vektor.z));
    printf("\n Inetnzitet vekitra je %f", Intenzitet(vektor));
    getch();
    return 0;
}

```

6)

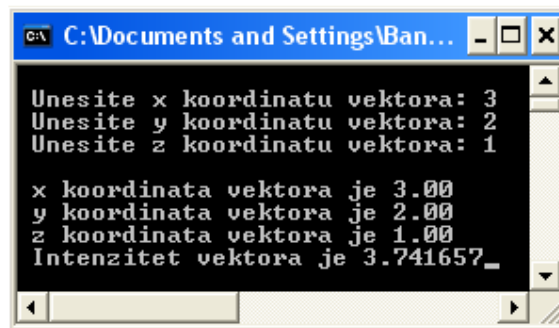
```

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

struct Vektor3D
{
    float x;
    float y;
    float z;
};

main()
{
    struct Vektor3D vektor,*pv;
    printf("\n Unesite x koordinatu vektora: ");
    scanf("%f",&(vektor.x));
    printf(" Unesite y koordinatu vektora: ");
    scanf("%f",&(vektor.y));
    printf(" Unesite z koordinatu vektora: ");
    scanf("%f",&(vektor.z));
    pv=&vektor;
    printf("\n x koordinata vektora je %.2f",pv->x);
    printf("\n y koordinata vektora je %.2f",pv->y);
    printf("\n z koordinata vektora je %.2f",pv->z);
    printf("\n Intenzitet vektora je %f",
        sqrt(pv->x*pv->x + pv->y*pv->y + pv->z*pv->z));
    getch();
    return 0;
}

```

*Испис на екрану*

4.16. Саставити главни функцију која проверава да ли се задата тачка налази унутар задатог круга. Параметри круга су координате центра и полупречник. Затим саставити програм који тестира претходну функцију.

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

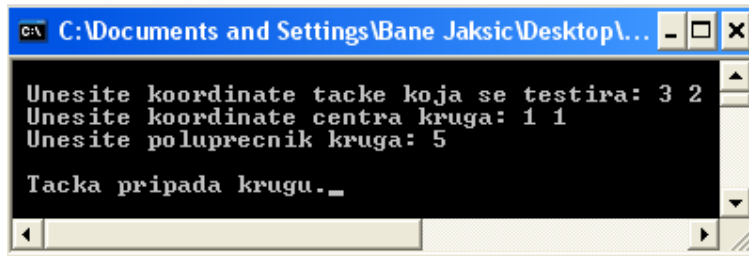
struct tacka
{
    double x;
    double y;
};

struct krug
{
    struct tacka o;
    double r;
};

double Rastojanje(struct tacka *a, struct tacka *b)
{
    return sqrt((a->x - b->x)*(a->x - b->x) +
                (a->y - b->y)*(a->y - b->y));
}

int PripadaKругу(struct tacka *t, struct krug *k)
{
    if(Rastojanje(t, &k->o) <= k->r)
        return 1;
    else
        return 0;
}

main()
{
    struct tacka t;
    struct krug k;
    printf("\n Unesite koordinate tacke koja se testira: ");
    scanf("%lf%lf", &t.x, &t.y);
    printf(" Unesite koordinate centra kruga: ");
    scanf("%lf%lf", &k.o.x, &k.o.y);
    printf(" Unesite poluprecnik kruga: ");
    scanf("%lf", &k.r);
    if(PripadaKругу(&t, &k))
        printf("\n Tacka pripada krugu.");
    else
        printf("\n Tacka ne pripada krugu.");
    getch();
    return 0;
}
```



Истис на екрану

4.17. Саставити главни функцију која проверава да ли се задата тачка налази унутар задатог правоугаоника. Функција треба да врати вредност различиту од нуле уколико се тачка налази унутар задатог правоугаоника. Затим саставити програм који тестира претходну функцију. Правоугаоник се налази у првом квадранту координатног система. Тачке правоугаоника које се задају су доња лева и горња десна.

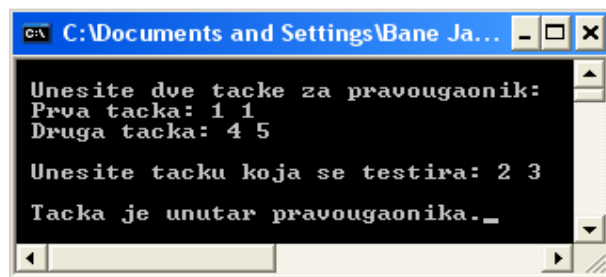
```
#include <stdio.h>

struct tacka
{
    int x;
    int y;
};

struct Pravougaonik
{
    struct tacka doleLevo;
    struct tacka goreDesno;
};

int Proveri(struct Pravougaonik Pr, struct tacka *t)
{
    return(t->x > Pr.doleLevo.x && t->x < Pr.goreDesno.x &&
           t->y > Pr.doleLevo.y && t->y < Pr.goreDesno.y);
}

main()
{
    struct tacka tt;
    int x1, y1, x2, y2;
    struct Pravougaonik prvg;
    printf("\n Unesite dve tacke za pravougaonik:\n");
    printf(" Prva tacka: ");
    scanf("%d %d", &x1, &y1);
    printf(" Druga tacka: ");
    scanf("%d %d", &x2, &y2);
    prvg.doleLevo.x = x1;
    prvg.doleLevo.y = y1;
    prvg.goreDesno.x = x2;
    prvg.goreDesno.y = y2;
    printf("\n Unesite tacku koja se testira: ");
    scanf("%d %d", &x1, &y1);
    tt.x = x1;
    tt.y = y1;
    if(Proveri(prvg,&tt))
        printf("\n Tacka je unutar pravougaonika.");
    else
        printf("\n Tacka je van pravougaonika.");
    getche();
    return 0;
}
```



Истис на екрану

4.18. Полином је представљен структуром која садржи ред полинома и низ коефицијената. Саставити функције за израчунавање збира, разлике, производа и количника два полинома који су задати помоћу коефицијената. Саставити и функције које читају и исписују полиноме. Затим саставити главни програм који тестира претходне функције.

```
#include <stdio.h>

typedef struct poli
{
    double a[21];
    int n;
} POLI;

POLI Zbir(POLI p1, POLI p2)
{
    POLI p; int i;
    p.n=(p1.n > p2.n) ? p1.n : p2.n;
    for(i=0; i<=p.n; i++)
        if(i>p2.n) p.a[i]=p1.a[i];
        else if(i>p1.n) p.a[i]=p2.a[i];
        else p.a[i]=p1.a[i]+p2.a[i];
    while(p.n>0 && p.a[p.n]==0)
        p.n--;
    return p;
}

POLI Razlika(POLI p1, POLI p2)
{
    POLI p; int i;
    p.n=(p1.n > p2.n) ? p1.n : p2.n;
    for(i=0; i<=p.n; i++)
        if(i>p2.n) p.a[i]=p1.a[i];
        else if(i > p1.n) p.a[i]=-p2.a[i];
        else p.a[i]=p1.a[i]-p2.a[i];
    while(p.n>0 && p.a[p.n]==0)
        p.n--;
    return p;
}

POLI Proizvod(POLI p1, POLI p2)
{
    POLI p; int i, j;
    p.n=(p1.n<0 || p2.n<0) ? -1 : p1.n+p2.n;
    for(i=0; i<=p.n; p.a[i++]=0);
    for(i=0; i<=p1.n; i++)
        for(j=0; j<=p2.n; j++)
            p.a[i+j] += p1.a[i]*p2.a[j];
    return p;
}

POLI Kolicnik (POLI p1, POLI p2, POLI *ostatak)
{
    POLI p; int i, j;
    p.n=p1.n-p2.n;
    for(i=p.n; i>=0; i--)
    {
        p.a[i]=p1.a[p2.n+i]/p2.a[p2.n];
        for(j=0; j<=p2.n; j++)
            p1.a[j+i] -= p2.a[j]*p.a[i];
    }
}
```

```

    while(p1.n>=0 && p1.a[p1.n]==0)
        p1.n--;
    *ostatak = p1;
    return p;
}

POLI Citaj()
{
    POLI p;
    int i;
    printf("\n Red polinoma: ");
    scanf("%d", &p.n);
    printf(" Koeficijenti: ");
    for(i=p.n; i>=0; i--)
        scanf("%lf", &p.a[i]);
    return p;
}

void Pisi(POLI p)
{
    int i;
    for(i=p.n; i>=0; i--)
        printf("%.2f, ", p.a[i]);
}

main ()
{
    POLI p1, p2, p3;
    printf("\n -PRVI POLINOM-");
    p1=Citaj();
    printf("\n -DRUGI POLINOM-");
    p2=Citaj ();
    printf("\n P1      = ");
    Pisi(p1);
    printf("\n P2      = ");
    Pisi(p2);
    printf("\n P1+P2 = ");
    p3=Zbir(p1, p2);
    Pisi(p3);
    printf("\n P1-P2 = ");
    Pisi(Razlika(p1,p2));
    printf("\n P1*P2 = ");
    Pisi(Proizvod(p1,p2));
    printf("\n P1/P2 = ");
    Pisi(Kolicnik(p1,p2,&p3));
    printf("\n P1%%P2 = ");
    Pisi(p3);
    printf("\n");
    getch();
    return 0;
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop...
-PRVI POLINOM-
Red polinoma: 3
Koeficijenti: 2 1 3 1

-DRUGI POLINOM-
Red polinoma: 2
Koeficijenti: 1 1 0

P1      = 2.00, 1.00, 3.00, 1.00,
P2      = 1.00, 1.00, 0.00,
P1+P2   = 2.00, 2.00, 4.00, 1.00,
P1-P2   = 2.00, 0.00, 2.00, 1.00,
P1*P2   = 2.00, 3.00, 4.00, 4.00, 1.00, 0.00,
P1/P2   = 2.00, -1.00,
P1%%P2  = 4.00, 1.00,

```

Истис на екрану

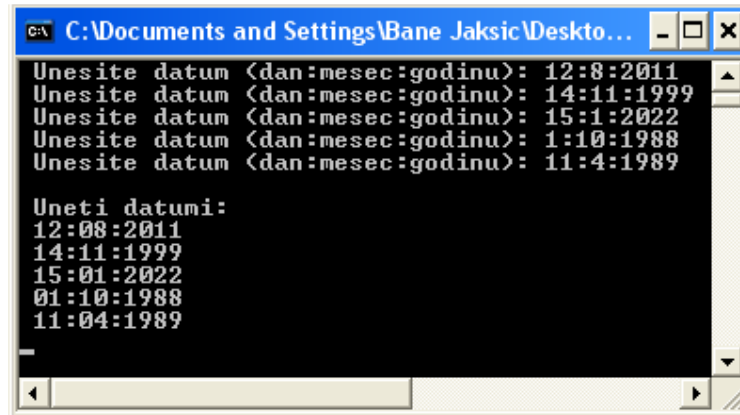
4.3 Низови структура

4.19. Саставити структуру која садржи податке о датуму (дан, месец и година), а затим користећи формирану структуру саставити главни програм који уноси податке о пет датума и испишује исте.

```
#include <stdio.h>

struct datum
{
    int dan, mesec, godina;
};

main()
{
    struct datum niz[5];
    int i;
    for(i=0; i<5; i++)
    {
        printf(" Unesite datum (dan:mesec:godinu): ");
        scanf("%d:%d:%d", &niz[i].dan, &niz[i].mesec, &niz[i].godina );
    }
    printf("\n Uneti datumi:\n");
    for(i=0; i<5; i++)
        printf(" %02d:%02d:%04d\n",
            niz[i].dan, niz[i].mesec, niz[i].godina );
    getch();
    return 0;
}
```



Испис на екрану

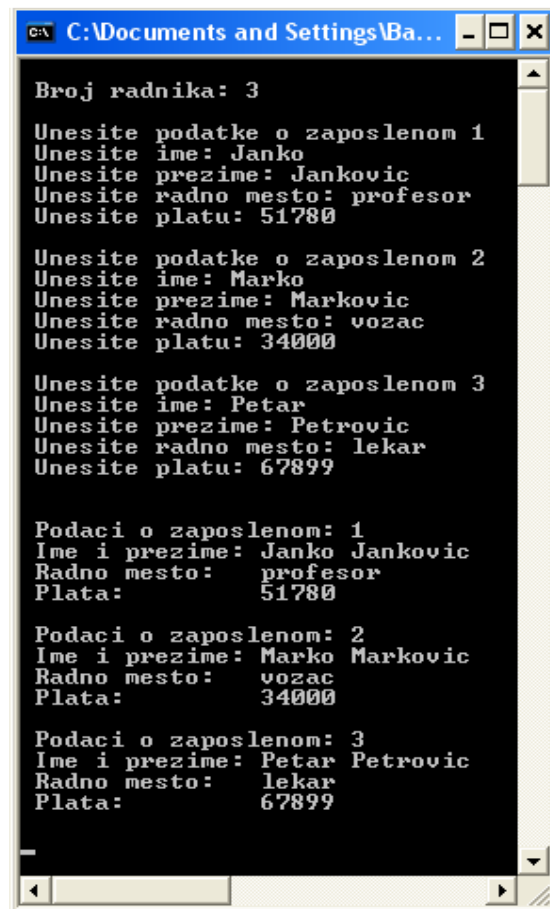
4.20. Саставити програм који учитава и табеларно испишује податке о **n** запослена радника (име и презиме, радно место, плата).

```
#include <stdio.h>
#define MAX 50
#define MAXR 100

typedef struct zaposleni
{
    char ime[MAX];
    char prezime[MAX];
    char radnomesto[MAX];
    int plata;
```



```
} ZAPOSLENI;  
  
main()  
{  
    int i, n;  
    ZAPOSLENI zp[MAXR];  
    printf("\n Broj radnika: ");  
    scanf("%d", &n);  
    for(i=0; i<n; i++)  
    {  
        printf("\n Unesite podatke o zaposlenom %d\n", i+1);  
        printf(" Unesite ime: ");  
        scanf("%s", &zp[i].ime);  
        printf(" Unesite prezime: ");  
        scanf("%s", &zp[i].prezime);  
        printf(" Unesite radno mesto: ");  
        scanf("%s", &zp[i].radnomesto);  
        printf(" Unesite platu: ");  
        scanf("%d", &zp[i].plata);  
    }  
    printf("\n\n");  
    for(i=0; i<n; i++)  
    {  
        printf(" Podaci o zaposlenom: %d\n", i+1);  
        printf(" Ime i prezime: \t%s %s\n", zp[i].ime, zp[i].prezime);  
        printf(" Radno mesto: \t%s\n", zp[i].radnomesto);  
        printf(" Plata: \t%d\n\n", zp[i].plata);  
    }  
    getch();  
    return 0;  
}
```



```
CA C:\Documents and Settings\Ba... - _ X  
Broj radnika: 3  
  
Unesite podatke o zaposlenom 1  
Unesite ime: Janko  
Unesite prezime: Jankovic  
Unesite radno mesto: profesor  
Unesite platu: 51780  
  
Unesite podatke o zaposlenom 2  
Unesite ime: Marko  
Unesite prezime: Markovic  
Unesite radno mesto: vazac  
Unesite platu: 34000  
  
Unesite podatke o zaposlenom 3  
Unesite ime: Petar  
Unesite prezime: Petrovic  
Unesite radno mesto: lekar  
Unesite platu: 67899  
  
Podaci o zaposlenom: 1  
Ime i prezime: Janko Jankovic  
Radno mesto: profesor  
Plata: 51780  
  
Podaci o zaposlenom: 2  
Ime i prezime: Marko Markovic  
Radno mesto: vazac  
Plata: 34000  
  
Podaci o zaposlenom: 3  
Ime i prezime: Petar Petrovic  
Radno mesto: lekar  
Plata: 67899
```

Испис на екрану

4.21. Саставити програм који омогућава унос тачака са стандардног улаза и израчунава квадрате растојања свих тачака од координатног почетка, збир квадрата растојања и тачку која је најдаља од координатног почетка. Резултате исписати на екрану.

```
#include <stdio.h>
#define MAX 100

typedef struct tacka
{
    int x, y;
} TACKA;

main()
{
    TACKA Tacke[MAX];
    int i, n, pozicija=0, maxRastojanje=0, kvRastojanje, zbir=0;
    printf("\n Unesite broj tacaka (manji od %d): ", MAX);
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        printf(" Unesite x-koordinatu %d. tacke: ", i+1);
        scanf("%d", &Tacke[i].x);
        printf(" Unesite y-koordinatu %d. tacke: ", i+1);
        scanf("%d", &Tacke[i].y);
    }
    for(i=0; i<n; i++)
    {
        kvRastojanje = Tacke[i].x*Tacke[i].x + Tacke[i].y*Tacke[i].y;
        printf("\n Kvadrat rastojanja %d. tacke je %d", i+1, kvRastojanje);
        if(kvRastojanje > maxRastojanje)
        {
            maxRastojanje = kvRastojanje;
            pozicija = i;
        }
        zbir += kvRastojanje;
    }
    printf("\n Zbir kvadrata rastojanja je %d", zbir);
    printf("\n Najdalja tacka od koordinatnog pocetka je tacka (%d,%d)",
        Tacke[pozicija].x, Tacke[pozicija].y);
    getch();
    return 0;
}
```

```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak21.exe
Unesite broj tacaka (manji od 100): 4
Unesite x-koordinatu 1. tacke: 1
Unesite y-koordinatu 1. tacke: 1
Unesite x-koordinatu 2. tacke: 2
Unesite y-koordinatu 2. tacke: 0
Unesite x-koordinatu 3. tacke: 2
Unesite y-koordinatu 3. tacke: 2
Unesite x-koordinatu 4. tacke: 3
Unesite y-koordinatu 4. tacke: 1

Kvadrat rastojanja 1. tacke je 2
Kvadrat rastojanja 2. tacke je 4
Kvadrat rastojanja 3. tacke je 8
Kvadrat rastojanja 4. tacke je 10
Zbir kvadrata rastojanja je 24
Najdalja tacka od koordinatnog pocetka je tacka (3,1)_
```

Испис на екрану

4.22. Саставити функцију која учитава податке о особи (име и презиме, адреса, старост), а затим у главном програму користећи претходну функцију уноси податке о **n** особа и штампа те податке.

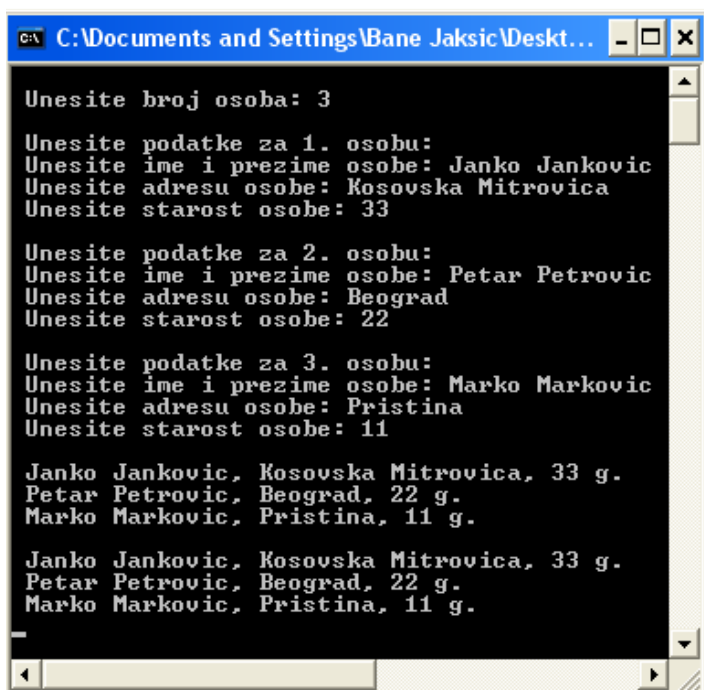
```
#include <stdio.h>
#define MAX 100

typedef struct licnost
{
    char ime[30];
    char adresa[50];
    int starost;
} LICNOST;

void Citaj(LICNOST *o)
{
    printf(" Unesite ime i prezime osobe: ");
    gets(o->ime);
    printf(" Unesite adresu osobe: ");
    gets(o->adresa);
    printf(" Unesite starost osobe: ");
    scanf("%d",&o->starost);
    while(getchar()!='\n');
}

main()
{
    int i,n;
    LICNOST osoba[MAX], *p;
    printf("\n Unesite broj osoba: ");
    scanf("%d",&n);
    while(getchar()!='\n');
    for(i=0;i<n;i++)
    {
        printf("\n Unesite podatke za %d. osobu:\n",i+1);
        Citaj(&osoba[i]);
    }
    printf("\n");
    /*prvi nacin za ispis ucitanih vrednosti*/
    for(i=0; i<n; i++)
        printf(" %s, %s, %d g.\n",
               osoba[i].ime, osoba[i].adresa, osoba[i].starost);
    printf("\n");
    /*drugi nacin ispisa koriscenjem pokazivaca*/
    for(p=osoba; p<osoba+n; p++)
        printf(" %s, %s, %d g.\n",p->ime, p->adresa, p->starost);

    getche();
    return 0;
}
```



Испис на екрану

4.23. Саставити програм који учитава податке о **n** MP3 песама (назив, извођач, албум, година издања), а затим исписује унете песме и приказује број песама издатих у 2011. години.

```

#include <stdio.h>
#define MAX 100

typedef struct pesme
{
    char naziv[30];
    char izvodjac[30];
    char album[30];
    int godizdanja;
} PESME;

void Citaj(PESME *p)
{
    printf(" Unesite naziv pesme: ");
    gets(p->naziv);
    printf(" Unesite ime izvodjaca: ");
    gets(p->izvodjac);
    printf(" Unesite ime albuma: ");
    gets(p->album);
    printf(" Unesite godinu izdavanja: ");
    scanf("%u",&p->godizdanja);
    while(getchar()!='\n');
}

main()
{
    PESME Pesma[MAX];
    int i, n, s=0;
    printf("\n Unesite broj pesama: ");

```

```
scanf("%d",&n);
while(getchar()!='\n');
for(i=0;i<n;i++)
{
    printf("\n Unesite podatke za %d. pesmu:\n",i+1);
    Citaj(&Pesma[i]);
}
printf("\n\n Završen unos podataka...\nPritisnite neki taster...");
getche();
for(i=0;i<n;i++)
{
    printf("%s, %s, %s, %d\n",Pesma[i].naziv, Pesma[i].izvodjac,
        Pesma[i].album, Pesma[i].godizdanja);
    if(Pesma[i].godizdanja==2011) s++;
}
printf("\n\n Pesama izdatih u 2011. je: %d", s);
getche();
return 0;
}
```

```
C:\Documents and Settings\Bane Jaksic...
Unesite broj pesama: 4

Unesite podatke za 1. pesmu:
Unesite naziv pesme: Stars
Unesite ime izvodjaca: Ti Pi Cal
Unesite ime albuma: Hits CD1
Unesite godinu izdavanja: 2011

Unesite podatke za 2. pesmu:
Unesite naziv pesme: Like I Love You
Unesite ime izvodjaca: RIO
Unesite ime albuma: Hits 2
Unesite godinu izdavanja: 2003

Unesite podatke za 3. pesmu:
Unesite naziv pesme: Sun Is Up
Unesite ime izvodjaca: Inna
Unesite ime albuma: Dance Hits
Unesite godinu izdavanja: 2011

Unesite podatke za 4. pesmu:
Unesite naziv pesme: Live Tonight
Unesite ime izvodjaca: Basto
Unesite ime albuma: Basto
Unesite godinu izdavanja: 2010

Završen unos podataka...
Stars, Ti Pi Cal, Hits CD1, 2011
Like I Love You, RIO, Hits 2, 2003
Sun Is Up, Inna, Dance Hits, 2011
Live Tonight, Basto, Basto, 2010

Pesama izdatih u 2011. je: 2_
```

Испис на екрану

4.24. Саставити програм за испис свих студената који имају натпросечну оцену и мање од 5 испита пренетих у наредну годину. Подаци који се уносе о студентима су: име, презиме, број индекса, просек, број пренетих испита.

```
#include <stdio.h>
#define MAX 100

typedef struct studenti
{
    char ime[15];
    char prezime[20];
    char brojIndexa[10];
    int brPrenetih;
    float prosek;
} STUDENTI;

main()
{
    STUDENTI Student[MAX];
    int i, n;
    float prosek=0;
    printf("\n Unesite broj studenata: ");
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        printf("\n PODACI O STUDENTU %d:\n", i+1);
        printf(" Unesite ime i prezime studenta: ");
        scanf("%s %s",&Student[i].ime, &Student[i].prezime);
        printf(" Unesite broj indeksa: ");
        scanf("%s",&Student[i].brojIndexa);
        printf(" Unesite broj prenetih ispita: ");
        scanf("%d",&Student[i].brPrenetih);
        printf(" Unesite prosek: ");
        scanf("%f",&Student[i].prosek);
    }
    printf("\n IZABRANI STUDENTI:\n");
    for(i=0; i<n; i++)
    {
        if(Student[i].prosek>prosek)
            if(Student[i].brPrenetih<5)
                printf(" %s %s %s\n", Student[i].ime,
                    Student[i].prezime, Student[i].brojIndexa);
    }
    getch();
    return 0;
}
```

```

C:\Documents and Settings\Bane Jaksic\Desktop\Za...
Unesite broj studenata: 4

PODACI O STUDENTU 1:
Unesite ime i prezime studenta: Marko Markovic
Unesite broj indeksa: 12/09
Unesite broj prenetih ispita: 3
Unesite prosek: 8.9

PODACI O STUDENTU 2:
Unesite ime i prezime studenta: Janko Jankovic
Unesite broj indeksa: 23/11
Unesite broj prenetih ispita: 5
Unesite prosek: 7.5

PODACI O STUDENTU 3:
Unesite ime i prezime studenta: Petar Petrovic
Unesite broj indeksa: 1/09
Unesite broj prenetih ispita: 2
Unesite prosek: 9.11

PODACI O STUDENTU 4:
Unesite ime i prezime studenta: Milos Milosevic
Unesite broj indeksa: 21/10
Unesite broj prenetih ispita: 8
Unesite prosek: 6.2

IZABRANI STUDENTI:
Marko Markovic 12/09
Petar Petrovic 1/09

```

Испис на екрану

4.25. Саставити програм којим се учитавају подаци за n радника, и испушују подаци за радника који има највећу плату. Подаци о радницима су: име, презиме и коефицијент. Плата радника добија се по формули: $\text{плата} = \text{цена рада} * \text{коефицијент}$. Цена рада једнака је за све раднике и уноси се на почетку.

```

#include <stdio.h>
#define MAX 100

typedef struct radnici
{
    char prezime[35];
    char ime[20];
    float koeficijent;
    float plata;
} RADNICI;

main()
{
    RADNICI Radnik[MAX];
    int n, i, rbr=0;
    float cenaRada, max;
    printf("\n Unesite broj radnika: ");
    scanf("%d",&n);
    printf(" Unesite cenu rada: ");
    scanf("%f",&cenaRada);
    for(i=0; i<n; i++)
    {
        printf("\n PODACI ZA %d. RADNIKA:\n", i+1);
    }
}

```

```

while(getchar()!='\n');
printf(" Prezime: ");
gets(Radnik[i].prezime);
printf(" Ime: ");
gets(Radnik[i].ime);
printf(" Koeficijent: ");
scanf("%f",&Radnik[i].koeficijent);
Radnik[i].plata=
    cenaRada*Radnik[i].koeficijent;
printf(" Plata: %.2f\n", Radnik[i].plata);
}
max=Radnik[0].plata;
for(i=1; i<n; i++)
{
    if(max<Radnik[i].plata)
    {
        max=Radnik[i].plata;
        rbr=i;
    }
}
printf("\n\n RADNIK SA NAJVISOM PLATOM:\n");
printf(" Prezime: %s\n", Radnik[rbr].prezime);
printf(" Ime: %s\n", Radnik[rbr].ime);
printf(" Koeficijent: %.2f\n",
    Radnik[rbr].koeficijent);
printf(" Plata: %.2f\n", Radnik[rbr].plata);
getche();
return 0;
}

```



```

C:\Documents and Settings...
Unesite broj radnika: 4
Unesite cenu rada: 1623

PODACI ZA 1. RADNIKA:
Prezime: Jankovic
Ime: Janko
Koeficijent: 23.21
Plata: 37669.83

PODACI ZA 2. RADNIKA:
Prezime: Markovic
Ime: Marko
Koeficijent: 11.29
Plata: 18323.67

PODACI ZA 3. RADNIKA:
Prezime: Milosevic
Ime: Milos
Koeficijent: 39.99
Plata: 64903.77

PODACI ZA 4. RADNIKA:
Prezime: Petrovic
Ime: Petar
Koeficijent: 19.29
Plata: 31307.67

RADNIK SA NAJVISOM PLATOM:
Prezime: Milosevic
Ime: Milos
Koeficijent: 39.99
Plata: 64903.77

```

Испис на екрану

4.26. Структура **radnik** садржи податке о имену, презимену, броју радних сати и плати радника. Саставити програм за повећање плате **n** радника према проведеном радном времену на послу. Месец има 160 радних сати. За прековремени рад од 20 сати месечно повишица је 15%, за прековремени рад од 30 сати је 25%. Исписати списак радника са свим подацима који добијају повишицу.

```

#include <stdio.h>
#define MAX 100
#define RADNISATI 160

typedef struct radnici
{
    char ime[15];
    char prezime[25];
    int brojRadnihSati;
    double plata;
}RADNICI;

/*Funkcija za unos radnika sa svim podacima*/
int unosRadnika(RADNICI *Radnik, int i)
{
    printf("\n UNESITE PODATKE O RADNIKU BR. %d:\n", i);
    printf(" Ime: ");

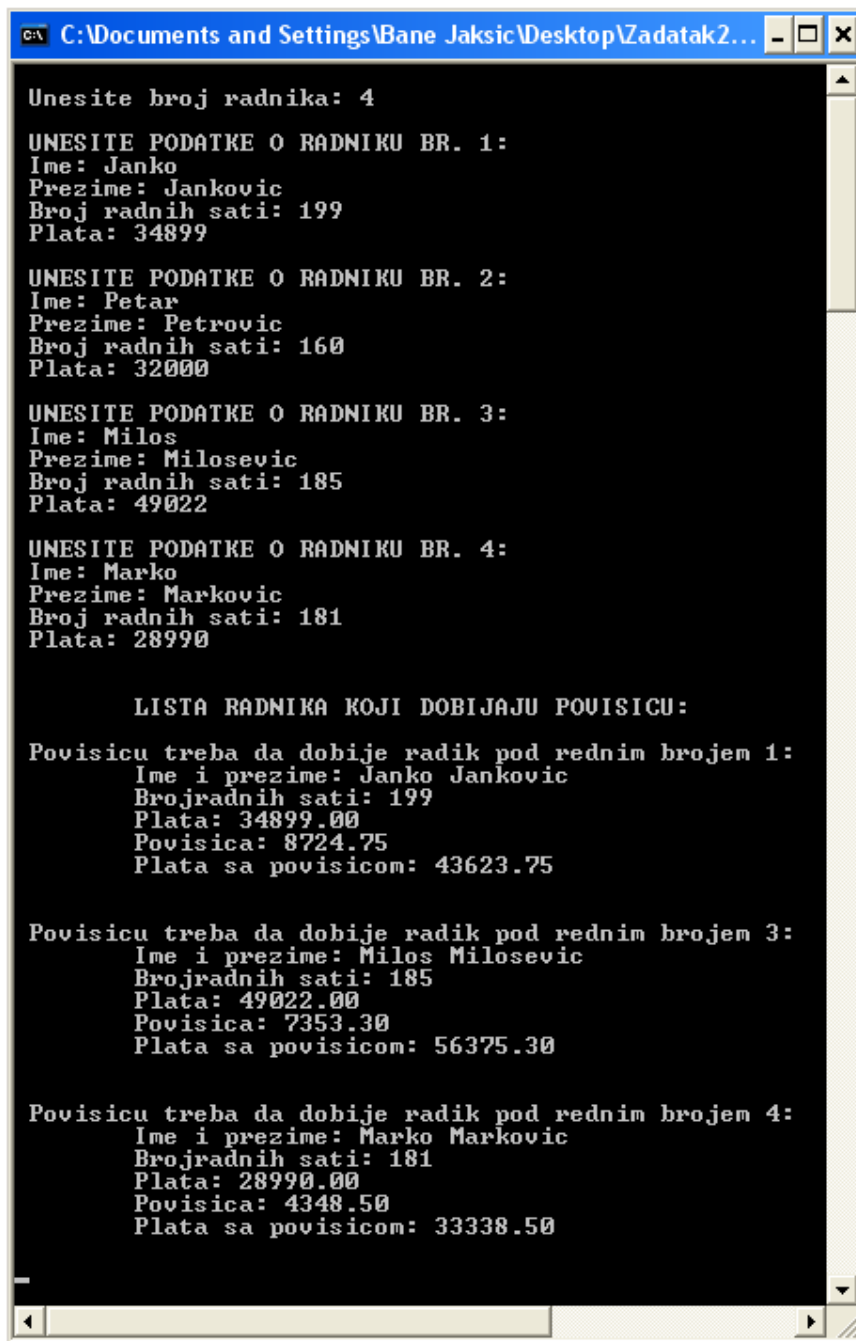
```



```
    gets(Radnik->ime);
    printf(" Prezime: ");
    gets(Radnik->prezime);
    printf(" Broj radnih sati: ");
    scanf("%d",&Radnik->brojRadnihSati);
    printf(" Plata: ");
    scanf("%lf",&Radnik->plata);
    if(Radnik->brojRadnihSati > RADNISATI+19)
        return 1;
    else
        return 0;
}

/*Funkcija za stampanje radnika koji su dobili povisicu*/
void stampaRadnika(RADNICI *Radnik, int i)
{
    double povisica=0.0;
    if(Radnik->brojRadnihSati > RADNISATI + 19)
    {
        if(Radnik->brojRadnihSati-RADNISATI >=30)
            povisica=(Radnik->plata*25)/100;
        else
            povisica=(Radnik->plata*15)/100;
        printf("\n Povisicu treba da dobije radik pod rednim brojem
            %d:\n",i+1);
        printf("\tIme i prezime: %s %s\n\tBrojradnih sati: %d\n",
            Radnik->ime, Radnik->prezime, Radnik->brojRadnihSati);
        printf("\tPlata: %.2lf\n",Radnik->plata);
        printf("\tPovisica: %.2lf\n",povisica);
        printf("\tPlata sa povisicom: %.2lf\n\n",Radnik->plata+povisica);
    }
}

main()
{
    int n=0, i, brojRadnikaSaPovisicom=0;
    RADNICI Radnik[MAX];
    printf("\n Unesite broj radnika: ");
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        while(getchar()!='\n');
        brojRadnikaSaPovisicom+=unosRadnika(&Radnik[i],i+1);
    }
    if(brojRadnikaSaPovisicom!=0)
    {
        printf("\n\n\tLISTA RADNIKA KOJI DOBIJAJU POVISICU:\n");
        for(i=0;i<n;i++)
            stampaRadnika(&Radnik[i],i);
    }
    else
        printf("\n NI JEDAN RADNIK NIJE ZADOVOLJIO KRITERIJUM
            ZA POVISICU!\n");
    getche();
    return 0;
}
```



```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak2...
Unesite broj radnika: 4

UNESITE PODATKE O RADNIKU BR. 1:
Ime: Janko
Prezime: Jankovic
Broj radnih sati: 199
Plata: 34899

UNESITE PODATKE O RADNIKU BR. 2:
Ime: Petar
Prezime: Petrovic
Broj radnih sati: 160
Plata: 32000

UNESITE PODATKE O RADNIKU BR. 3:
Ime: Milos
Prezime: Milosevic
Broj radnih sati: 185
Plata: 49022

UNESITE PODATKE O RADNIKU BR. 4:
Ime: Marko
Prezime: Markovic
Broj radnih sati: 181
Plata: 28990

LISTA RADNIKA KOJI DOBIJAJU POVISICU:

Povisicu treba da dobije radnik pod rednim brojem 1:
Ime i prezime: Janko Jankovic
Brojradnih sati: 199
Plata: 34899.00
Povisica: 8724.75
Plata sa povisicom: 43623.75

Povisicu treba da dobije radnik pod rednim brojem 3:
Ime i prezime: Milos Milosevic
Brojradnih sati: 185
Plata: 49022.00
Povisica: 7353.30
Plata sa povisicom: 56375.30

Povisicu treba da dobije radnik pod rednim brojem 4:
Ime i prezime: Marko Markovic
Brojradnih sati: 181
Plata: 28990.00
Povisica: 4348.50
Plata sa povisicom: 33338.50
```

Испис на екрану

4.27. Саставити програм који учитава податке за **n** особа (име, презиме и датум рођења), а затим на основу унетог редног броја хороскопског знака исписује податке о особама рођеним у задатом хороскопском знаку.

```
#include <stdio.h>
#define MAX 100

typedef struct osobe
{
```

```

    char ime [35];
    int dan;
    int mes;
    int god;
} OSOBE;

void Citaj(OSOBE *o)
{
    printf(" Ime i prezime: ");
    gets(o->ime);
    printf(" Datum rođenja (dd.mm.yyyy): ");
    scanf("%d.%d.%d",&o->dan,&o->mes,&o->god);
    while(getchar()!='\n');
}

main()
{
    OSOBE Osoba[MAX];
    int i,n,h;
    printf("\n Unesite broj osoba: ");
    scanf("%d",&n);
    while(getchar()!='\n');
    for(i=0; i<n; i++)
    {
        printf("\n UNESITE PODATKE ZA %d. OSOBU:\n",i+1);
        Citaj(&Osoba[i]);
    }
    printf(" Unos podataka je završen...\n");
    printf("\n Izaberite hroskopski znak:\n");
    printf(" 1. OVAN\n 2. BIK\n 3. BLIZANCI\n 4. RAK\n 5. LAV\n 6. DEVICA ");
    printf("\n 7. VAGA \n 8. SKORPIJA\n 9. STRELAC\n 10. JARAC\n ");
    printf("\n 11. VODOLIJA\n 12. RIBE\n");
    printf("\n Unesite redni broj horoskopskog znaka: ");
    scanf("%d",&h);
    while(getchar()!='\n');
    printf("\n\n Osobe rođene u znaku pod rednim brojem %d su:\n",h);
    for(i=0;i<n;i++)
    {
        switch(h)
        {
            case 1:
                if(Osoba[i].dan>21 && Osoba[i].dan<=31 && Osoba[i].mes==3 ||
                    Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==4)
                    printf(" %s,\t%d.%d.%d\n",
                        Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
                break;
            case 2:
                if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==4 ||
                    Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==5)
                    printf(" %s,\t%d.%d.%d\n",
                        Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
                break;
            case 3:
                if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==5 ||
                    Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==6)
                    printf(" %s,\t%d.%d.%d\n",
                        Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
                break;
            case 4:
                if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==6 ||
                    Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==7)

```

```
        printf(" %s,\t%d.%d.%d\n",
                Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 5:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==7 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==8)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 6:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==8 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==9)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 7:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==9 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==10)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 8:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==10 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==11)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 9:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==11 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==12)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 10:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==12 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==1)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 11:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==1 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==2)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    case 12:
        if(Osoba[i].dan>21 && Osoba[i].dan<=30 && Osoba[i].mes==2 ||
            Osoba[i].dan>=1 && Osoba[i].dan<20 && Osoba[i].mes==3)
            printf(" %s,\t%d.%d.%d\n",
                    Osoba[i].ime, Osoba[i].dan, Osoba[i].mes, Osoba[i].god);
        break;
    default:
        printf(" Greska pri biranju znaka!!!");
        break;
    }
}
getche();
return 0;
}
```



Испис на екрану

4.4 Сортирање низова структура

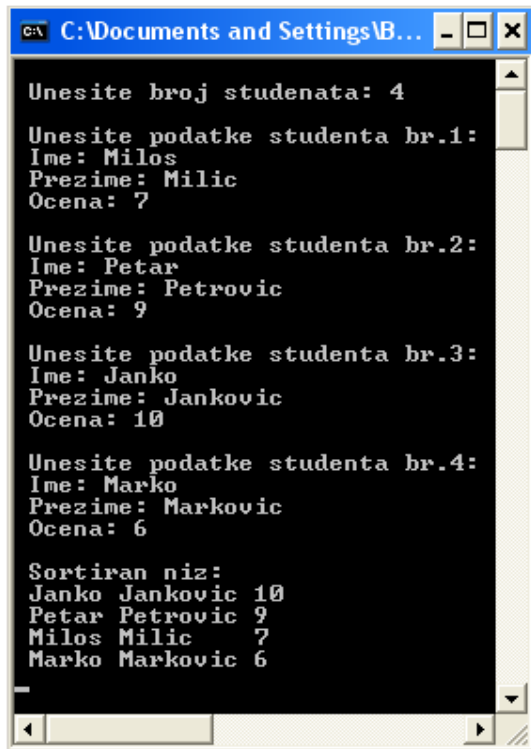
4.28. Структура о студентима се састоји од имена, презимена и оцене. Саставити програм којим се учитава низ од **n** студената и сортира их по њиховим оценама. Исписати сортирани списак студената.

```
#include <stdio.h>
#define MAX 100

typedef struct studenti
{
    char ime[15];
    char prezime[20];
    int ocena;
} STUDENTI;

main()
```

```
{
    STUDENTI Student[MAX], pom;
    int n, i, j;
    printf("\n Unesite broj studenata: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("\n Unesite podatke studenta br.:%d:\n", i+1);
        printf(" Ime: ");
        scanf("%s", &Student[i].ime);
        printf(" Prezime: ");
        scanf("%s", &Student[i].prezime);
        while(getchar()!='\n');
        printf(" Ocena: ");
        scanf("%d", &Student[i].ocena);
    }
    for(i=0; i<n-1; i++)
        for(j=i; j<n; j++)
            if(Student[i].ocena < Student[j].ocena)
            {
                pom=Student[i];
                Student[i]=Student[j];
                Student[j]=pom;
            }
    printf("\n Sortiran niz:\n");
    for(i=0; i<n; i++)
        printf(" %s %s\t%d\n",
            Student[i].ime, Student[i].prezime, Student[i].ocena);
    getch();
    return 0;
}
```



```
C:\Documents and Settings\B...
Unesite broj studenata: 4
Unesite podatke studenta br.1:
Ime: Milos
Prezime: Milic
Ocena: 7
Unesite podatke studenta br.2:
Ime: Petar
Prezime: Petrovic
Ocena: 9
Unesite podatke studenta br.3:
Ime: Janko
Prezime: Jankovic
Ocena: 10
Unesite podatke studenta br.4:
Ime: Marko
Prezime: Markovic
Ocena: 6
Sortiran niz:
Janko Jankovic 10
Petar Petrovic 9
Milos Milic 7
Marko Markovic 6
```

Испис на екрану

4.29. Структура о особама се састоји од имена и старости. Саставити програм којим се учитава низ од **n** особа и сортира их лексикографски. Уколико се јављају више особа са истим именом сортирати их по старости. Исписати сортирани списак.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

typedef struct osobe
{
    char ime[40];
    int starost;
} OSOBE;

main()
{
    OSOBE Osoba[100], pom;
    int n, i, j;
    printf("\n Unesite broj osoba: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("\n Unesite podatke o %d. osobi:\n", i+1);
        printf(" Ime: ");
        scanf("%s", &Osoba[i].ime);
        while(getchar()!='\n');
        printf(" Starost: ");
        scanf("%d", &Osoba[i].starost);
    }
    /*Ukoliko je ime i-te osobe leksikografski ispred
    (veca od) imena druge osobe ili ako su istog imena
    i i-ta osoba starija od j-te...*/
    for(i=0; i<n-1; i++)
        for(j=i+1; j<n; j++)
            if(strcmp(Osoba[i].ime, Osoba[j].ime) >0 ||
                (strcmp(Osoba[i].ime, Osoba[j].ime)==0 &&
                 Osoba[i].starost > Osoba[j].starost))
            {
                /*...razmeni mesta i-toj i j-toj Osobi.*/
                strcpy(pom.ime, Osoba[j].ime);
                pom.starost = Osoba[j].starost;
                strcpy(Osoba[j].ime, Osoba[i].ime);
                Osoba[j].starost = Osoba[i].starost;
                strcpy(Osoba[i].ime, pom.ime);
                Osoba[i].starost = pom.starost;
            }
    printf("\n Sortiran niz:\n");
    for(i=0; i<n; i++)
        printf(" %s %d\n", Osoba[i].ime, Osoba[i].starost);
    getch();
    return 0;
}
```

```

C:\Documents and Settings\... - [X]
Unesite broj osoba: 5
Unesite podatke o 1. osobi:
Ime: Marko
Starost: 33
Unesite podatke o 2. osobi:
Ime: Janko
Starost: 20
Unesite podatke o 3. osobi:
Ime: Marko
Starost: 25
Unesite podatke o 4. osobi:
Ime: Milos
Starost: 44
Unesite podatke o 5. osobi:
Ime: Aca
Starost: 12
Sortiran niz:
Aca 12
Janko 20
Marko 25
Marko 33
Milos 44

```

Испис на екрану

4.30. Структура Ханојске куле је дата са параметрима: штап са дисковима, број дискова, назив куле. Користећи ову структуру саставити програм који решава проблем тзв. "ханојских кула": дата су три вертикална штапа, на једном се налази n дискова полупречника 1, 2, 3,... до n , тако да се највећи налази на дну, а најмањи на врху. Остала два штапа су празна. Потребно је преместити дискове на други штап тако да буду у истом редоследу, премештајући један по један диск, при чему се ни у једном тренутку не сме ставити већи диск преко мањег.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_KULA 100
#define MAX_IME 50

typedef struct
{
    int s[MAX_KULA]; /*Stap sa diskovima*/
    int n; /*Broj diskova na stapu*/
    char ime[MAX_IME]; /*Naziv kule*/
} KULE;

/*Funkcija postavlja naziv kule na dato ime, a zatim na stap
postavlja diskove velicine n, n-1, ..., 2, 1 redom*/
void InicijalKule(KULE *kula, char *ime, int n)
{
    int i;
    strcpy(kula->ime, ime);
    kula->n = n;
}

```



```

    for(i=0; i<n; i++)
        kula->s[i] = n-i;
}

/*Funkcija prikazuje sadrzaj na datoj kuli*/
void StampajKulu(KULE *kula)
{
    int i;
    printf(" %s: ", kula->ime);
    for(i=0; i<kula->n; i++)
        printf("%d ", kula->s[i]);
    putchar('\n');
}

/*Funkcija premesta jedan disk sa vrha prve kule na vrh druge kule */
void Pomeri(KULE *sa, KULE *na)
{
    /*Proveravamo da li je potez ispravan*/
    if(sa->n == 0 || (na->n > 0 && sa->s[sa->n - 1] >= na->s[na->n - 1]))
    {
        printf(" Nedozvoljeni potez: %d sa %s na %s!!\n",
            sa->s[sa->n - 1],
            sa->ime,
            na->ime );
        exit(1);
    }
    else
    {
        /*Prikaz opisa poteza*/
        printf(" Premestanje diska %d sa %s na %s\n",
            sa->s[sa->n - 1],
            sa->ime,
            na->ime);
        /*Premestanje diska*/
        na->s[na->n++] = sa->s[--sa->n];
    }
}

/*Rekurzivna funkcija koja premesta n diska sa kule x na kulu y.
Kao pomocna kula koristi se kula z.*/
void Hanoi(KULE *x, KULE *y, KULE *z, int n)
{
    /*Izlaz iz rekurzije*/
    if(n == 0)
        return;
    /*Rekurzivno premestamo n-1 disk sa x na z, pomocu kule y*/
    Hanoi(x, z, y, n-1);
    /*Premestamo jedan disk sa x na y*/
    Pomeri(x,y);
    /*Prikaz stanja kula nakon poteza*/
    StampajKulu(x);
    StampajKulu(y);
    StampajKulu(z);
    /*Premestamo n-1 disk sa z na y, pomocu kule x*/
    Hanoi(z, y, x, n-1);
}

main()
{
    KULE x, y, z;
    int n;
    /*Ucitavamo dimenziju problema*/

```

```

printf("\n n= ");
scanf("%d", &n);
/*Inicijalizujemo kule. Kula x ima n diskova, ostale su prazne*/
InicijalKule(&x, "X", n);
InicijalKule(&y, "Y", 0);
InicijalKule(&z, "Z", 0);
/*Prikaz kula na pocetku*/
StampajKulu(&x);
StampajKulu(&y);
StampajKulu(&z);
/*Poziv funkcije Hanoi()*/
Hanoi(&x, &y, &z, n);
getche();
return 0;
}

```

```

C:\Documents and Settings\B...
n= 3
X: 3 2 1
Y:
Z:
Premestanje diska 1 sa X na Y
X: 3 2
Y: 1
Z:
Premestanje diska 2 sa X na Z
X: 3
Z: 2
Y: 1
Premestanje diska 1 sa Y na Z
Y:
Z: 2 1
X: 3
Premestanje diska 3 sa X na Y
X:
Y: 3
Z: 2 1
Premestanje diska 1 sa Z na X
Z: 2
X: 1
Y: 3
Premestanje diska 2 sa Z na Y
Z:
Y: 3 2
X: 1
Premestanje diska 1 sa X na Y
X:
Y: 3 2 1
Z:

```

Испис на екрану

4.31. Саставити програм који проналази и испишује све анаграме међу унетим речима. Две речи су анаграми ако и само ако се састоје од истог броја истих слова. Речи при уносу су раздвојене тастером ENTER. Уно речи се завршава сигналом EOF.

Алгоритам се заснива на одређивању канонске форме сваке речи. Канонска форма се одређује сортирањем слова сваке речи. Речи су анаграми ако и само ако су им канонске форме једнаке. Сортирањем низа речи на основу канонских форми анаграми се појаве један уз други у низу.

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define MAX 50

struct rec
{
    char original[MAX];
    char kanon[MAX];
};

/*funkcija koja učitava rec*/
int UcitajRec(char s[], int lim)
{

```

```

    int c, i=0;
    while(!isalpha(c=getchar()))
        if(c==EOF) return 0;
    do
    {
        s[i++] = c;
    }
    while(i<lim-1 && isalpha(c=getchar()));
    s[i] = '\0';
    return i;
}

/*funkcija koja vrsi poredjenje dva slova u reci*/
int PorediSlova(const void* a, const void* b)
{
    return *(char*)a - *(char*)b;
}

/*funkcija koja vrsi poredjenje po dve reci, na osnovu njihovih kanonskih formi*/
int PorediReci(const void* a, const void* b)
{
    return strcmp(((struct rec*)a)->kanon, ((struct rec*)b)->kanon);
}

/*funkcija koja proverava da li su reci anagrami*/
int Anagram(struct rec a, struct rec b)
{
    return strcmp(a.kanon, b.kanon) == 0
        && strcmp(a.original, b.original) != 0;
}

main()
{
    int i, n;
    struct rec reci[1000];
    printf("\n UCITAJTE RECI:\n");
    for(i=0; UcitajRec(reci[i].original, MAX) != 0; i++)
    {
        /*odredjujemo kanonsku formu svake reci*/
        strcpy(reci[i].kanon, reci[i].original);
        qsort(reci[i].kanon, strlen(reci[i].kanon),
                sizeof(char), &PorediSlova);
    }
    n=i;
    printf("\n Procitano %d reci.\n", n);
    /*sortiramo niz reci na osnovu kanonske forme*/
    qsort(reci, n, sizeof(struct rec), &PorediReci);
    /*pronalezimo anagrame kao susedne elemente niza i ispisujemo ih */
    for(i=1; i<n; i++)
        if(Anagram(reci[i-1], reci[i]))
        {
            printf(" Anagram : %s - %s", reci[i-1].original, reci[i].original);
            while(i+1 < n && Anagram(reci[i], reci[i+1]))
            {
                printf(" - %s", reci[i+1].original);
                i++;
            }
            printf("\n");
        }
    getch();
    return 0;
}

```

```

C:\Documents and Sett...
UCITAJTE RECI:
madam
hanan
damam
zub
buz
vulkan
mama
anhan
tata
kanvul
^Z

Procitano 10 reci.
Anagram : damam - madam
Anagram : anhan - hanan
Anagram : vulkan - kanvul
Anagram : zub - buz

```

Испис на екрану

4.32. Са улаза се уносе речи све док се не прочита EOF. Саставити програм који броји појављивање сваке од кључних речи: **break**, **continue**, **float**, **for**, **if**, **return**, **struct**, **while**. На крају се речи испишу по опадајућем броју појављивања. За претрагу и сортирање користити уграђене функције **bsearch()** и **qsort()**.

```

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

/*Svaka ključna rec se odlikuje imenom i brojem pojavljivanja*/
typedef struct ključna
{
    char rec[20];
    int num;
}KLJUCNA;

/*Kreiramo niz struktura sortiranih leksikografski po
imenu ključne reci, kako bismo ubrzali pronalazak reci.*/
KLJUCNA KljučneReci[]={{"break",0},
                        {"continue",0},
                        {"float",0},
                        {"for",0},
                        {"if",0},
                        {"return",0},
                        {"struct",0},
                        {"while",0}};

/*Funkcija cita sledeću rec sa standardnog ulaza.*/
int UcitajRec(char rec[], int lim)
{
    int c, i=0;
    while(!isalpha(c=getchar()) && c!=EOF)
        ;
    if(c==EOF) return -1;
    do
    {
        rec[i++]=c;
    }

```

```

C:\Doc...
UCITAJTE RECI:
break
continue
continue
integer
while
for
while
while
break
for
for
for
real
^Z

for 4
while 3
continue 2
break 2
return 0
struct 0
if 0
float 0

```

Испис на екрану

```

    } while(--lim>0 && isalpha(c=getchar()));
    rec[i]='\0';
    return i;
}

/*Funkcija leksikografskog poredjenja za bsearch.*/
int cmp(const void* a, const void* b)
{
    return strcmp((char*)a, (*(KLJUCNA*)b).rec);
}

/*Funkcija numerickog poredjenja za qsort.*/
int numcmp(const void* a, const void* b)
{
    return ((*(KLJUCNA*)b).num-(*(KLJUCNA*)a).num);
}

main()
{
    char rec[80];
    int i;
    int BrojKljucnihReci=sizeof(KljucneReci)/sizeof(KLJUCNA);
    printf("UCITAJTE RECI:\n");
    while (UcitajRec(rec,80)!=-1)
    {
        /*Trazimo rec u spisku kljucnih reci binarnom pretragom */
        KLJUCNA* k=(KLJUCNA*)bsearch(
                                (void*)rec,
                                (void*)KljucneReci,
                                BrojKljucnihReci,
                                sizeof(KLJUCNA),
                                cmp
                                );
        /*Ukoliko je pronadjena uvecavamo broj pojavljivanja.*/
        if(k!=NULL)
            (*k).num++;
    }
    /*Sortiramo niz na osnovu broja pojavljivanja.*/
    qsort((void*)KljucneReci, BrojKljucnihReci, sizeof(KLJUCNA), numcmp);
    /*Vrsimo ispis.*/
    printf("\n\n");
    for(i=0; i<BrojKljucnihReci; i++)
        printf("%s %d\n", KljucneReci[i].rec, KljucneReci[i].num);
    getche();
    return 0;
}

```

4.5 Структуре и динамичка зона меморије

4.33. Структура садржи податке о животињама: име, врста и број година. Саставити програм који учита податке за **n** животиња, а затим испишује исте користећи динамичку алокацију низа структура преко одговарајућег низа показивача.

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct animals
{
    char ime[25];
    char vrsta[25];
    int godina;
} ANIMALS;

main()
{
    ANIMALS *Animal[MAX], *a;
    int i, n;
    printf("\n Unesite broj zivotinja: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        Animal[i]=(ANIMALS *)malloc(sizeof(ANIMALS));
        if(Animal[i]==NULL)
        {
            printf(" GRESKA!!!\n");
            exit(1);
        }
        printf("\n ZIVOTINJA BR. %d:\n", i+1);
        while(getchar()!='\n');
        printf(" Ime: ");
        gets(Animal[i]->ime);
        printf(" Vrsta: ");
        gets(Animal[i]->vrsta);
        printf(" Godina: ");
        scanf("%d",&Animal[i]->godina);
    }
    printf("\n PODACI:\n");
    for(i=0; i<n; i++)
        printf(" %s je %s, i ima %d godina.\n",
            Animal[i]->ime, Animal[i]->vrsta, Animal[i]->godina);
    for(i=0; i<12; i++)
        free(Animal[i]);
    getch();
    return 0;
}

```

```

C:\Documents and Settings\Ban...
Unesite broj zivotinja: 3

ZIVOTINJA BR. 1:
Ime: John
Vrsta: pas
Godina: 6

ZIVOTINJA BR. 2:
Ime: Kiki
Vrsta: papagaj
Godina: 3

ZIVOTINJA BR. 3:
Ime: Cezar
Vrsta: pas
Godina: 5

PODACI:
John je pas, i ima 6 godina.
Kiki je papagaj, i ima 3 godina.
Cezar je pas, i ima 5 godina.

```

Испис на екрану

4.34. Подаци о вектору се састоје од координата **x**, **y** и **z**. Саставити структуру за чување података о вектору, а затим саставити програм који за учитава **n** вектора и сортира и исписује их по интензитету коришћењем сортирање уметањем. Низ структура сместити у динамичку зону меморије.

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

typedef struct vektori
{
    float x;
    float y;
    float z;
}VEKTORI;

void Ucitaj(VEKTORI *Vektor, int n)
{
    int i;
    for(i=0; i<n; i++)
    {
        printf("\n UNESITE KOORDINATE VEKTORA BR.%d:\n", i+1);
        printf(" x= ");
        scanf("%f", &Vektor[i].x);
        printf(" y= ");
        scanf("%f", &Vektor[i].y);
        printf(" z= ");
        scanf("%f", &Vektor[i].z);
    }
}

void Ispisi(VEKTORI *Vektor, int n)
{
    int i;
    for(i=0; i<n; i++)
        printf(" (%.2f, %.2f, %.2f)\n", Vektor[i].x, Vektor[i].y, Vektor[i].z);
}

float Intenzitet(VEKTORI v)
{
    return (float)sqrt(v.x*v.x+v.y*v.y+v.z*v.z);
}

void Sortiraj(VEKTORI *a, int n)
{
    int i, j;
    for(i=1; i<n; i++)
    {
        VEKTORI x = a[i];
        j=i-1;
        while(j>0 && Intenzitet(x) < Intenzitet(a[j]))
        {
            a[j+1]=a[j];
            j=j-1;
        }
        a[j+1]=x;
    }
}

main()
{
```

```
int n;  
VEKTORI *Vektor;  
printf("\n Unesite broj vektora: ");  
scanf("%d", &n);  
Vektor=(VEKTORI*)malloc(sizeof (VEKTORI)*n);  
if(Vektor==NULL)  
{  
    printf("\n GRESKA!");  
    return 1;  
}  
  
Ucitaj(Vektor,n);  
Sortiraj(Vektor,n);  
printf("\n Sortirani vektori po intenzitetu:\n");  
Ispisi(Vektor,n);  
free(Vektor);  
getche();  
return 0;  
}
```

Испис на екрану

4.35. Саставити програм који исписује број појављивања сваке речи са стандардног улаза. Структура која описује унету реч састоји се од имена речи и њене дужине. Речи приликом уноса се раздвајају тастером ENTER. Крај уноса је означен сигналом EOF. Користити динамички низ и динамичку алокацију меморије.


```

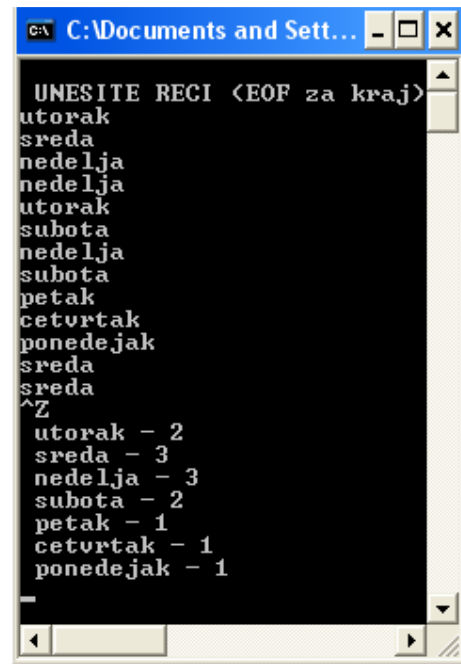
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define KORAK 10

typedef struct rec
{
    char ime[80];
    int brPojav;
} REC;

/*Funkcija ucitava rec i vraca njenu duzinu
ili -1 ukoliko smo dosli do znaka EOF*/
int UcitajRec(char word[], int max)
{
    int c, i=0;
    while(isspace(c=getchar()))
        ;
    while(!isspace(c) && c!=EOF && i<max-1)
    {
        word[i++]=c;
        c=getchar();
    }
    word[i]='\0';
    if(c==EOF) return -1;
    else return i;
}

main()
{
    /*Dinamicki niz reci je opisan pokazivacem na pocetak, tekucim brojem
    upisanih elemenata i tekucim brojem alociranih elemenata*/
    REC *nizReci;
    int i, duzina=0, alocirano=0;
    char procitanaRec[80];
    printf("\n UNESITE RECI (EOF za kraj):\n");
    while(UcitajRec(procitanaRec,80)!=-1)
    {
        /*Proveravamo da li rec vec postoji u nizu*/
        for(i=0; i<duzina; i++)
        /*Ako bismo uporedili procitana_rec == niz_reci[i].ime
        bili bi uporedjeni pokazivaci a ne odgovarajuci sadrzaji.
        Zato koristimo strcmp. */
        if(strcmp(procitanaRec, nizReci[i].ime)==0)
        {
            nizReci[i].brPojav++;
            break;
        }
        /*Ukoliko rec ne postoji u nizu*/
        if(i==duzina)
        {
            REC novaRec;
            /*Ako bismo dodelili nova_rec.ime = procitana_rec
            izvrsila bi se dodela pokazivaca a ne kopiranje niske
            procitana_rec u nova_rec.ime. Zato koristimo strcpy.*/
            strcpy(novaRec.ime, procitanaRec);
            novaRec.brPojav=1;
            /*Ukoliko je niz "kompletно popunjen" vrsimo realokaciju*/
            if(duzina==alocirano)
            {
                alocirano+=KORAK;

```



Испис на екрану

```

    {
        /*Alociramo novi niz, veci nego sto je bio prethodni*/
        REC *noviNiz=(REC *)malloc(allocirano*sizeof(REC));
        if(noviNiz == NULL)
        {
            free(nizReci);
            printf(" Greska prilikom alokacije memorije!\n");
            exit(1);
        }
        /*Kopiramo elemente starog niza u novi*/
        for(i=0; i<duzina; i++)
            noviNiz[i]=nizReci[i];
        free(nizReci); /*Uklanjammo stari niz*/
        nizReci=noviNiz; /*Stari niz postaje novi*/
    }
}
/*Upisujemo rec u niz*/
nizReci[duzina]=novaRec;
duzina++;
}
}
/*Ispisujemo elemente niza*/
for(i=0; i<duzina; i++)
    printf(" %s - %d\n",nizReci[i].ime, nizReci[i].brPojav);
free(nizReci);
getche();
return 0;
}

```

4.36. Саставити програм за распоређивање студената по салама за полагање испита. На почетку програма треба учитати број студената и укупан број сала. Структура садржи податке о салама: редни број сале и број места у свакој од сала. Сале треба попуњавати редом, почев од оних са највећим бројем места. На крају треба исписати редне бројеве оних сала, које ће се користити за полагање испита. Користити динамичку алокацију меморије приликом резервисања места у салама.

```

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

typedef struct sala
{
    int brSale, mesta;
} SALA;

main()
{
    SALA *salaUkupno;
    int sl, st, i, j;
    printf("\n Broj sala: ");
    scanf("%d", &sl);
    printf(" Broj studenata: ");
    scanf("%d", &st);
    salaUkupno=malloc((sl+1)*sizeof(SALA));
    if(salaUkupno == NULL)
    {
        printf(" Greska u alokaciji memorije!\n");
    }
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop\VZ...
Broj sala: 5
Broj studenata: 300
Broj mesta u sali 1: 120
Broj mesta u sali 2: 30
Broj mesta u sali 3: 90
Broj mesta u sali 4: 15
Broj mesta u sali 5: 80

Za polaganje ispita koristice se sledece sale:
1
3
5
2

```

Испис на екрану

```

    return 1;
}
for(i=0; i<sl; i++)
{
    salaUkupno[i].brSale=i+1;
    printf(" Broj mesta u sali %d: ", i+1);
    scanf("%d", &salaUkupno[i].mesta);
}
for(i=0; i<sl-1; i++)
{
    for(j=i+1; j<sl; j++)
    {
        if(salaUkupno[i].mesta < salaUkupno[j].mesta)
        {
            salaUkupno[sl] = salaUkupno[i];
            salaUkupno[i] = salaUkupno[j];
            salaUkupno[j] = salaUkupno[sl];
        }
    }
}
printf("\n Za polaganje ispita koristice se sledece sale:\n");
printf(" %d\n", salaUkupno[0].brSale);
for(i=1; i<sl; i++)
{
    st-=salaUkupno[i-1].mesta;
    if(st>0)
        printf(" %d\n", salaUkupno[i].brSale);
    else
        break;
}
free(salaUkupno);
getche();
return 0;
}

```

4.6 Уније

4.37. Саставити програм који користећи унију са три члана (цео број, реални број и знак) штампа тренутну вредност уније. Члановима уније иницијализовати вредности у програму.

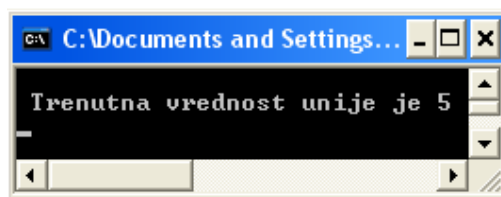
```

#include <stdio.h>

typedef union podaci
{
    int i;
    float f;
    char c;
} PODACI;

main()
{
    PODACI unija;

```



Испис на екрану

```

unija.c='A';
unija.i=5;
/*Dozvoljen je pristup samo poslednje dodeljenom clanu unije*/
printf("\n Trenutna vrednost unije je %d\n",unija.i);
/*Pogresno bi bilo da se napise
printf("Trenutna vrednost unije je %c\n",unija.c); */
getche();
return 0;
}

```

4.38. Саставити програм који користећи унију, која може да садржи реалан или цео број, за унети цео и реалан број штампва њихове вредности.

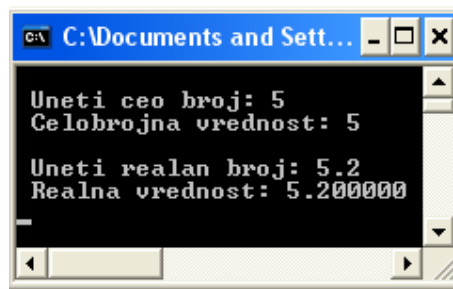
```

#include <stdio.h>

typedef union CeoIliRealan
{
    int ceo;
    double realan;
} CEOILIREALAN;

main ()
{
    CEOILIREALAN x;
    /*Koristimo x kao ceo broj*/
    printf("\n Uneti ceo broj: ");
    scanf("%d", &x.ceo);
    printf(" Celobrojna vrednost: %d\n", x.ceo);
    /*Koristimo x kao realan broj*/
    printf("\n Uneti realan broj: ");
    scanf("%lf", &x.realan);
    printf(" Realna vrednost: %f\n", x.realan);
    getche();
    return 0;
}

```



Испис на екрану

4.39. Шта се исписује на екрану након извршавања следећег програмског кода:

```

#include <stdio.h>
#define CHARACTER 'C'
#define INTEGER 'I'
#define FLOAT 'F'

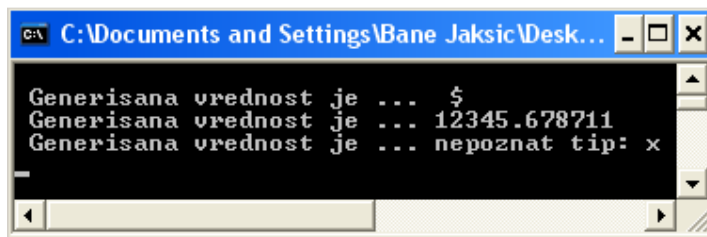
struct struktura
{
    char tip;
    union unija
    {
        char c;
        int i;
        float f;
    } un;
};

void Stampaj(struct struktura st)
{
    printf("\n Generisana vrednost je ... ");
    switch(st.tip)

```

```
{
    case CHARACTER: printf(" %c", st.un.c);
                    break;
    case INTEGER: printf(" %d", st.un.i);
                  break;
    case FLOAT: printf("%f", st.un.f);
                break;
    default: printf("nepoznat tip: %c\n", st.tip);
             break;
}
}

main()
{
    struct struktura pr;
    pr.tip=CHARACTER;
    pr.un.c='$';
    Stampaj(pr);
    pr.tip=FLOAT;
    pr.un.f=(float)12345.67890;
    Stampaj(pr);
    pr.tip='x';
    pr.un.i=111;
    Stampaj(pr);
    getch();
    return 0;
}
```



Испис на екрану

5 ДАТОТЕКЕ

5.1 Основне операције са датотекама

Табела 5.1: Функције за рад са текстуалним датотекама дефинисане у библиотеци `<stdio.h>`

| функција | значење |
|------------------------|---|
| <code>fopen()</code> | отварање датотеке |
| <code>fclose()</code> | затварање датотеке |
| <code>fscanf()</code> | форматирано читање из датотеке |
| <code>fgetc()</code> | читање карактер по карактер из датотеке |
| <code>fgets()</code> | читање ред по ред из датотеке |
| <code>fprintf()</code> | форматиран упис у датотеку |
| <code>fputc()</code> | упис карактер по карактер у датотеку |
| <code>fputs()</code> | упис ред по ред у датотеку |

Табела 5.2: Мод отварања текстуалне датотеке

| ознака | значење |
|--------|---|
| "w" | Отварање за упис од почетка. |
| "r" | Отварање за читање. |
| "a" | Отварање за упис у продужетку постојећег садржаја. |
| "w+" | Отварање за упис од почетка и за читање после тога. |
| "r+" | Отварање за читање и упис после тога. |
| "a+" | Отварање за упис у продужетку постојећег садржаја и за читање после тога. |

5.1. Саставити програм којим се у датотеку **podaci.txt** уписује првих 10 целих бројева, а затим се из исте датотеке читају бројеви док се не достигне до краја датотеке, а затим се бројеви исписују на стандарни излаз (екран).

```
#include <stdio.h>

main()
{
    int i, br;
    FILE *dato; /*Deklarisanje pokazivaca na datoteku*/
    /*Otvaramo datoteku sa imenom podaci.txt za pisanje*/
    dato=fopen("podaci.txt", "w");

    /*Ukoliko otvaranje nije uspeo, fopen vraca NULL. U tom slucaju,
```

```
prijavljujemo gresku i završavamo program */
if(dato==NULL)
{
    printf("\n Greska prilikom otvaranja datoteke podaci.txt za
pisanje!\n");
    exit(1);
}

/*Upisujemo u datoteku prvih 10 prirodnih brojeva (svaki u posebnom redu)*/
for(i=0; i<10; i++)
    fprintf(dato, "%d\n", i);

/*Zatvaramo datoteku*/
fclose(dato);

/*Otvaramo datoteku sa imenom podaci.txt za citanje*/
dato= fopen("podaci.txt", "r");

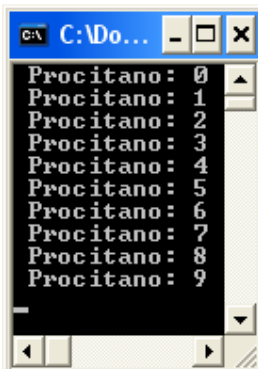
/*Ukoliko otvaranje nije uspeo, fopen vraca NULL. U tom slucaju,
prijavljujemo gresku i završavamo program*/
if(dato==NULL)
{
    printf("\n Greska prilikom otvaranja datoteke podaci.txt za
citanje!\n");
    exit(1);
}

/*Citamo brojeve iz datoteke dok ne stignemo do kraja i ispisujemo ih
na standardni izlaz*/
while(1)
{
    /*Pokusavamo da procitamo broj*/
    fscanf(dato, "%d", &br);

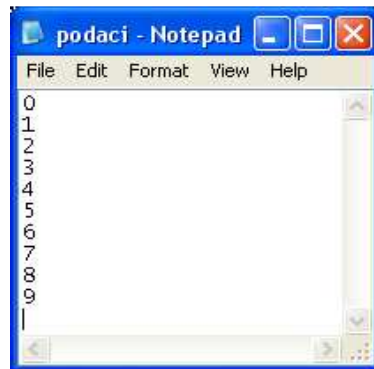
    /*Ukoliko smo dosli do kraja datoteke, prekidamo*/
    iffeof(dato))
        break;

    /*Ispisujemo procitani broj*/
    printf(" Procitano: %d\n", br);
}

/*Zatvaramo datoteku*/
fclose(dato);
getche();
return 0;
}
```



Изглед на екрану



Изглед датотеке podaci.txt

5.2. Дата је датотека **podaci2.txt** чији је садржај приказан на слици. Саставити програм који у постојећој датотеци дописује текст "Pozdrav svima".

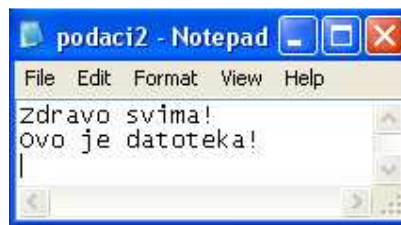
```
#include <stdio.h>

main()
{
    FILE *dato;

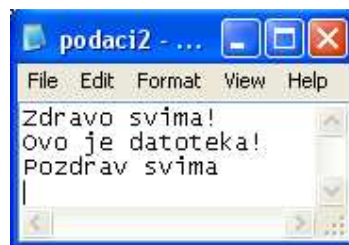
    /*Otvaramo datoteku za nadovezivanje i proveravamo da li je doslo do greske*/
    if((dato=fopen("podaci2.txt", "a"))==NULL)
    {
        printf("\n Greska prilikom otvaranja datoteke podaci2.txt\n");
        exit(1);
    }

    /*Upisujemo sadrzaj u datoteku*/
    fprintf(dato, "Pozdrav svima\n");

    /*Zatvaramo datoteku*/
    fclose(dato);
    getche();
    return 0;
}
```



Изглед датотеке **podaci2.txt**

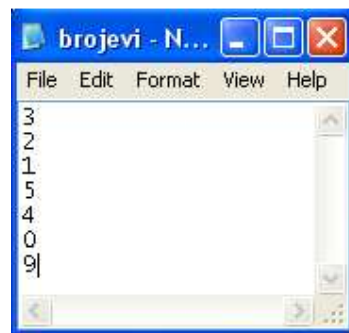


Изглед датотеке **podaci2.txt** након извршавања програма

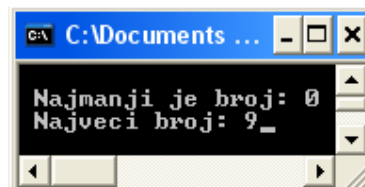
5.3. Сваки ред датотеке **brojevi.txt** садржи по један цео број. Саставити програм који штампа најмањи и највећи цео број на екрану.

```
#include <stdio.h>

main()
{
    FILE *dato;
    int najmanji, najveći, broj;
    dato=fopen("brojevi.txt", "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fscanf(dato, "%d", &broj);
    najmanji=broj;
    najveći=broj;
    while(fscanf(dato, "%d", &broj)!=EOF)
    {
        if(broj<najmanji) najmanji=broj;
        if(broj>najveći) najveći=broj;
    }
    fclose(dato);
    printf("\n Najmanji je broj: %d", najmanji);
    printf("\n Najveći broj: %d", najveći);
    getche();
    return 0;
}
```



Изглед датотеке **brojevi.txt**



Испис на екрану

5.4. Дата је датотека **BrojeviStari.txt** која у сваком реду садржи по један цео број. Саставити програм који формира датотеку **BrojeviNovi.txt** тако што из датотеке **BrojeviStari.txt** преписује бројеве из парних редова, а из непарних удвостручује.

```
#include <stdio.h>

main()
{
    int broj, i=1;
    FILE *a,*b;
    a=fopen("BrojeviStari.txt","r");
    if(a==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    b=fopen("BrojeviNovi.txt","w");
    if(b==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(fscanf(a,"%d",&broj)!=EOF)
    {
        if(i%2==0)
            fprintf(b,"%d\n", broj);
        else
            fprintf(b,"%d\n", 2*broj);
        i++;
    }
    fclose(a);
    fclose(b);
    getche();
    return 0;
}
```



Изглед датотеке **BrojeviStari.txt**



Изглед датотеке **BrojeviNovi.txt**
након извршавања програма

5.5. Дата је датотека **BrojeviX.txt** која у сваком реду садржи по један природни број. Саставити програм који формира датотеку **BrojeviY.txt** тако што из датотеке **BrojeviX.txt** преписује само оне бројеве који се састоје од цифара 3 и 7.

```
#include <stdio.h>

main()
{
    int broj, pom, cif, ind;
    FILE *a,*b;
    a=fopen("BrojeviX.txt","r");
    if(a==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    b=fopen("BrojeviY.txt","w");
    if(b==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
}
```

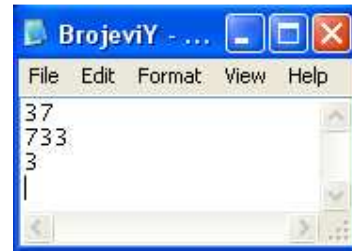


Изглед датотеке **BrojeviX.txt**

```

while(fscanf(a, "%d", &broj) != EOF)
{
    ind=1;
    pom=broj;
    while(pom!=0)
    {
        cif=pom%10;
        if(cif!=3 && cif!=7)
            ind=0;
        pom/=10;
    }
    if(ind && broj!=0)
        fprintf(b, "%d\n", broj);
}
fclose(a);
fclose(b);
getche();
return 0;
}

```



Изглед датотеке **BrojeviY.txt**
након извршавања програма

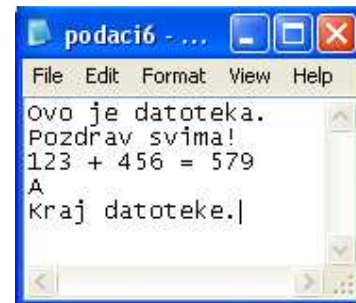
5.6. Саставити програм који креира датотеку **podaci6.txt** чији је садржај приказан на слици. Користити функције за уписивање у датотеку **fprint()**, **fputs()**, **fputc()**.

```

#include <stdio.h>

main()
{
    FILE *dato;
    dato=fopen("podaci6.txt", "w");
    if(dato==NULL)
    {
        printf("Greska prilikom otvaranja datoteke");
        exit(1);
    }
    fprintf(dato, "Ovo je datoteka.\n");
    fprintf(dato, "Pozdrav svima!\n");
    fprintf(dato, "%d + %d = %d\n", 123, 456, 123+456);
    /*Funkcija fputc upisuje jedan karakter u datoteku.*/
    fputc('A', dato);
    /*Funkcija fputs upisuje string u datoteku.*/
    fputs("\nKraj datoteke.", dato);
    fclose(dato);
    getche();
    return 0;
}

```



Изглед датотеке **podaci6.txt**

5.7. Саставити програм који исписује на екрану колико има редова и колико знакова у датотеци **podaci6.txt**.

```

#include <stdio.h>

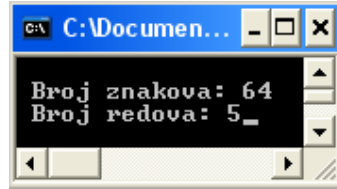
main()
{
    FILE *dato;
    long znak;
    int brZnak=0, brRed=1;
    dato=fopen("podaci6.txt", "r");
    if(dato==NULL)

```

```

{
    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
/*Funkcija fgetc ucitava iz datoteke jedan karakter.*/
while((znak=getc(dato))!=EOF)
{
    if(znak=='\n') brRed++;
    brZnak++;
}
fclose(dato);
printf("\n Broj znakova: %d", brZnak);
printf("\n Broj redova: %d", brRed);
getche();
return 0;
}

```



Испис на екрану

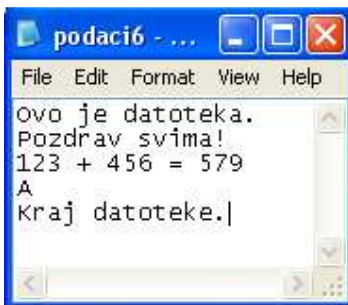
5.8. Саставити програм који употребом функције `fgets()` чита садржај датотеке **podaci6.txt** и исписује на екрану. Максимална дужина која се чита је по 20 знакова.

```

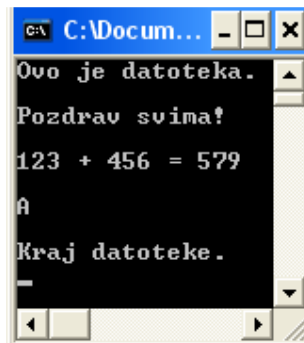
#include <stdio.h>
#define MAX 20

main()
{
    FILE *dato;
    char str[MAX];
    dato=fopen("podaci6.txt", "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    /*Funkcija fgets() cita iz datoteke string do
    sledeceg znaka nove linije ili max-1 znakova*/
    while(fgets(str, MAX, dato)!=NULL)
        puts(str);
    fclose(dato);
    getche();
    return 0;
}

```



Изглед датотеке podaci6.txt



Испис на екрану

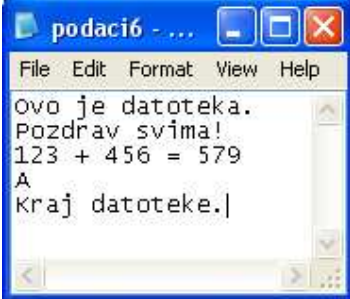
5.9. Саставити програм који преписује садржај датотеке **podaci6.txt** у датотеку **podaci6Novo.txt**:

- а) карактер по карактер;
- б) линију по линију.

а)

```
#include <stdio.h>

main()
{
    char c;
    FILE *ulaz, *izlaz;
    ulaz=fopen("podaci6.txt", "r");
    if(ulaz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    izlaz=fopen("podaci6Novo.txt", "w");
    if(izlaz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while((c=fgetc(ulaz))!=EOF)
        fputc(c,izlaz);
    fclose(ulaz);
    fclose(izlaz);
    getche();
    return 0;
}
```




Изглед датотеке **podaci6.txt**

б)

```
#include <stdio.h>
#define MAX 100

main()
{
    char linija[MAX];
    FILE *ulaz, *izlaz;
    ulaz=fopen("podaci6.txt", "r");
    if(ulaz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    izlaz= fopen("podaci6Novo.txt", "w");
    if(izlaz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(fgets(linija, MAX, ulaz)!=NULL)
        fputs(linija, izlaz);
    fclose(ulaz);
    fclose(izlaz);
    getche();
    return 0;
}
```

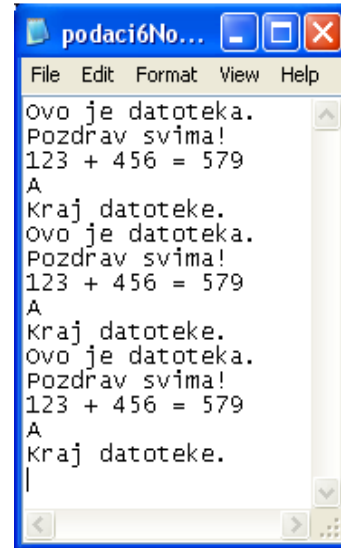


Изглед датотеке **podaci6Novo.txt**
након извршавања програма

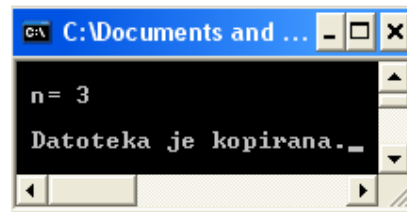
5.10. Саставити програм који за унто **n** преписује **n** пута садржај датотеке **podaci6.txt** у датотеку **podaci6NovoN.txt** карактер по карактер.

```
#include <stdio.h>

main()
{
    int i, c, n;
    FILE *ulaz, *izlaz;
    if((ulaz=fopen("podaci6.txt","r"))==NULL)
    {
        printf("Greska pri otvaranju datoteke!");
        exit(1);
    }
    if((izlaz=fopen("podaci6NovoN.txt","w"))==NULL)
    {
        printf("Greska pri otvaranju datoteke!");
        exit(1);
    }
    printf("\n n= ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        while((c=fgetc(ulaz))!=EOF)
            fputc(c,izlaz);
        /*postavljamo poziciju na pocetak fajla*/
        rewind(ulaz);
        fputc('\n',izlaz);
    }
    fclose(ulaz);
    fclose(izlaz);
    printf("\n Datoteka je kopirana.");
    getch();
    return 0;
}
```



Изглед датотеке
podaci6NovoN.txt након
извршавања програма

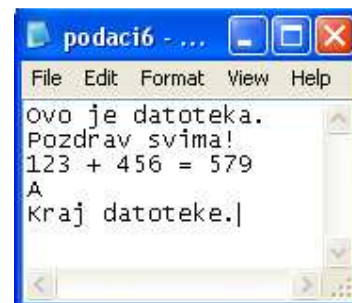


Испис на екрану

5.11. Саставити програм који преписује садржај датотеке **podaci6.txt** у датотеку **podaci6A.txt** уз промену малог слова 'a' у велико 'A'. Имена датотека се уносе са тастатуре.

```
#include <stdio.h>

main()
{
    FILE *ulaz, *izlaz;
    char imeUlaz[50], imeIzlaz[50], znak;
    printf("\n Unesite ime ulazne datoteke: ");
    scanf("%s", imeUlaz);
    printf(" Unesite ime izlazne datoteke: ");
    scanf("%s", imeIzlaz);
    ulaz=fopen(imeUlaz, "r");
    izlaz=fopen(imeIzlaz, "w");
    if((ulaz==NULL) || (izlaz==NULL))
    {
        printf("Greska pri otvaranju datoteke!");
    }
}
```

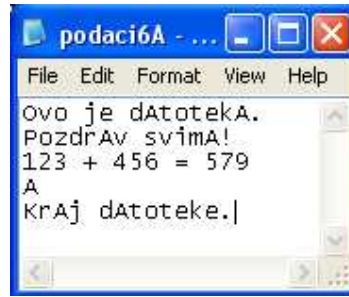


Изглед датотеке **podaci6.txt**

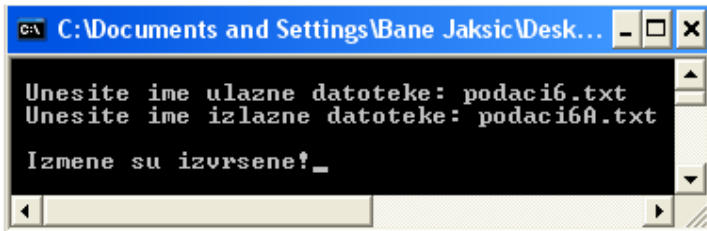
```

    fclose(ulaz);
    fclose(izlaz);
    exit(1);
}
while((znak=fgetc(ulaz)) != EOF)
{
    if(znak=='a')znak='A';
    fputc(znak, izlaz);
}
fclose(ulaz);
fclose(izlaz);
printf("\n Izmene su izvorsene!");
getche();
return 0;
}

```



Изглед датотеке **podaci6A.txt**
након извршавања програма



Испис на екрану

5.12. Дата је датотека **slova.txt**. Шта се исписује на екрану након извршавања следећег програмског кода:

```

#include <stdio.h>

main()
{
    FILE* ulaz;
    char c;
    ulaz=fopen("slova.txt", "r");
    if(ulaz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    /*nakon otvaranja datoteke za citanje, pocetna pozicija je postavljena na 0*/
    /*1 char podatak = 1 bajt*/
    c=fgetc(ulaz); /*procitano je slovo sa pozicije 0 (a),
                   i pozicija u fajlu je postavljena na 1*/
    putchar(c); /*ispisano je slovo a*/
    putchar('\n');
    c=fgetc(ulaz); /*procitano je slovo sa pozicije 1 (b)
                   i pozicija u fajlu je postavljena na 2*/
    putchar(c); /*ispisano je slovo b*/
    putchar('\n');
    printf("Tekuca pozicija: %ld\n", ftell(ulaz));
    fseek(ulaz,5,SEEK_CUR); /*pozicija u fajlu se pomera za +5 mesta (bajtova) u
                           odnosu na tekucu poziciju, tekuca pozicija nam je
                           bila 2, pa je sada 2+5=7*/
    c=fgetc(ulaz); /*citamo karakter sa pozicije 7,
                   i pozicija se uvecava za 1-- sada je 8*/
    putchar(c);
    putchar('\n');
    fseek(ulaz,10, SEEK_CUR); /*pozicija u fajlu se pomera za +10 mesta
                              (bajtova) u odnosu na tekucu poziciju, tekuca
                              pozicija nam ej bila 8, pa je sada 8+10=18*/
}

```

```

c=fgetc(ulaz); /*citamo karakter sa pozicije 18 i pozicija se
                uvecava za 1 -- sada je 19*/
putchar(c);
putchar('\n');
fseek(ulaz, -4, SEEK_CUR); /*pozicija u fajlu se pomera za -4 mesta (bajta)
                            u odnosu na tekucu poziciju,tekuca pozicija nam
                            je 19, pa je sada 19-4=15*/
c=fgetc(ulaz); /*citamo karakter sa pozicije 15, i pozicija se
                uvecava za 1 -- sada je 16*/
putchar(c);
putchar('\n');
printf("Tekuca pozicija: %ld\n", ftell(ulaz));
/*tekuca pozicija u fajlu se moze dobiti i sa fseek(ulaz,0,SEEK_CUR)*/
fseek(ulaz,4, SEEK_SET); /*pozicija u fajlu se pomera za 4 mesta(bajta) u
                        odnosu na pocetak fajla -- poziciju 0, tekuca
                        pozicija nam je 0+4=4 */
c=fgetc(ulaz); /*citamo karakter sa pozicije 4 i poziciju
                uvecavamo za 1-- sada je to 5*/
putchar(c);
putchar('\n');
printf("Povratna vrednost funkcije fseek: %d\n",
        fseek(ulaz,-126,SEEK_SET));
/*ovaj poziv fije ne prolazi, tekuca pozicija ostaje 5*/
c=fgetc(ulaz); /*citamo karakter sa pozicije 5 i pozicija se uvecava za 1*/
putchar(c);
putchar('\n');
/*pozicioniramo se na kraj fajla*/
/*broj slova u datoteci je 26, svako slovo jedan bajt - 26 bajtova;
  racuna se i EOF, znaci 27 bajtova, sto znaci da se pozicioniramo na 28mi bajt
  numeracija pocinje od 0, pa ocekujemo da bude ispisana pozicija 27*/
fseek(ulaz,0,SEEK_END);
printf("Pozicioniranje na kraj: %ld\n", ftell(ulaz));
fseek(ulaz, -3, SEEK_END); /*modifikujemo tekucu poziciju relativno u
                            odnosu na kraj fajla: pozicija je 27, idemo za
                            3 pozicije unazad, pa je nova pozicija 27-3=24*/
printf("Tekuca pozicija: %ld\n", ftell(ulaz));
c=fgetc(ulaz); /*citamo karakter sa pozicije 24 i pozicija
                se uvecava za 1 -- sada je 25*/
putchar(c);
putchar('\n');
printf("Tekuca pozicija: %ld\n", ftell(ulaz));
/*SMEMO DA PRISTUPAMO OPSEGU NAKON KRAJA FAJLA*/
printf("Povratna vrednost funkcije fseek: %d\n",
        fseek(ulaz,10,SEEK_END));
c=fgetc(ulaz);
putchar(c);
putchar('\n');
/*postavljamo poziciju na pocetak fajla*/
rewind(ulaz);
printf("Tekuca pozicija: %ld\n", ftell(ulaz));
/*isto smo mogli postici i sa: fseek(ulaz,0,SEEK_SET)*/
/*zatvaramo datoteku za citanje*/
fclose(ulaz);
getche();
return 0;
}

```

Функција `int fseek(FILE* stream, long offset, int whence)` служи за подешавање позиције у фајлу. Нова позиција се добија тако што се `offset` величина дода на вредност наведену `whence` аргументом. Повратна вредност функције је 0 ако је све у реду, односно -1 ако је дошло до неке грешке.

Аргументи функције су јој редом:

`stream` - фајл у оквиру кога се врши подешавање,

`offset` - вредност у бајтовима - може бити и позитивна и негативна величина,

`whence` - параметар који назначавачу у односу на шта се врши подешавање, може имати вредности:

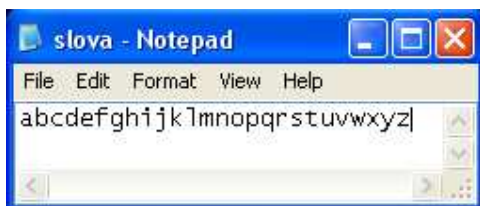
`SEEK_SET` - подешавање у односу на почетак фајла,

`SEEK_CUR` - подешавање у односу на текућу позицију у фајлу,

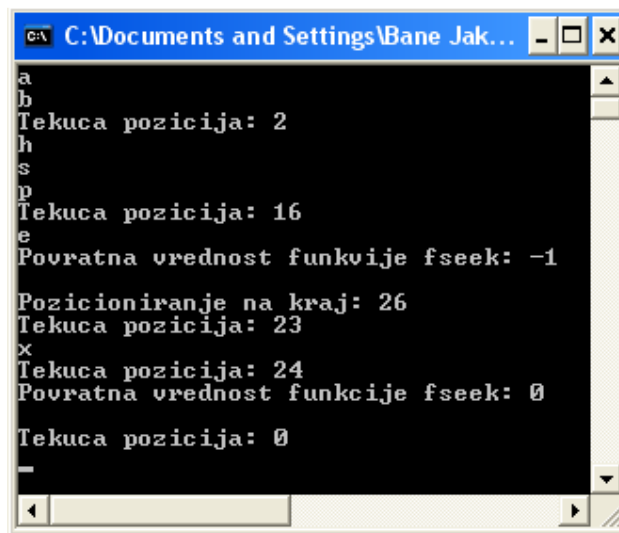
`SEEK_END` - подешавање у односу на крај фајла.

Функција `long ftell(FILE* stream)` враћа текућу позицију у фајлу, повратна вредност је или текући `offset` или -1 за случај грешке.

Функција `void rewind(FILE* stream)` је функција која поставља позицију у фајлу на 0 тј. на његов почетак.



Изглед датотеке *slova.txt*



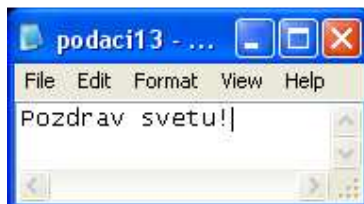
Испис на екрану

5.13. Саставити програм који у датотеку **podaci13.txt** уписује текст 'Pozdrav svima!', а затим од позиције 11 (позиције која је 10 бајтова удаљена од почетка датотеке, односно првог карактера) уписује текста 'etu'.

```

#include <stdio.h>

main()
{
    FILE *dato;
    dato=fopen("podaci13.txt","w");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fputs("Pozdrav svima!", dato);
    fseek(dato, 10, SEEK_SET);
    fputs("etu", dato);
    fclose(dato);
    getch();
    return 0;
}
  
```



Изглед датотеке *podaci13.txt* након извршавања програма

5.14. Саставити програм којим се у датотеку **podaci14.txt** уносе три цела броја, а затим прочита те бројеве и њихов збир испишује на екрану. Задатак решити употребом функција:

- a) **fprintf()** и **fscanf()**,
б) **fwrite()** и **fread()**.

a)

```
#include <stdio.h>

main()
{
    FILE *ulaz, *izlaz;
    int i, broj, suma=0;
    if((ulaz=fopen("podaci14.dat", "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    printf("\n Unesite tri broja u datoteci: ");
    for(i=0; i<3; i++)
    {
        scanf("%d",&broj);
        fprintf(ulaz,"%d ",broj);
    }
    fclose(ulaz);
    getche();
    if((izlaz=fopen("podaci14.dat", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    for(i=0; i<3; i++)
    {
        fscanf(izlaz,"%d",&broj);
        suma+=broj;
    }
    printf("\n Suma brojeva u datoteci: %d",suma);
    fclose(izlaz);
    getche();
    return 0;
}
```

б)

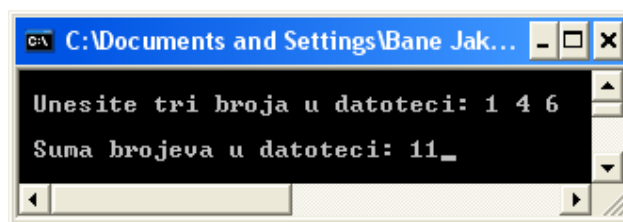
```
#include <stdio.h>

main()
{
    FILE *ulaz, *izlaz;
    int i, broj, suma=0;
    if((ulaz=fopen("podaci14.txt", "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    printf("\n Unesite tri broja u datoteci: ");
    for(i=0; i<3; i++)
    {
        scanf("%d",&broj);
        fwrite(&broj, sizeof(broj),1,ulaz);
    }
}
```

```
fclose(ulaz);
getche();
if((izlaz=fopen("podaci14.txt", "r"))==NULL)
{
    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
for(i=0; i<3; i++)
{
    fread(&broj, sizeof(int),1,izlaz);
    suma+=broj;
}
printf("\n Suma brojeva u datoteci: %d",suma);
fclose(izlaz);
getche();
return 0;
}
```



Изглед датотеке **podaci14.txt** након извршавања програма



Испис на екрану

Функција `fwrite()` служи за запис у бинарну датотеку произвољног броја бајта с неке меморијске локације. Прототип функције је:

```
int fwrite(&buf, int size, int count, fp);
```

Аргументи имају следећа значења:

- `buf` је адреса меморијске локације с које се подаци записују у датотеку `fp`,
- `size` означава величину у бајтима појединог елемента који се уписује,
- `count` означава укупни број елемената који се записују.

Функција враћа вредност која је једнака броју елемената који су успешно записани. Ако је та вредност различита од `count`, то значи да је настала грешка.

Функција `fread()` служи за читавање произвољног броја бајта на неку меморијску локацију. Прототип функције је:

```
int fread(&buf, int size, int count, fp);
```

Аргументи имају следећа значења:

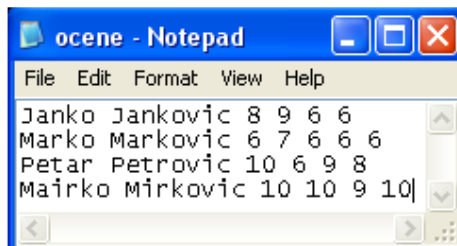
- `buf` је адреса меморијске локације у коју се уписују подаци из датотеке `fp`,
- `size` означава величину у бајтима појединог елемента који се читава,
- `count` означава укупни број елемената који се читавају у меморију.

Функција враћа вредност која је једнака броју елемената који су успешно читани. Ако је та вредност различита од `count`, то значи да је настала грешка или је достигнут крај датотеке.

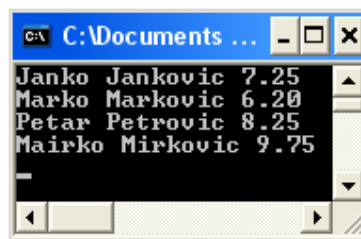
5.15. Дата је датотеке **ocene.txt** која садржи имена и презимена студената и њихове оцене. Саставити програм који за сваког студента израчунава и исписује на екрану просечну оцену.

```
#include <stdio.h>

main()
{
    FILE *dato;
    float prosek;
    int ocena, brOcena;
    char ime[20];
    char prezime[20];
    dato=fopen("ocene.txt", "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(fscanf(dato, "%s", ime) != EOF)
    {
        fscanf(dato, "%s", prezime);
        brOcena= 0;
        prosek=0.0;
        while(!feof(dato) &&
            fscanf(dato, "%d", &ocena)>0)
        {
            brOcena++;
            prosek += ocena;
        }
        if(brOcena>0)
            prosek/=brOcena;
        printf("%s %s %.2f\n", ime, prezime, prosek);
    }
    fclose(dato);
    getche();
    return 0;
}
```



Изглед датотеке *ocene.txt*



Испис на екрану

5.2 Датотеке са низовима и матрицама

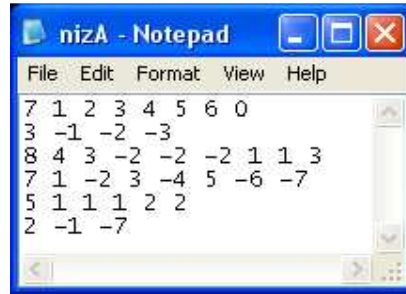
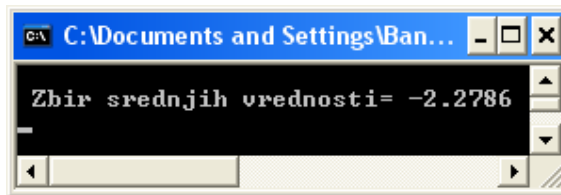
5.16. У датотеци **nizA.txt** у сваком реду се налази један цео број **n** и **n** реалних бројева. Саставити програм који у нову секвенцијалну датотеку **nizB.txt** упише оне редове из почетне датотеке у којим је средња вредност реалних бројева већа од нуле и на крају испише на главном излазу збир средњих вредности реалних бројева у свим редовим почетне датотеке.

```

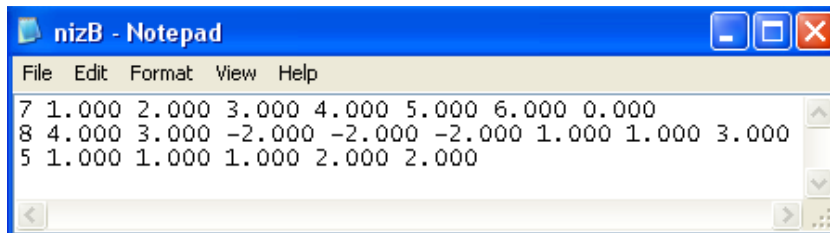
#include <stdio.h>
#define N 100

main()
{
    int i, n;
    double a[N], s, z=0;
    FILE *ul, *iz;
    ul=fopen("nizA.txt", "r");
    if(ul==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    iz=fopen("nizB.txt", "w");
    if(iz==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(fscanf(ul,"%d",&n)>0)
    {
        s=0;
        for(i=0; i<n; i++)
        {
            fscanf(ul,"%lf",&a[i]);
            s+=a[i];
        }
        if(n)
            s/=n;
        z+=s;
        if(s>0)
        {
            fprintf(iz,"%d",n);
            for(i=0; i<n; i++)
                fprintf(iz, " %.3f", a[i]);
            fprintf(iz, "\n");
        }
    }
    fclose(ul); fclose(iz);
    printf("\n Zbir srednjih vrednosti= %.4f\n", z);
    getche();
    getche();
    return 0;
}

```

Изглед датотеке *nizA.txt*

Испис на екрану

Изглед датотеке *nizB.txt* након извршавања програма

5.17. Саставити функцију за претварање низа целих бројева (међу чијим елементима могу да буду и једнаки) у скуп чији су сви елементи различити. Затим саставити програм који прочита низ целих бројева из датотеке *niz.txt*, претвори га у скуп, испише добијени резултат у датотеку *skup.txt*. У датотеци *niz.txt* у првом реду се налази број елмената низа, а у другом елементи. Имена датотека учитати на главном улазу.

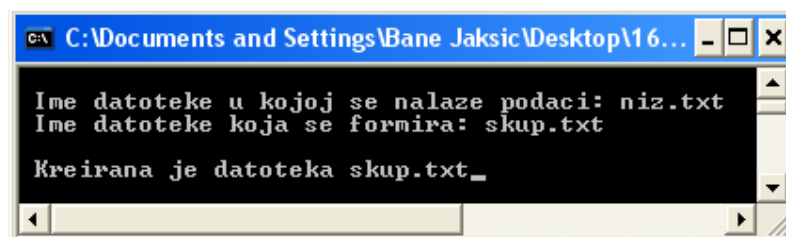
```

#include <stdio.h>
#define MAX 100

void Uskup(int a[], int *n)
{
    int i, j, k=0;
    for(i=0; i<*n; i++)
    {
        for(j=0; j<k && a[j]!=a[i]; j++);
        if(j==k)
            a[k++]=a[i];
    }
    *n=k;
}

main()
{
    int a[MAX], i, n;
    FILE *dato1, *dato2;
    char ime1[20], ime2[20];
    printf("\n Ime datoteke u kojoj se nalaze podaci: ");
    scanf("%s", &ime1);
    printf(" Ime datoteke koja se formira: ");
    scanf("%s", &ime2);
    dato1=fopen(ime1, "r");
    if(dato1==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    dato2=fopen(ime2, "w");
    if(dato2==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fscanf(dato1, "%d", &n);
    fprintf(dato2, "\nNiz= ");
    for(i=0; i<n; i++)
    {
        fscanf(dato1, "%d", &a[i]);
        fprintf(dato2, "%d ", a[i]);
    }
    Uskup(a, &n);
    fprintf(dato2, "\nSkup= ");
    for(i=0; i<n; i++)
        fprintf(dato2, "%d ", a[i]);
    fputc('\n', dato2);
    fclose(dato1);
    fclose(dato2);
    printf("\n Kreirana je datoteka %s", ime2);
    getche();
    return 0;
}

```

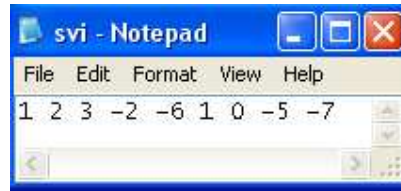
Изглед датотеке *niz.txt*Изглед датотеке *skup.txt* након извршавања програма

Испис на екрану

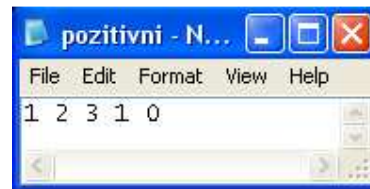
5.18. Саставити програм којим се формирају три датотеке са низовима целих бројева. Број и елементи низа се уносе са тастатуре, а затим се сви унети смештају у датотеку **svi.txt**, позитивни елементи у **pozitivni.txt** и негативни у **negativni.txt**. Имена датотека се уносе на главном улазу.

```
#include <stdio.h>
#define MAX 100

main()
{
    int a[MAX];
    int i, n;
    FILE *dato1, *dato2, *dato3;
    char ime1[20], ime2[20], ime3[20];
    printf("\n Ime datoteke za sve brojeve: ");
    scanf("%s", ime1);
    printf(" Ime datoteke za pozitivne brojeve: ");
    scanf("%s", ime2);
    printf(" Ime datoteke za negativne brojeve: ");
    scanf("%s", ime3);
    dato1=fopen(ime1, "w");
    dato2=fopen(ime2, "w");
    dato3=fopen(ime3, "w");
    if(dato1==NULL || dato2==NULL || dato3==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    printf(" Broj elemenata niza: ");
    scanf("%d", &n);
    printf(" Elementi niza: ");
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
        fprintf(dato1,"%d ",a[i]);
        if(a[i]>=0)
            fprintf(dato2,"%d ",a[i]);
        else
            fprintf(dato3,"%d ",a[i]);
    }
    close(dato1);
    close(dato2);
    close(dato3);
    getche();
    return 0;
}
```



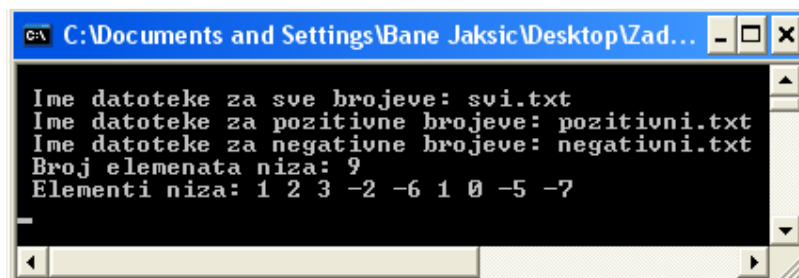
Изглед датотеке **svi.txt** након извршавања програма



Изглед датотеке **pozitivni.txt** након извршавања програма



Изглед датотеке **negativni.txt** након извршавања програма



Испис на екрану

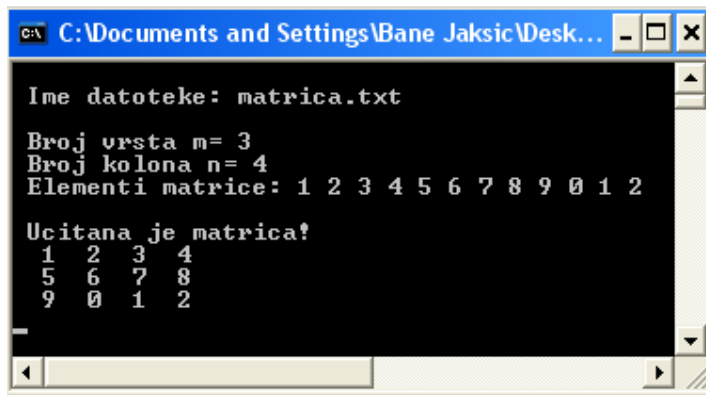
5.19. Саставити програм који формира матрицу целих бројева диманзија **$m \times n$** и уписује у датотеку **matrica.txt**. Уписивање се врши тако што се у први ред упишу **m** и **n**, а после тога елементи матрице у сваком реду по једна врста. Исписати матрицу у облику таблице на екрану. Име датотеке се уноси са главног улаза.

```
#include <stdio.h>

main()
{
    int i, j, m, n, a[20][20];
    FILE *dato;
    char ime[20];
    printf("\n Ime datoteke: ");
    scanf("%s",&ime);
    printf("\n Broj vrsta m= ");
    scanf("%d",&m);
    printf(" Broj kolona n= ");
    scanf("%d",&n);
    if((dato=fopen(ime,"w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fprintf(dato,"%d%d\n",n,m);
    printf(" Elementi matrice: ");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d",&a[i][j]);
            fprintf(dato,"%d\t",a[i][j]);
        }
        fprintf(dato,"\n");
    }
    fclose(dato);
    printf("\n Ucitana je matrica!\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            printf("%3d",a[i][j]);
        printf("\n");
    }
    getch();
    return 0;
}
```



Изглед датотеке **matrica.txt** након извршавања програма

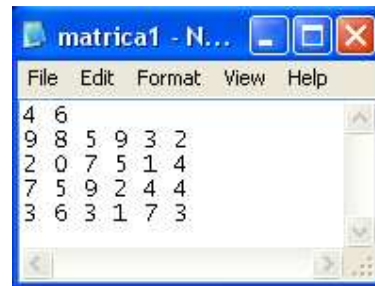


Испис на екрану

5.20. Саставити програм којим се из правоугаоне матрице целих бројева изоставља врста и колона које садрже највећи елемент матрице. Подаци о почетној матрици се налазе у датотеци **matrica1.txt**. У првом реду се налазе број врста и колона, а у наставку по једна врста матрице у сваком реду. Резултат треба уписати у другу ну датотеку **matrica2.txt**. Формат записивања матрице у датотеци **matrica2.txt** треба да је исти као у **matrica1.txt**. Имена датотека се уносе на главном улазу.

```
#include <stdio.h>

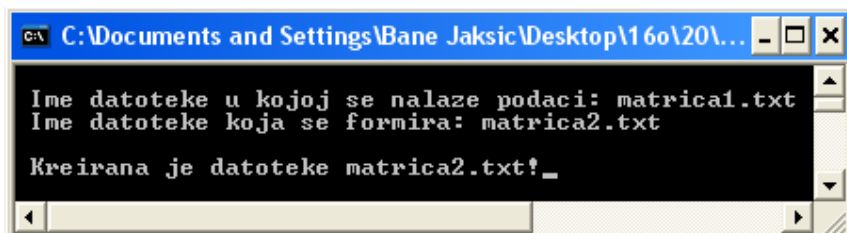
main()
{
    int i, j, m, n, a[30][30], max, imax, jmax;
    FILE *dato;
    char ime1[20], ime2[20];
    printf("\n Ime datoteke u kojoj se nalaze podaci: ");
    scanf("%s", &ime1);
    printf(" Ime datoteke koja se formira: ");
    scanf("%s", &ime2);
    dato=fopen(ime1, "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fscanf(dato, "%d%d", &m, &n);
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            fscanf(dato, "%d", &a[i][j]);
    fclose(dato);
    /*Nalazenje mesta najveceg elementa: */
    max=a[0][0];
    imax=jmax=0;
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            if(a[i][j]>max)
            {
                max=a[i][j];
                imax=i;
                jmax=j;
            }
    /*Izostavljanje kolone:*/
    for(i=0; i<m; i++)
        for(j=jmax; j<n-1; j++)
            a[i][j]=a[i][j+1];
    n--;
    /*Izostavljanje vrste:*/
    for(j=0; j<n; j++)
        for(i=imax; i<m-1; i++)
            a[i][j]=a[i+1][j];
    m--;
    /*Upisivanje rezultata u datoteku:*/
    dato=fopen(ime2, "w");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fprintf(dato, "%d %d\n", m, n);
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
```

Изглед датотеке **matrica1.txt**Изглед датотеке **matrica2.txt** након извршавања програма


```

        fprintf(dato, "%d ", a[i][j]);
        fputc('\n', dato);
    }
    fclose(dato);
    printf("\n Kreirana je datoteke %s!", ime2);
    getch();
    return 0;
}

```



Испис на екрану

5.21. Саставити програм за међусобну замену најмањег и највећег елемента правоугаоне матрице целих бројева димензија $m \times n$ међусобном заменом одговарајуће две врсте и две колоне матрице. Подаци о почетној матрици се налазе у датотеци **MatricaPre.txt**. У првом реду се налазе број врста и колоне, а у наставку по једна врста матрице у сваком реду. Резултат треба уписати у датотеци **MatricaPosle.txt**. Формат записивања матрице у датотеци **MatricaPosle.txt** треба да је исти као у датотеци **MatricaPre.txt**. Имена датотека се уносе на главном улазу.

```

#include <stdio.h>

main()
{
    int i, j, m, n, a[30][30];
    int min, max, imin, jmin, imax, jmax, b;
    FILE *dato;
    char imel[20], ime2[20];
    printf("\n Ime datoteke u kojoj se nalaze podaci: ");
    scanf("%s", &imel);
    printf(" Ime datoteke koja se formira: ");
    scanf("%s", &ime2);
    dato=fopen(imel, "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fscanf(dato, "%d%d", &m, &n);
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            fscanf(dato, "%d", &a[i][j]);
    fclose(dato);
    /*Nalazenje mesta najmanjeg i najveceg elementa:*/
    min=max=a[0][0];
    imin=jmin=imax=jmax=0;
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            if(a[i][j]<min)
            {
                min=a[i][j];
                imin=i;
            }
}

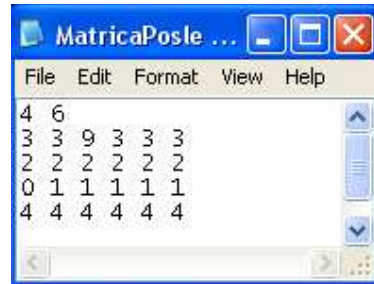
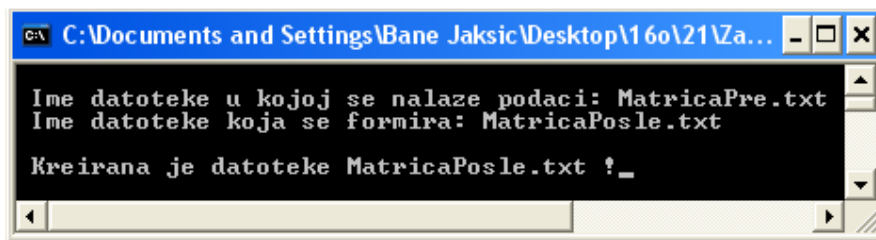
```

```

        jmin=j;
    }
    else if(a[i][j]>max)
    {
        max=a[i][j];
        imax=i;
        jmax=j;
    }

    /*Medjusobna zamena dve vrste i dve kolone:*/
    for(i=0; i<m; i++)
    {
        b=a[i][jmin];
        a[i][jmin]=a[i][jmax];
        a[i][jmax]=b;
    }
    for(j=0; j<n; j++)
    {
        b=a[imin][j];
        a[imin][j]=a[imax][j];
        a[imax][j]=b;
    }
    /*Upisivanje rezultata u datoteku:*/
    dato=fopen(ime2, "w");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    fprintf(dato, "%d %d\n", m, n);
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
            fprintf (dato, "%d ", a[i][j]);
        fputc ('\n', dato);
    }
    fclose(dato);
    printf("\n Kreirana je datoteke %s !", ime2);
    getche();
    return 0;
}

```

Изглед датотеке *MatricaPre.txt*Изглед датотеке *MatricaPosle.txt* након извршавања програма

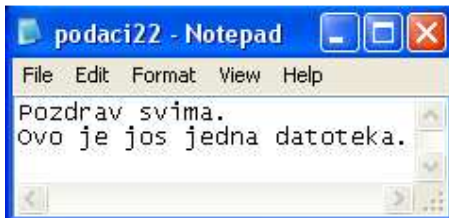
Испис на екрану

5.3 Датотеке са стринговима

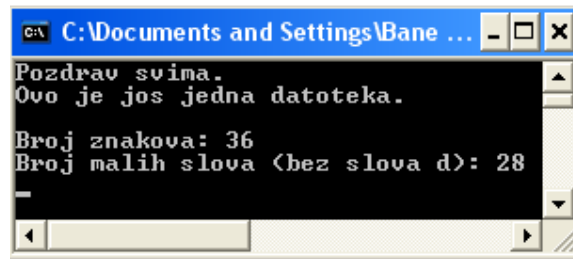
5.22. Саставити програм који из датотеке **podaci22.txt** чита и исписује садржај текста на екрану и одређује укупан број знакова и број малих слова без слова 'd'.

```
#include <stdio.h>

main()
{
    FILE *dato;
    int c, brZnakova=0, brMalihSlova=0;
    if((dato=fopen("podaci22.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while((c=getc(dato)) != EOF)
    {
        putchar(c);
        if(c!=' ')
            brZnakova++;
        if(c>='a' && c<='z' && c!='d')
            brMalihSlova++;
    }
    printf("\nBroj znakova: %d", brZnakova);
    printf("\nBroj malih slova (bez slova d): %d.\n", brMalihSlova);
    fclose(dato);
    getch();
    return 0;
}
```



Изглед датотеке **podaci22.txt**



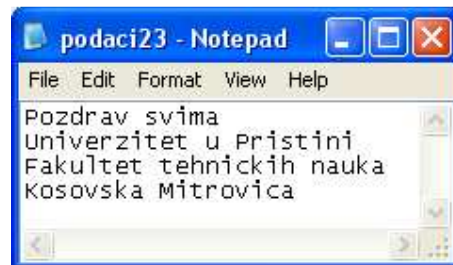
Испис на екрану

5.23. Саставити програм који на екрану исписује број речи, редова и знакова који се налазе у датотеци **podaci23.txt**.

```
#include <stdio.h>

typedef enum {NETACNO, TACNO} LOGIC;

main()
{
    int znak, brRed=1, brZnak=0, brRec=0;
    FILE *dato;
    LOGIC Ureci=NETACNO;
    dato=fopen("podaci23.txt", "r");
    Ureci=NETACNO;
    if(dato==NULL)
```

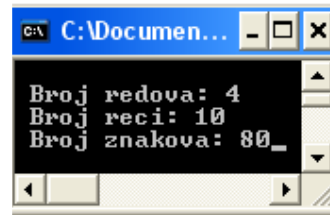


Изглед датотеке **podaci23.txt**

```

{
    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
while((znak=getc(dato))!=EOF)
{
    brZnak++;
    if(znak==' ' || znak=='\t' || znak=='\n')
        Ureci=NETACNO;
    else if(Ureci==NETACNO)
    {
        brRec++;
        Ureci=TACNO;
    }
    if(znak=='\n') brRed++;
}
printf("\n Broj redova: %d", brRed);
printf("\n Broj reci: %d", brRec);
printf("\n Broj znakova: %d", brZnak);
getche();
return 0;
}

```



Испис на екрану

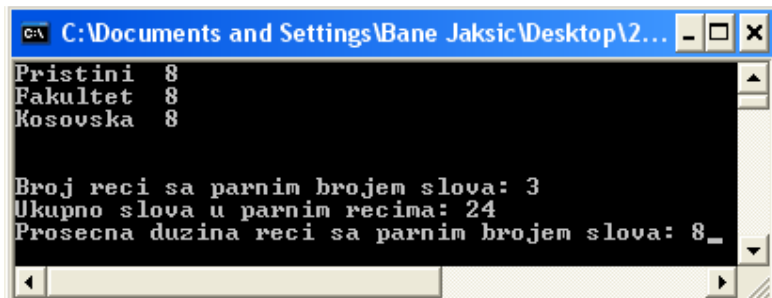
5.24. Саставити програм који из датотеке **podaci23.txt** чита речи и одређује и испишује на екрану број речи са парним бројем слова, просечан број слова у речима парне дужине, укупан број слова кога чине речи парне дужине. На екрану исписати и речи са парним бројем слова.

```

#include <stdio.h>
#include <string.h>

main()
{
    FILE *dato;
    char rec[20];
    int duzina, ukupnaDuzina=0, prosek, brReci=0;
    if((dato=fopen("podaci23.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(!feof(dato))
    {
        fscanf(dato, "%s", rec);
        duzina=(int)strlen(rec);
        if(duzina%2==0)
        {
            ukupnaDuzina+=duzina;
            brReci++;
            printf("%s  %d\n", rec, duzina);
        }
    }
    prosek=ukupnaDuzina/brReci;
    printf("\n\nBroj reci sa parnim brojem slova: %d ", brReci);
    printf("\nUkupno slova u parnim recima: %d", ukupnaDuzina);
    printf("\nProsečna duzina reci sa parnim brojem slova: %d", prosek);
    fclose(dato);
    getche(); return 0;
}

```

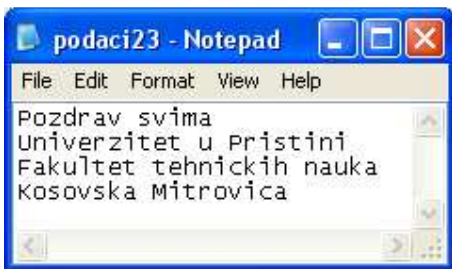
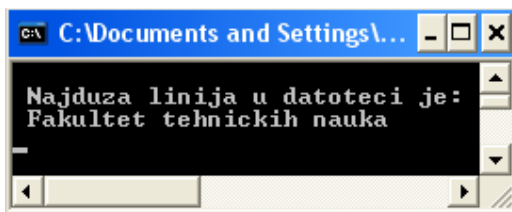


Испис на екрану

5.25. Саставити програм који на екрану испишује најдужу линију која се налази у датотеци **podaci23.txt**.

```
#include <stdio.h>
#include <string.h>
#define MAX 50

main()
{
    FILE *dato;
    char maxs[30], string[30];
    strcpy(maxs, "");
    if((dato=fopen("podaci23.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(fgets(string, MAX, dato)!=NULL)
        if(strlen(maxs)<strlen(string))
            strcpy(maxs, string);
    printf("\n Najduza linija u datoteci je:\n %s", maxs);
    fclose(dato);
    getch();
    return 0;
}
```

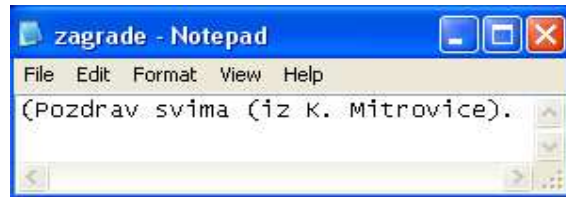
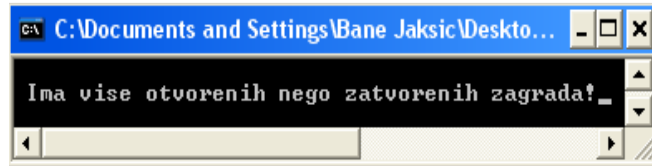
Изглед датотеке **podaci23.txt**

Испис на екрану

5.26. Саставити програм који испишује обавештење на екрану да ли се тексту у датотеци **zagrade.txt** заграде добро упарене.

```
#include <stdio.h>

main()
{
    FILE *dato;
    int c, brOtv=0, brZatv=0;
    dato=fopen("zgrade.txt", "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while((c=getc(dato))!=EOF)
    {
        if(c=='(') brOtv++;
        if(c==')') brZatv++;
    }
    if(brOtv > brZatv)
        printf("\n Ima vise otvorenih nego zatvorenih zagrada!");
    if(brOtv < brZatv)
        printf("\n Ima vise zatvorenih nego otvorenih zagrada!");
    if(brOtv == brZatv)
        printf("\n Zgrade su ispravno zapisane!");
    fclose(dato);
    getch();
    return 0;
}
```

Изглед датотеке *zgrade.txt*

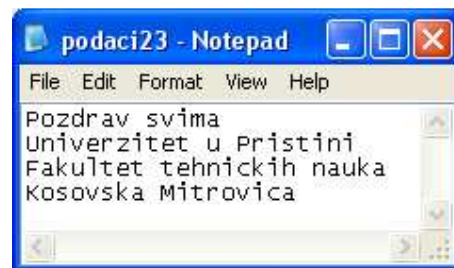
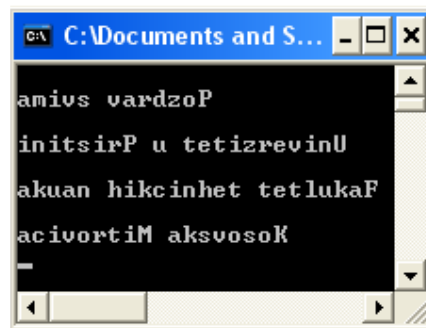
Испис на екрану

5.27. Саставити функцију која исписује на екрану садржај датотеке тако што сваку линију исписује у обрнутом редоследу. Саставити програм који тестира претходну функцију над датотеком **podaci23.txt**.

```
#include <stdio.h>
#define MAX 500

void Stampaj(char *s)
{
    char *s1;
    s1=s;
    while(*s1)
        s1++;
    s1--;
    while(s1 >= s)
    {
        putchar(*s1);
        s1--;
    }
    putchar('\n');
}

main()
{
    FILE *dato;
    int max;
    char s[MAX];
    if((dato=fopen("podaci23.txt", "r"))==NULL)
    {
```

Изглед датотеке *podaci23.txt*

Испис на екрану

```

    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
while(fgets(s, MAX, dato) != NULL)
    Stampaj(s);
fclose(dato);
getche();
return 0;
}

```

5.28. Саставити програм који врши преписивање садржаја датотеке **pogresan.txt** у датотеку **ispravan.txt** уз претварање почетних слова реченице у велика, а свих осталих слова у мала. Крај реченице се обележава тачком (.), знаком узвика (!) или знаком питања (?).

```

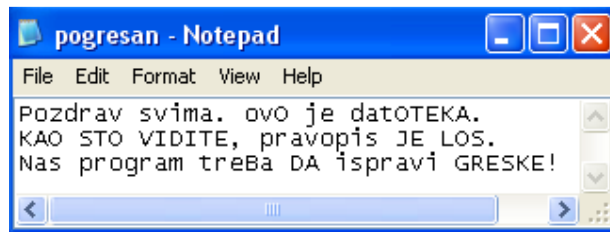
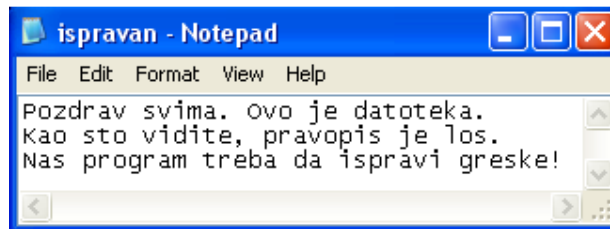
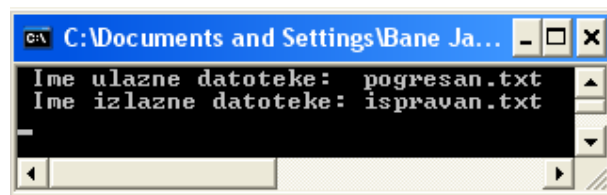
#include <stdio.h>
#include <ctype.h>
#include <string.h>

```

```

main()
{
    int zn, prvi=1;
    FILE *ulaz, *izlaz;
    char ime1[30], ime2[30];
    printf(" Ime ulazne datoteke: ");
    scanf("%s", ime1);
    printf(" Ime izlazne datoteke: ");
    scanf("%s", ime2);
    ulaz=fopen(ime1, "r");
    izlaz=fopen(ime2, "w");
    while((zn=fgetc(ulaz)) != EOF)
    {
        if(isupper(zn))
        {
            if(!prvi) zn+='a'-'A';
            else prvi=0;
        }
        else if(islower(zn))
        {
            if(prvi)
            {
                zn+='A'-'a';
                prvi=0;
            }
        }
        else if(zn=='.' || zn=='!' || zn=='?')
            prvi=1;
        fputc(zn, izlaz);
    }
    fclose(ulaz);
    fclose(izlaz);
    getche();
    return 0;
}

```

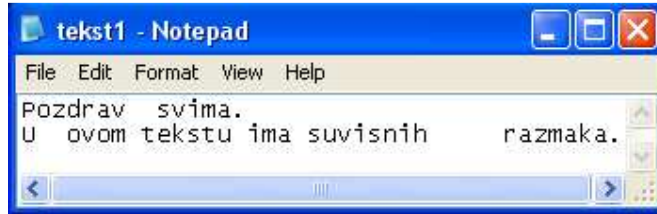
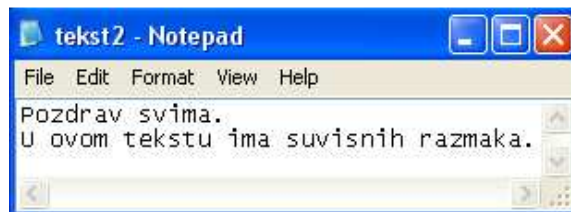
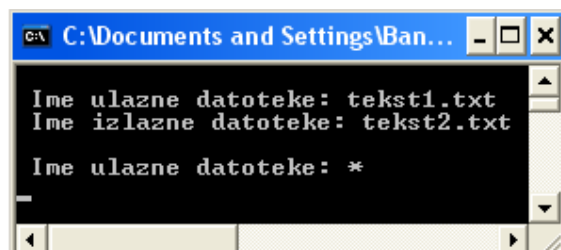
Изглед датотеке **pogresan.txt**Изглед датотеке **ispravan.txt** након извршавања програма

Испис на екрану

5.29. Саставити програм који врши преписивање садржаја датотеке **tekst1.txt** у другу датотеку **tekst2.txt** уз изостављање сувишних размака између речи. Уредност текста у редове треба да се очува. Имена датотека се уносе са главног улаза. Програм обрађује произвољан број датотека, све док за име датотеке не прочита '*'.

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

main()
{
    char ime1[20], ime2[20];
    FILE *ulaz, *izlaz;
    int zn, ima=1;
    while(1)
    {
        printf("\n Ime ulazne datoteke: ");
        scanf("%s", ime1);
        if(ime1[0]=='*') break;
        printf (" Ime izlazne datoteke: ");
        scanf ("%s", ime2);
        if(ime2[0]=='*') break;
        if((ulaz=fopen(ime1, "r"))==NULL)
        {
            printf("\n Greska pri otvaranju datoteke!");
            exit(1);
        }
        if((izlaz=fopen(ime2, "w"))==NULL)
        {
            printf("\n Greska pri otvaranju datoteke!");
            exit(1);
        }
        while((zn=fgetc(ulaz)) != EOF)
        {
            if(! isspace(zn))
            {
                fputc(zn, izlaz);
                ima=0;
            }
            else
            {
                if(zn == '\n') fputc('\n', izlaz);
                else if(! ima) fputc(' ', izlaz);
                ima=1;
            }
        }
        fclose(izlaz);
        fclose(ulaz);
    }
    getche();
    return 0;
}
```

Изглед датотеке *tekst1.txt*Изглед датотеке *tekst2.txt* након извршавања програма

Испис на екрану

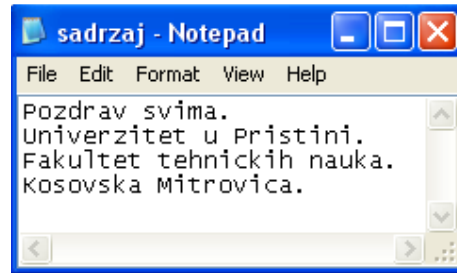
5.30. Саставити програм који читава линије из датотеке **sadrzaj.txt**, а затим их исписује на екрану сортиране по дужини, почев од најкраће линије.

```
#include <stdio.h>
#include <string.h>
#define MAX_RED 1024
#define MAX_REDOVA 1024

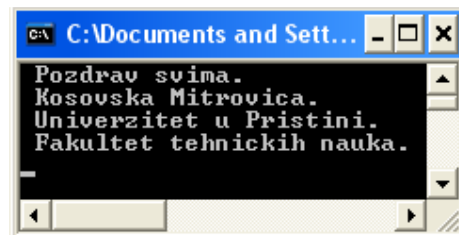
/*Funkcija razmenjuje vrednosti dva pokazivaca na karaktere.
S obzirom da je potrebno preneti adrese ovih promenljivih,
parametri su tipa "pokazivac na pokazivac na char".*/
void Razmeni(char **s, char **t)
{
    char *p;
    p=*s;
    *s=*t;
    *t=p;
}

/*Funkcija implementira uobicajeni algoritam sortiranja izborom
najmanjeg elementa, pri cemu se pod najmanjim elementom ovde
podrazumeva pokazivac koji pokazuje na string koji je najkraci*/
void Sortiraj(char *redovi[], int n)
{
    int i, j, min;
    for(i=0; i<n-1; i++)
    {
        min=i;
        for(j=i+1; j<n; j++)
            if(strlen(redovi[j]) < strlen(redovi[min]))
                min=j;
        if(min != i)
            Razmeni(&redovi[min], &redovi[i]);
    }
}

main()
{
    char redovi[MAX_REDOVA][MAX_RED];
    char *predovi[MAX_REDOVA];
    int i, n;
    FILE *dato;
    dato=fopen("sadrzaj.txt", "r");
    if(dato==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    /*Citamo linije, i smestamo ih u dvodimenzioni niz karaktera redovi[] (i-tu
liniju smestamo u niz redovi[i]). Citamo najvise MAX_REDOVA. Pokazivac
predovi[i] postavljamo da pokazuje na prvi karakter u nizu redovi[i].*/
    for(i = 0; i < MAX_REDOVA; i++)
        if(fgets(redovi[i], MAX_RED, dato) != NULL)
            predovi[i] = redovi[i];
        else break;
    n=i;
    fclose(dato);
    /*Sortiramo niz pokazivaca*/
    Sortiraj(predovi, n);
    for(i=0; i<n; i++)
        printf(" %s", predovi[i]);
    getch(); return 0;
}
```



Изглед датотеке *sadrzaj.txt*



Испис на екрану

5.31. Саставити програм који из датотеке **sadrzaj.txt** чита текст и сортира низ ниски:

- лексикографски,
- дужински,
- дужински, при чему ниске исте дужине се сортирају лексикографски.

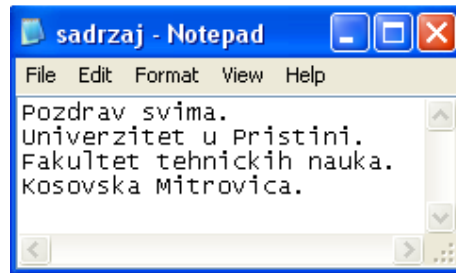
Сортиране ниске редом сместити у новокреиране датотеке, редом, **leksiko.txt**, **duzinski.txt** и **duz-leksiko.txt**. Сортиране ниске истовремено исписати и на екрану. За сортирање ниски формирати одговарајуће функције. Имена датотека се уносе на главном улазу.

```
#include <stdio.h>
#include <string.h>
#define MAX_NISKI 1000
#define MAX_DUZINA 30

void Leksikografski(char niske[][MAX_DUZINA], int n)
{
    int i, j, min;
    char pom[MAX_DUZINA];
    for(i=0; i<n-1; i++)
    {
        min = i;
        for(j=i+1; j<n; j++)
            if(strcmp(niske[j], niske[min])<0)
                min = j;
        if(min != i)
        {
            strcpy(pom,niske[i]);
            strcpy(niske[i], niske[min]);
            strcpy(niske[min], pom);
        }
    }
}

void Duzinski(char niske[][MAX_DUZINA], int n)
{
    int i, j, r, min;
    char pom[MAX_DUZINA];
    for (i = 0; i < n - 1; i++)
    {
        min=i;
        for(j=i+1; j<n; j++)
            if((r=strlen(niske[j])-strlen(niske[min]))<0)
                min=j;
        if(min != i)
        {
            strcpy(pom,niske[i]);
            strcpy(niske[i], niske[min]);
            strcpy(niske[min], pom);
        }
    }
}

void DuzinskiLeksikografski(char niske[][MAX_DUZINA], int n)
{
    int i, j, r, min;
    char pom[MAX_DUZINA];
    for(i=0; i<n-1; i++)
    {
        min=i;
```



Изглед датотеке *sadrzaj.txt*

```

        for(j=i+1; j<n; j++)
            if((r=strlen(niske[j])-strlen(niske[min]))<0)
                min = j;
                else if(r==0 && strcmp(niske[j], niske[min])<0)
                    min=j;
            if(min != i)
            {
                strcpy(pom,niske[i]);
                strcpy(niske[i], niske[min]);
                strcpy(niske[min], pom);
            }
        }
    }
}

main()
{
    int i, n;
    char niske[MAX_NISKI][MAX_DUZINA];
    char ime[30], imel[30], imed[30], imedl[30];
    FILE *ul, *iz1, *iz2, *iz3;
    printf("\n Ime datoteke u kojoj se nalaze podaci: ");
    scanf("%s", &ime);
    printf(" Ime datoteke za leksikografsko sortiranje: ");
    scanf("%s", &imel);
    printf(" Ime datoteke za duzinsko sortiranje: ");
    scanf("%s", &imed);
    printf(" Ime datoteke za duzinsko-leksikog. sortiranje: ");
    scanf("%s", &imedl);
    if((ul=fopen(ime, "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    if((iz1=fopen(imel, "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    if((iz2=fopen(imed, "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    if((iz3=fopen(imedl, "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    for(i=0; fscanf(ul, "%s", niske[i] )!=EOF; i++);
    n=i;
    Leksikografski(niske,n);
    printf("\n\n -Leksikografski sortirane niske-\n");
    for(i=0; i<n; i++)
    {
        printf(" %s\n",niske[i]);
        fprintf(iz1, "%s\n", niske[i]);
    }
    Duzinski(niske, n);
    printf("\n\n -Po duzini sortirane niske-\n");
    for(i=0; i<n; i++)
    {

```

```
    printf(" %s\n",niske[i]);
    fprintf(iz2, "%s\n", niske[i]);
}
DuzinskiLeksikografski(niske, n);
printf("\n\n -Sortirane niske po duzini, pa leksikografski-\n");
for(i=0; i<n; i++)
{
    printf(" %s\n",niske[i]);
    fprintf(iz3, "%s\n", niske[i]);
}
fclose(ul);
fclose(iz1);
fclose(iz2);
fclose(iz3);
getche();
return 0;
}
```

```
C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak31.exe

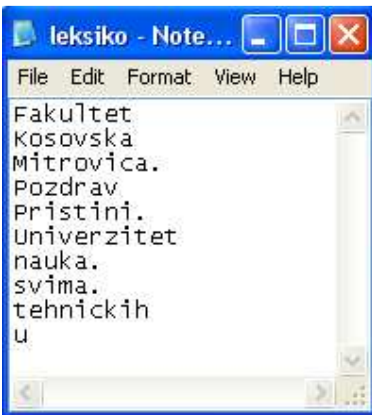
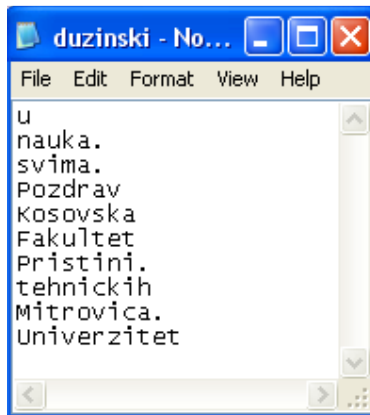
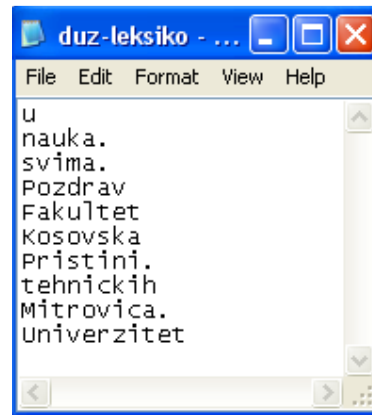
Ime datoteke u kojoj se nalaze podaci: sadrzaj.txt
Ime datoteke za leksikografsko sortiranje: leksiko.txt
Ime datoteke za duzinsko sortiranje: duzinski.txt
Ime datoteke za duzinsko-leksikog. sortiranje: duz-leksiko.txt

-Leksikografski sortirane niske-
Fakultet
Kosovska
Mitrovica.
Pozdrav
Pristini.
Univerzitet
nauka.
svima.
tehnickih
u

-Po duzini sortirane niske-
u
nauka.
svima.
Pozdrav
Kosovska
Fakultet
Pristini.
tehnickih
Mitrovica.
Univerzitet

-Sortirane niske po duzini, pa leksikografski-
u
nauka.
svima.
Pozdrav
Fakultet
Kosovska
Pristini.
tehnickih
Mitrovica.
Univerzitet
```

Изпис на екрану

Изглед датотеке *leksiko.txt*, након извршавања програмаИзглед датотеке *duzinski.txt* након извршавања програмаИзглед датотеке *duz-leksiko.txt* након извршавања програма

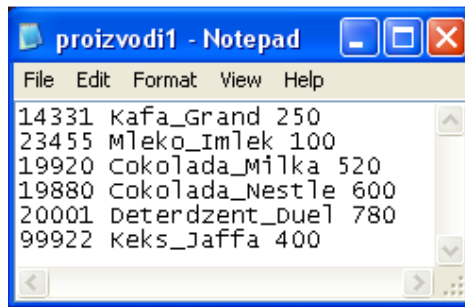
5.4 Датотеке са структурама

5.32. Датотека **proizvodi1.txt** садржи податке о атриклима у продавници. Сваки артикал се одликује следећим карактеристикама: бар код (петодигитни број), име и цена. Саставити програм који чита податке о производима из датотеке и на екрану исписује податке о производима чија је цена мања од 500 динара.

```
#include <stdio.h>
#define MAX 100

typedef struct artikli
{
    int barKod;
    char ime[30];
    float cena;
} ARTIKLI;

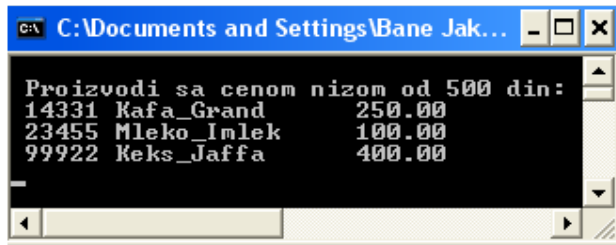
main()
{
    ARTIKLI artikal[MAX];
    int i=0;
    FILE *dato;
    if((dato=fopen("proizvodi.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    printf("\n Proizvodi sa cenom nizom od 500 din:\n");
    while(1)
    {
        fscanf(dato, "%d%s%f",
               &artikal[i].barKod, &artikal[i].ime, &artikal[i].cena);
        if(feof(dato)) break;
        if(artikal[i].cena<500)
        {
            printf ("%6d %-15s %4.2f\n",
```

Изглед датотеке *proizvodi1.txt*

```

        artikal[i].barKod, artikal[i].ime, artikal[i].cena);
    i++;
}
fclose(dato);
getche();
return 0;
}

```



Испис на екрану

5.33. Датотека **proizvodiPDV.txt** садржи податке о производима који се продају у оквиру одређене продавнице. Сваки производ се одликује следећим подацима: бар-код, име, цена (без ПДВ-а) и ПДВ. Формирати датотеку **proizvodiPDVNovo.txt** у којој се смештају подаци о свим производима из датотеке **proizvodiPDV.txt** са крајњом ценом (крајња цена = цена*(1+ПДВ)).

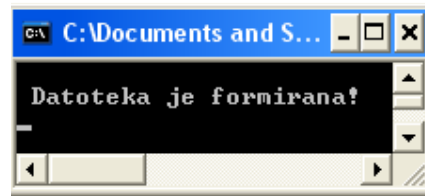
```

#include <stdio.h>
#define MAX 100

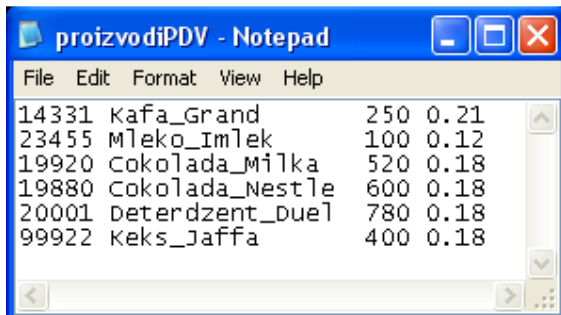
typedef struct artikli
{
    int barKod;
    char ime[30];
    float cena;
    float PDV;
} ARTIKLI;

main()
{
    ARTIKLI artikal[MAX];
    int i=0;
    FILE *dato1, *dato2;
    if((dato1=fopen("proizvodiPDV.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    if((dato2=fopen("proizvodiPDVNovo.txt", "w"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(1)
    {
        fscanf(dato1, "%d %s %f %f", &artikal[i].barKod, &artikal[i].ime,
            &artikal[i].cena, &artikal[i].PDV);
        if(feof(dato1)) break;
        artikal[i].cena*=(1+artikal[i].PDV);
        fprintf(dato2, "%5d %-15s %4.2f\n",
            artikal[i].barKod, artikal[i].ime, artikal[i].cena);
        i++;
    }
    fclose(dato1); fclose(dato2);
    printf("\n Datoteka je formirana!\n");
    getche(); return 0;
}

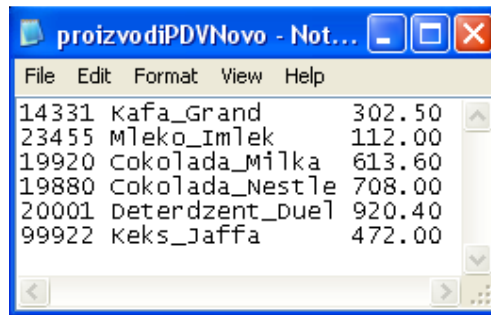
```



Испис на екрану



| File | Edit | Format | View | Help |
|-------|-----------------|--------|------|------|
| 14331 | Kafa_Grand | 250 | 0.21 | |
| 23455 | Mleko_Imlek | 100 | 0.12 | |
| 19920 | Cokolada_Milka | 520 | 0.18 | |
| 19880 | Cokolada_Nestle | 600 | 0.18 | |
| 20001 | Deterdzent_Duel | 780 | 0.18 | |
| 99922 | Keks_Jaffa | 400 | 0.18 | |

Изглед датотеке **proizvodiPDV.txt**


| File | Edit | Format | View | Help |
|-------|-----------------|--------|------|------|
| 14331 | Kafa_Grand | 302.50 | | |
| 23455 | Mleko_Imlek | 112.00 | | |
| 19920 | Cokolada_Milka | 613.60 | | |
| 19880 | Cokolada_Nestle | 708.00 | | |
| 20001 | Deterdzent_Duel | 920.40 | | |
| 99922 | Keks_Jaffa | 472.00 | | |

Изглед датотеке **proizvodiPDVNovo.txt** након извршавања програма

5.34. Саставити програм који из датотеке **igraci.txt** чита податке о играчима (висина, тежина, број кошева, број асистенција, број украдених лопти, број блокада), проналази играча са највећим бројем остварених поена и на стандардном излазу исписује име тог играча и број остварених поена. Укупан број поена рачунати као:

број кошева * 1 + број асистенција * 0.5 + број украдених лопти * 0.3 + број блокада * 0.22

Задатак решити употребом функције за читање садржаја датотеке, функције за испис података и функције за рачунање укупног броја поена.

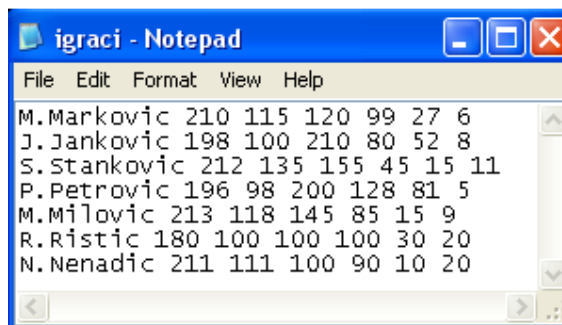
```
#include <stdio.h>
#define MAX 100

typedef struct igraci
{
    char ime[30];
    int visina;
    int tezina;
    int brKoseva;
    int brAsistencija;
    int brUkradenihLopti;
    int brBlokada;
    float ukupanBrBodova;
} IGRACI;

IGRACI igrac[MAX];
IGRACI najbolji;
int brIgraca=0;
int brBodova=0;

int ucitajIgraca(FILE *dato, IGRACI *i)
{
    fscanf(dato, "%s%d%d%d%d%d", i->ime, &(i->visina), &(i->tezina),
           &(i->brKoseva), &(i->brAsistencija), &(i->brUkradenihLopti),
           &(i->brBlokada));
    if(feof(dato)) return 0;
    return 1;
}

float ukupanBrBodova(IGRACI i)
{
    float poeniKoseva=i.brKoseva*1;
    float poeniAsistencije=i.brAsistencija*0.5;
    float poeniUkradenihLopti=i.brUkradenihLopti*0.3;
    float poeniBlokada=i.brBlokada * 0.22;
    i.ukupanBrBodova=
```



| File | Edit | Format | View | Help |
|-------------|------|--------|------|-----------|
| M.Markovic | 210 | 115 | 120 | 99 27 6 |
| J.Jankovic | 198 | 100 | 210 | 80 52 8 |
| S.Stankovic | 212 | 135 | 155 | 45 15 11 |
| P.Petrovic | 196 | 98 | 200 | 128 81 5 |
| M.Milovic | 213 | 118 | 145 | 85 15 9 |
| R.Ristic | 180 | 100 | 100 | 100 30 20 |
| N.Nenadic | 211 | 111 | 100 | 90 10 20 |

Изглед датотеке **igraci.txt**

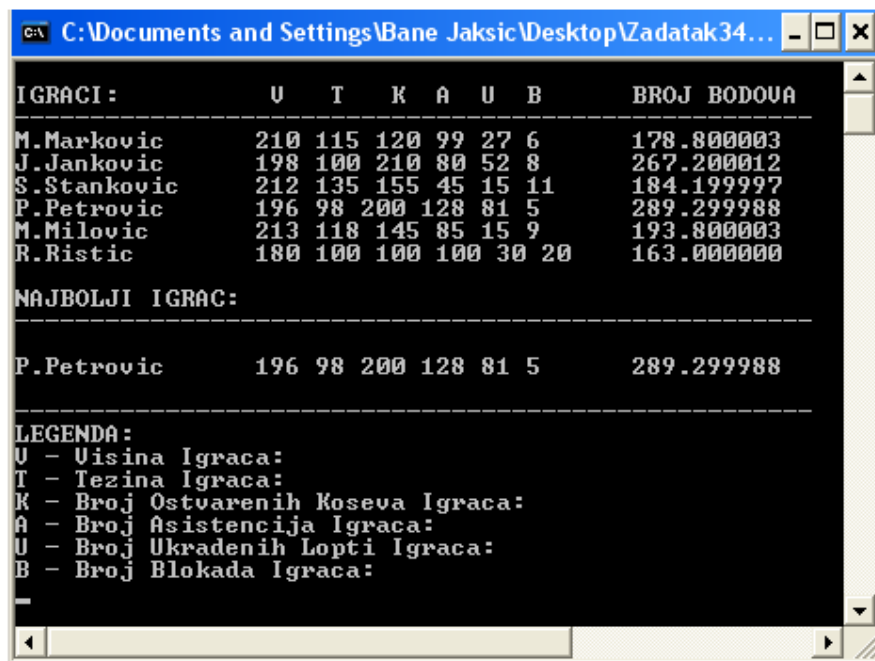
```

        poeniKoseva+poeniAsistencije+poeniUkradenihLopti+poeniBlokada;
    if(i.ukupanBrBodova > brBodova)
    {
        brBodova=i.ukupanBrBodova;
        najbolji=i;
    }
    return i.ukupanBrBodova;
}

void ispisiIgrace()
{
    int r;
    printf("\nIGRACI:          V   T   K   A   U   B          BROJ BODOVA\n");
    printf("-----\n");
    for(r=0; r<brIgraca; r++)
    printf("%-10s \t%d %d %d %d %d \t %f \n",
        igrac[r].ime,igrac[r].visina,igrac[r].tezina,igrac[r].brKoseva,
        igrac[r].brAsistencija,igrac[r].brUkradenihLopti,
        igrac[r].brBlokada, ukupanBrBodova(igrac[r]));
    printf("\nNAJBOLJI IGRAC:\n-----
    -----");
    printf("\n%-10s \t%d %d %d %d %d \t %f \n\n",
        najbolji.ime, najbolji.visina, najbolji.tezina,
        najbolji.brKoseva,najbolji.brAsistencija,
        najbolji.brUkradenihLopti, najbolji.brBlokada,
        najbolji.ukupanBrBodova);
    printf("-----\n");
    printf("LEGENDA:\n");
    printf("V - Visina Igraca:\n");
    printf("T - Tezina Igraca:\n");
    printf("K - Broj Ostvarenih Koseva Igraca:\n");
    printf("A - Broj Asistencija Igraca:\n");
    printf("U - Broj Ukradenih Lopti Igraca:\n");
    printf("B - Broj Blokada Igraca:\n");
}

main()
{
    FILE *dato;
    if((dato=fopen("igraci.txt", "r"))==NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    while(ucitajIgraca(dato, &igrac[brIgraca]))
    brIgraca++;
    ispisiIgrace();
    fclose(dato);
    getche();
    return 0;
}

```

| IGRACI: | U | T | K | A | U | B | BROJ BODOVA |
|-------------|-----|-----|-----|-----|----|----|-------------|
| M.Markovic | 210 | 115 | 120 | 99 | 27 | 6 | 178.800003 |
| J.Jankovic | 198 | 100 | 210 | 80 | 52 | 8 | 267.200012 |
| S.Stankovic | 212 | 135 | 155 | 45 | 15 | 11 | 184.199997 |
| P.Petrovic | 196 | 98 | 200 | 128 | 81 | 5 | 289.299988 |
| M.Milovic | 213 | 118 | 145 | 85 | 15 | 9 | 193.800003 |
| R.Ristic | 180 | 100 | 100 | 100 | 30 | 20 | 163.000000 |

NAJBOLJI IGRAC:

| | | | | | | | |
|------------|-----|----|-----|-----|----|---|------------|
| P.Petrovic | 196 | 98 | 200 | 128 | 81 | 5 | 289.299988 |
|------------|-----|----|-----|-----|----|---|------------|

LEGENDA:
 U - Visina Igraca:
 T - Tezina Igraca:
 K - Broj Ostvarenih Koseva Igraca:
 A - Broj Asistencija Igraca:
 U - Broj Ukradenih Lopti Igraca:
 B - Broj Blokada Igraca:

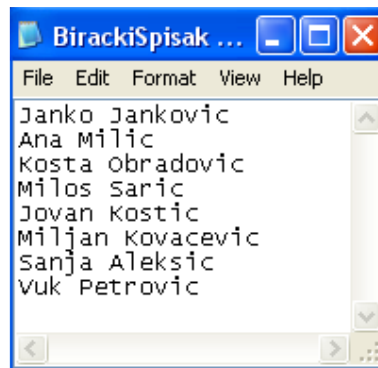
Испис на екрану

5.35. Датотека **BirackiSpisak.txt** садржи имена и презимена грађана. Саставити програм који списак из датотеке штампа на екрану прво сортиран по имену особа, а затим по презимену. На крају дати обавештење колико особа има исти редни број на оба сортирана списка.

```
#include <stdio.h>
#include <string.h>
#define MAX 1000

typedef struct gradjanin
{
    char ime[15];
    char prezime[15];
} GRADJANIN;

void SortirajIme(GRADJANIN *a, int n)
{
    int i, j;
    int min;
    GRADJANIN pom;
    for(i=0; i<n-1; i++)
    {
        /*Unutrasnja petlja pronalazi poziciju min, na kojoj
        se nalazi najmanji od elemenata a[i].ime,...,a[n-1].ime.*/
        min=i;
        for(j=i+1; j<n; j++)
            if(strcmp(a[j].ime, a[min].ime) < 0)
                min = j;
        /*Zamena elemenata na pozicijama (i) i min. Ovo se radi
        samo ako su (i) i min razliciti, inace je nepotrebno.*/
        if(min != i)
        {
            pom=a[i];
            a[i]=a[min];
            a[min]=pom;
        }
    }
}
```

Изглед датотеке **BirackiSpisak.txt**

```

        a[i]=a[min];
        a[min]=pom;
    }
}

void SortirajPrezime(GRADJANIN a[], int n)
{
    int i, j, min;
    GRADJANIN pom;
    for(i=0; i<n-1; i++)
    {
        /*Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
        najmanji od elemenata a[i].prezime,...,a[n-1].prezime.*/
        min=i;
        for(j=i+1; j<n; j++)
            if(strcmp(a[j].prezime, a[min].prezime) < 0)
                min=j;
        /*Zamena elemenata na pozicijama (i) i min. Ovo se radi
        samo ako su (i) i min razliciti, inace je nepotrebno.*/
        if(min != i)
        {
            pom=a[i];
            a[i]=a[min];
            a[min]=pom;
        }
    }
}

int Pretraga(GRADJANIN a[], int n, GRADJANIN *x )
{
    int i;
    for(i=0; i<n; i++)
        if(strcmp(a[i].ime, x->ime)==0 &&
            strcmp(a[i].prezime, x->prezime)==0)
            return i;
    return -1;
}

main()
{
    GRADJANIN spisak1[MAX], spisak2[MAX];
    int i, n, istiRbr=0;
    FILE *dato;
    if((dato= fopen("BirackiSpisak.txt", "r"))== NULL)
    {
        printf("\n Greska pri otvaranju datoteke!");
        exit(1);
    }
    for(i=0; fscanf(dato,"%s %s", spisak1[i].ime, spisak1[i].prezime) !=
        EOF; i++ )
        spisak2[i]=spisak1[i];
    n=i++;
    fclose(dato);
    SortirajIme(spisak1, n);
    printf("\n\n -Biracki spisak [uredjen prema imenima]-\n");
    for(i=0; i<n; i++)
        printf(" %d. %s %s\n",i+1,spisak1[i].ime, spisak1[i].prezime);
    SortirajPrezime(spisak2, n);
    printf("\n\n -Biracki spisak [uredjen prema prezimenima]-\n");
    for(i=0; i<n; i++)
        printf(" %d. %s %s\n",i+1,spisak2[i].ime, spisak2[i].prezime);
}

```

```

    for(i=0; i<n ;i++)
        if(i==Pretraga(spisak2,n, &spisak1[i]))
            istiRbr++;
    printf("\n\n Broj gradjana koji imaju isti redni broj na oba spiska:
           %d\n",istiRbr);
    getch();
    return 0;
}

```

```

C:\Documents and Settings\Bane Jaksic\Desktop\Zadatak35.exe
-Biracki spisak [uredjen prema imenima]-
1. Ana Milic
2. Janko Jankovic
3. Jovan Kostic
4. Kosta Obradovic
5. Miljan Kovacevic
6. Milos Saric
7. Sanja Aleksic
8. Uuk Petrovic

-Biracki spisak [uredjen prema prezimenima]-
1. Sanja Aleksic
2. Janko Jankovic
3. Jovan Kostic
4. Miljan Kovacevic
5. Ana Milic
6. Kosta Obradovic
7. Uuk Petrovic
8. Milos Saric

Broj gradjana koji imaju isti redni broj na oba spiska: 2

```

Испис на екрану

5.36. Саставити програм којим се формира датотека **ucenik.txt** која садрже податке о **n** ученика (име, адреса, разред, одељење). Подаци о једном ученику се налазе у једном реду. Након формирања датотеке из ње прочитати и на екрану исписати податке о ученицима који су разред **x**. Број **x** се уноси са тастатуре.

```

#include <stdio.h>
#define MAX 100

typedef struct ucenci
{
    char ime[15];
    char prezime[15];
    char adresa[15];
    int razred;
    int odeljenje;
} UCENICI;

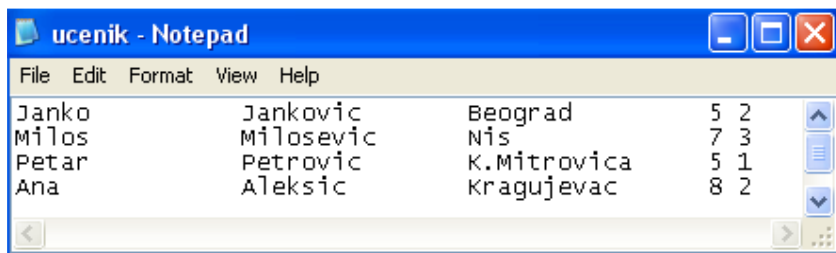
main()
{
    UCENICI ucenik[MAX];
    int i, n, x;
    FILE *dato;

```

```

if((dato=fopen("ucenik.txt","w"))==NULL)
{
    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
printf("\n Broj ucenika: ");
scanf("%d",&n);
for(i=0; i<n; i++)
{
    printf("\n -UCENIK %d-\n",i+1);
    printf(" Ime: ");
    scanf("%s", &ucenik[i].ime);
    printf(" Prezime: ");
    while(getchar()!='\n');
    scanf("%s", &ucenik[i].prezime);
    printf(" Adresa: ");
    scanf("%s", &ucenik[i].adresa);
    printf(" Razred: ");
    scanf("%d",&ucenik[i].razred);
    printf(" Odeljenje: ");
    scanf("%d",&ucenik[i].odeljenje);
    fprintf(dato,"%-15s%-15s%-15s%2d%2d\n",ucenik[i].ime,
        ucenik[i].prezime, ucenik[i].adresa,
        ucenik[i].razred, ucenik[i].odeljenje);
}
fclose(dato);
printf("\n\n Datoteka je kreirana.\n");
printf("\n Rezred za pretragu: ");
scanf("%d",&x);
if((dato=fopen("ucenik.txt","r"))==NULL)
{
    printf("\n Greska pri otvaranju datoteke!");
    exit(1);
}
printf("\n -Ucenici %d-og razreda-\n", x);
for(i=0; i<n; i++)
{
    fscanf(dato, "%s%s%d%d",&ucenik[i].ime, &ucenik[i].prezime,
        &ucenik[i].adresa,&ucenik[i].razred,&ucenik[i].odeljenje);
    if(ucenik[i].razred==x)
        printf(" %-10s%-10s%-15s%2d%2d\n",ucenik[i].ime, ucenik[i].prezime,
            ucenik[i].adresa, ucenik[i].razred, ucenik[i].odeljenje);
}
fclose(dato);
getche();
return 0;
}

```

Изглед датотеке *ucenik.txt* након извршавања програма

```
C:\Documents and Settings\Bane Jaksic\Desktop...  
Broj ucenika: 4  
  
-UCENIK 1-  
Ime: Janko  
Prezime: Jankovic  
Adresa: Beograd  
Razred: 5  
Odeljenje: 2  
  
-UCENIK 2-  
Ime: Milos  
Prezime: Milosevic  
Adresa: Nis  
Razred: 7  
Odeljenje: 3  
  
-UCENIK 3-  
Ime: Petar  
Prezime: Petrovic  
Adresa: K.Mitrovica  
Razred: 5  
Odeljenje: 1  
  
-UCENIK 4-  
Ime: Ana  
Prezime: Aleksic  
Adresa: Kragujevac  
Razred: 8  
Odeljenje: 2  
  
Datoteka je kreirana.  
Rezred za pretragu: 5  
  
-Ucenici 5-og razreda-  
Janko      Jankovic   Beograd    5 2  
Petar      Petrovic   K.Mitrovica 5 1
```

Испис на екрану

5.37. Датотека **ispiti.txt** садржи податке о 5 студената и испитима у облику низа структура. Структура садржи поља: **ime**, **prezime**, **predavanja**, **kolokvijumi** и **ispit**, која редом означавају име студента, презиме студента, број поена са предавања, број поена са колоквијума и број поена са испита.

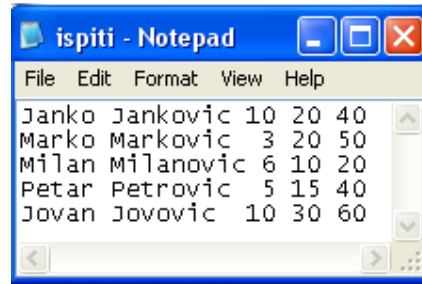
Саставити програм који чита податке о студентима и испитује да ли је студент положио испит (испит је положен ако је збир поена са предавања, колоквијума и испита већи од 50). Уколико је студент положио испит, његове податке (име, презиме, укупан број поена) сместити у датотеку **polozili.txt** или уколико није положио испит исте податке сместити у датотеку **nisu-polozili.txt**. Структура која се смешта у ове две датотеке садржи поља **ime**, **prezime** и **ukupnoPoena**.

На екрану исписати број студената који су положили и број студената који нису положили испит.

```
#include <stdio.h>

typedef struct student
{
    char ime[15];
    char prezime[15];
    int predavanja;
    int kolokvijumi;
    int ispit;
    int ukupanBrBodova;
}STUDENT;
```

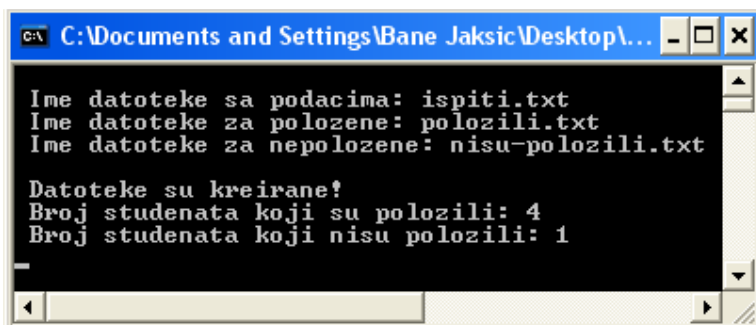
```
main()
{
    STUDENT studenti[5];
    int i, brp=0, brn=0;
    FILE *f, *p, *n;
    char ime1[20], ime2[20], ime3[20];
    printf("\n Ime datoteke sa podacima: ");
    scanf("%s",&ime1);
    printf(" Ime datoteke za polozone: ");
    scanf("%s",&ime2);
    printf(" Ime datoteke za nepolozone: ");
    scanf("%s",&ime3);
    f=fopen(ime1, "r");
    if(f == NULL)
    {
        printf("Greska prilikom otvaranja datoteke!\n");
        return 1;
    }
    p=fopen(ime2, "w");
    if(p == NULL)
    {
        printf("Greska prilikom otvaranja datoteke!\n");
        return 1;
    }
    n=fopen(ime3, "w");
    if(n == NULL)
    {
        printf("Greska prilikom otvaranja datoteke!\n");
        return 1;
    }
    for(i=0;i<5;i++)
    {
        fscanf(f, "%s%d%d%d", &studenti[i].ime, &studenti[i].prezime,
            &studenti[i].predavanja, &studenti[i].kolokvijumi,
            &studenti[i].ispit);
        studenti[i].ukupanBrBodova = (studenti[i].predavanja +
            studenti[i].kolokvijumi + studenti[i].ispit);
        if(studenti[i].ukupanBrBodova>50)
        {
            fprintf(p, "%s %s %d\n", studenti[i].ime, studenti[i].prezime,
                studenti[i].ukupanBrBodova);
            brp++;
        }
        else
        {
            fprintf(n, "%s %s %d\n", studenti[i].ime, studenti[i].prezime,
                studenti[i].ukupanBrBodova);
            brn++;
        }
    }
}
```

Изглед датотеке *ispiti.txt*

```

    }
}
fclose(f);
fclose(p);
fclose(n);
printf ("\n Datoteke su kreirane!\n");
printf (" Broj studenata koji su polozili: %d\n", brp);
printf (" Broj studenata koji nisu polozili: %d\n", brn);
getche();
return 0;
}

```



```

C:\Documents and Settings\Bane Jaksic\Desktop\...
Ime datoteke sa podacima: ispiti.txt
Ime datoteke za polozene: polozili.txt
Ime datoteke za nepolozene: nisu-polozili.txt

Datoteke su kreirane!
Broj studenata koji su polozili: 4
Broj studenata koji nisu polozili: 1

```

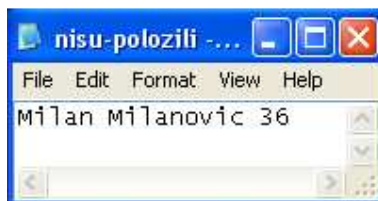
Испис на екрану



```

File Edit Format View Help
Janko Jankovic 70
Marko Markovic 73
Petar Petrovic 60
Jovan Jovovic 100

```

Изглед датотеке **polozili.txt** након извршавања програма


```

File Edit Format View Help
Milan Milanovic 36

```

Изглед датотека **nisu-polozili.txt** након извршавања програма

5.38. Датотека **Studenti.txt** садржи податке о студентима: име и презиме, број индекса, шифра предмета и оцена из предмета. Имена студената се могу јавити више пута за различиту шифру предмета. Саставити програм који креира датотеку **Proseci.txt** која се састоји од шифре предмета и просечне оцене из тог предмета. Имена датотека се уносе на главном улазу.

```

#include <stdio.h>
#define MAX 100

typedef struct studenti
{
    char ime[30];
    int index;
    int sifra;
    int ocena;
}STUDENTI;

main()
{
    FILE *ulaz, *izlaz;
    char ime1[20], ime2[20];
    STUDENTI st[100];

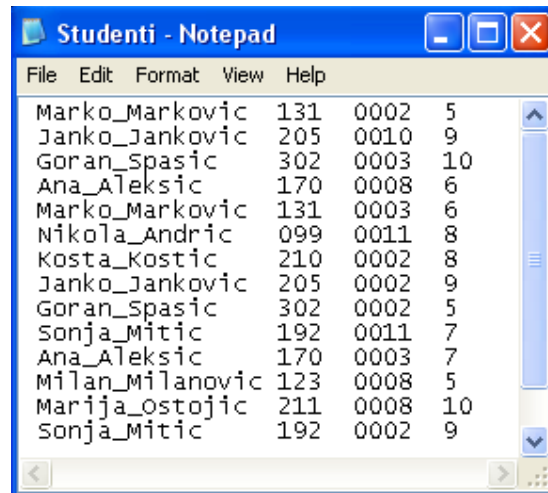
```

```

int i, j, t, k=0, zbirOcena, brStd, brIsp, nadjeniIsti;
int razSifre[100];
float prosek[100];
printf("\n Ime datoteke sa podacima: ");
scanf("%s",&ime1);
printf(" Ime datoteke koja se kreira: ");
scanf("%s",&ime2);
if((ulaz=fopen(ime1, "r"))==NULL)
{
    printf("Greska prilikom otvaranja datoteke!\n");
    return 1;
}
if((izlaz=fopen(ime2, "w"))==NULL)
{
    printf("Greska prilikom otvaranja datoteke!\n");
    return 1;
}
i=0;
while(1)
{
    fscanf(ulaz, "%s%d%d%d\n", &st[i].ime, &st[i].index,
                                     &st[i].sifra, &st[i].ocena);

    if(feof(ulaz)) break;
    i++;
}
brStd=i;
fclose(ulaz);
for(i=0; i<brStd-1; i++)
{
    nadjeniIsti=0;
    for(j=i+1; j<brStd; j++)
        if(st[i].sifra == st[j].sifra)
        {
            nadjeniIsti=1;
            break;
        }
    if(!nadjeniIsti)
    {
        razSifre[k]=st[i].sifra;
        k++;
    }
}
razSifre[k]=st[brStd-1].sifra;
t=0;
while(t<=k)
{
    zbirOcena=0;
    brIsp=0;
    for(i=0; i<=brStd; i++)
    {
        if(st[i].sifra==razSifre[t])
        {
            zbirOcena+=st[i].ocena;
            brIsp++;
        }
    }
    prosek[t]=(float)zbirOcena/brIsp;
    t++;
}
for(i=0; i<=k; i++)

```

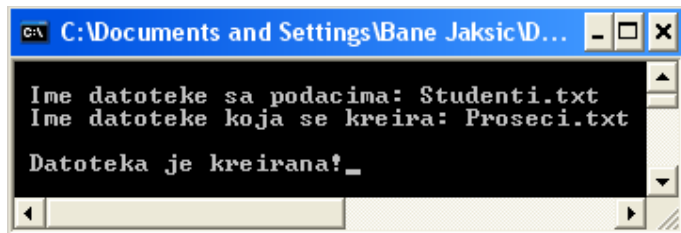


Изглед датотеке Studenti.txt

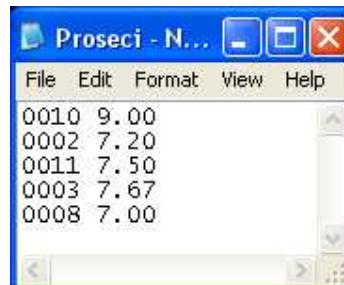

```

    fprintf(izlaz, "%04d %4.2f\n", razSifre[i], prosek[i]);
    fclose(izlaz);
    printf("\n Datoteka je kreirana!");
    getch();
    return 0;
}

```



Испис на екрану

Изглед датотеке **Proseci.txt** након извршавања програма

5.39. У Косовској Митровици је у периоду од 01. до 10. августа 2011. мерена температура ваздуха у терминима 07, 13 и 19 часова. Подаци о температурама заједно са даном мерења су смештени у датотеци **Temperature.txt** (као што је приказано на слици). Саставити програм који формира датотеку **Prosečne.txt** у којој се смештају просечне дневне температуре (збир свих дневних мерења подељен са 3) сортиране од највише ка најнижој. Поред просечне дневне температуре поставити и датум када је измерена та температура.

На екрану одштампати датум и просечну температуру најтоплијег и најхладнијег дана, као и просечне тепературе у 07, 13 и 19 часова (збир свих температура у 07 (односно 13 и 19) подељено са 10).

Имена датотека унети на главном улазу.

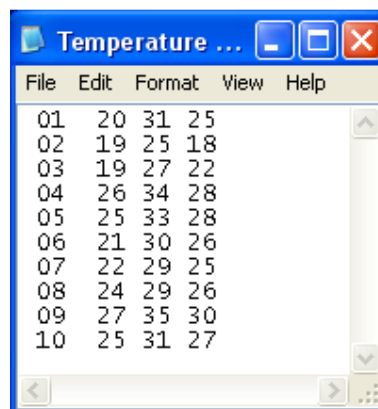
```

#include <stdio.h>

typedef struct temperature
{
    int datum;
    int m7;
    int m13;
    int m19;
} TEMPERATURE;

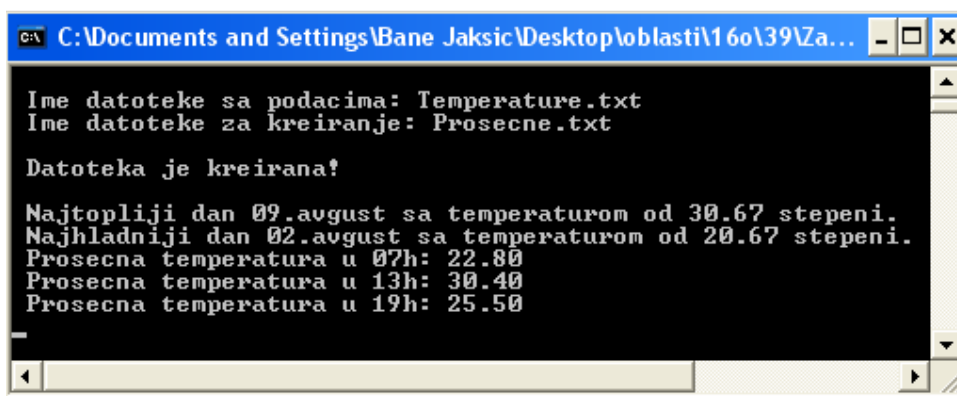
main()
{
    TEMPERATURE temp[10];
    int i, j, uk7=0, uk13=0, uk19=0, pomd;
    float pr7, pr13, pr19, prdnevna[10], pom;
    FILE *ulaz, *izlaz;
    char ime1[20], ime2[20];
    printf("\n Ime datoteke sa podacima: ");
    scanf("%s",&ime1);
    printf(" Ime datoteke za kreiranje: ");
    scanf("%s",&ime2);
    if((ulaz=fopen(ime1, "r"))== NULL)
    {
        printf("Greska prilikom otvaranja datoteke!\n");
        return 1;
    }
}

```

Изглед датотеке **Temperature.txt**

```
if((izlaz=fopen(ime2, "w"))== NULL)
{
    printf("Greska prilikom otvaranja datoteke!\n");
    return 1;
}
/*Citanje podataka iz datoteke*/
/*Racunanje ukupnih temperatura u 7h, 13h i 19h i dnevne prosečne temperature*/
for(i=0; i<10; i++)
{
    fscanf(ulaz, "%d%d%d", &temp[i].datum, &temp[i].m7,
                                                &temp[i].m13, &temp[i].m19);

    uk7+=temp[i].m7;
    uk13+=temp[i].m13;
    uk19+=temp[i].m19;
    prdnevna[i]=(float)(temp[i].m7 + temp[i].m13 + temp[i].m19)/3;
}
/*Racunjane prosečnih temperatura u 7h, 13h i 19h*/
pr7=(float)uk7/10;
pr13=(float)uk13/10;
pr19=(float)uk19/10;
/*Sortiranje prosečnih temperatura od najviše ka najniže*/
for(i=0; i<9; i++)
    for(j=i+1; j<10; j++)
        if(prdnevna[i] < prdnevna[j])
        {
            pom=prdnevna[i];
            pomd=temp[i].datum;
            prdnevna[i]=prdnevna[j];
            temp[i].datum=temp[j].datum;
            prdnevna[j]=pom;
            temp[j].datum=pomd;
        }
/*Upis podataka u novokreiranu datoteku*/
for(i=0; i<10; i++)
    fprintf(izlaz, " %02d    %.2f\n", temp[i].datum, prdnevna[i]);
fclose(ulaz);
fclose(izlaz);
printf("\n Datoteka je kreirana!\n\n");
/*Ispis podataka na ekran*/
printf(" Najtopliji dan %02d.avgust sa temperaturom od %.2f stepeni.\n",
        temp[0].datum, prdnevna[0]);
printf(" Najhladniji dan %02d.avgust sa temperaturom od %.2f stepeni.\n",
        temp[9].datum, prdnevna[9]);
printf(" Prosečna temperatura u 07h: %.2f\n", pr7);
printf(" Prosečna temperatura u 13h: %.2f\n", pr13);
printf(" Prosečna temperatura u 19h: %.2f\n", pr19);
getche();
return 0;
}
```

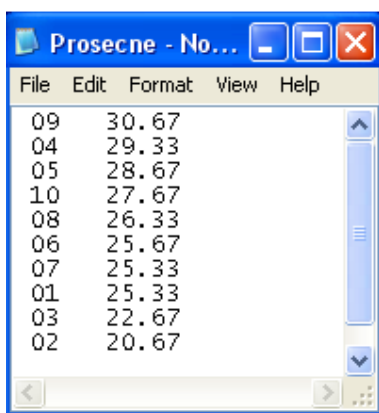


```
C:\Documents and Settings\Bane Jaksic\Desktop\oblasti\160\39\Za...
Ime datoteke sa podacima: Temperature.txt
Ime datoteke za kreiranje: Prosecne.txt

Datoteka je kreirana!

Najtopliji dan 09.avgust sa temperaturom od 30.67 stepeni.
Najhladniji dan 02.avgust sa temperaturom od 20.67 stepeni.
Prosecna temperatura u 07h: 22.80
Prosecna temperatura u 13h: 30.40
Prosecna temperatura u 19h: 25.50
```

Испис на екрану



*Изглед датотеке **Prosecne.txt** након извршавања програма*

ЛИТЕРАТУРА

- [1] Brian W. Kernighan, Dennis M. Ritchie: **The C Programming Language**, New Jersey, 1988.
- [2] Laslo Kraus: **Programski jezik C sa rešenim zadacima**, Akademska misao, Beograd, 2006.
- [3] Laslo Kraus: **Rešeni zadaci iz programskog jezika C**, Akademska misao, Beograd, 2005.
- [4] Ivo Mateljan: **Programiranje C jezikom**, Split, 2005/2006.
- [5] Jozo J. Dujmović: **Programski jezici i metode programiranja**, Naučna knjiga, Beograd, 1990.
- [6] B. S. Gottfried: **Theory and Problems of Programming with C**, Schaum's outline series, McGraw-Hill, 1996.
- [7] Clovis Tondo, Scott Gimpel: **Programski jezik C – rešenja zadataka**, CET, Beograd, 2004.
- [8] A.Hansen: **Programiranje na jeziku C – potpuni vodič za programski jezik C**, Mikroknjiga, Beograd, 2000.
- [9] Igor Đurović, Slobodan Đukanović, Vesna Popović: **Programski jezik C sa zbirkom riješenih zadataka**, ETF Podgorica, Podgorica, 2006.
- [10] Milan Čabarkapa: **C osnovi programiranja**, Krug, Beograd, 1996.
- [11] Milan Čabarapa, Stanka Matković: **C/C++ zbirka zadataka**, Krug, Beograd, 2003.
- [12] Milan Čabarkapa, Nevenka Spalević: **Metodička zbirka zadataka iz programiranja**, Sova, Novi Beograd, 1997.
- [13] Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селъун М.И.: **Задачи по программированию**, Наука, Москва, 1988.
- [14] Абрамов В.Г., Тирфионов Н.П., Трифонава Г.Н., **Введение в язык Паскаль**, Наука, Москва, 1988.