Програмирање други разред

Упознавање ученика са наставним планом и програмом

У другом разреду настављамо са упознавањем програмског језика C и његових могућности.

Наставни план и програм се одвија у учионици (2 часа недељно) и кабинету са рачунарима (2 часа недељно).

Области (модули) које ћемо изучавати ове године су:

- Једнодимензионални низ или вектор (8+8)
- Показивачи (4+4)
- Функције (14+14)
- Вишедимензионални низови (14+14)
- Стрингови и текстуалне датотеке (20+20)
- Структуре и бинарне датотеке (10+10)

Литература:

- Ласло Краус, Програмирање за III разред електротехничке школе, ЗУНС, Београд, 2004, КБ 23301
- ANSI C стандард у електронском облику
- Књиге и збирке из програмирања у електронском облику
- Упутства и материјали за предавања и вежбе у електронском облику

Уместо предговора

Овај приручник има задатак да вас у кратким цртама подсети на све оно што смо радили и о чему смо причали на часовима, како у учионици, тако и у кабинету са рачунарима.

Сваки појам је представљен на једноставан, разумљив начин.

Примери програма (програмски код) су кратки, једноставни, илустративни и имају задатак да вам објасне основне принципе.

Дати примери представљају један начин решавања датог проблема. Готово сваки пример се може реализовати кроз другачији програмски код.

Имате слободу да наведене примере мењате, усавршавате, у складу са вашим жељама, идејама, ...

Неки од савета у вези са програмирањем:

Не идите пребрзо. Научите нешто исправно пре него што кренете даље.

Учење подразуме читање из књига, уџбеника, садржаја на Интернету, ...

Под учењем програмирања подразумева се и читање и разумевање написаног програмског кода!

Када прочитате са разумевањем примере програмских кодова, покрените их!

Напишите властити код што пре!

Научите да користите поруке из развојног система за програмирање, да користите програм за исправљање грешака унутар развојног система, ...

Морате много да радите на себи и својим вештинама!

Задајте себи пројекте, прво мале, а затим све веће и веће.

За неколико месеци ухватићете себе како радите ствари за које сте раније мислили да их никада нећете достићи.

Водите рачуна да у животу постоје и друге ствари осим програмирања!

Направите равнотежу између школе, учења, одмора, слободног времена, породице, пријатеља, ...

Будите веома ефикасни у процесу учења!

Стекните знање за краће време и добијте више слободног времена!

Једнодимензионални низ или вектор

Да се подсетимо:

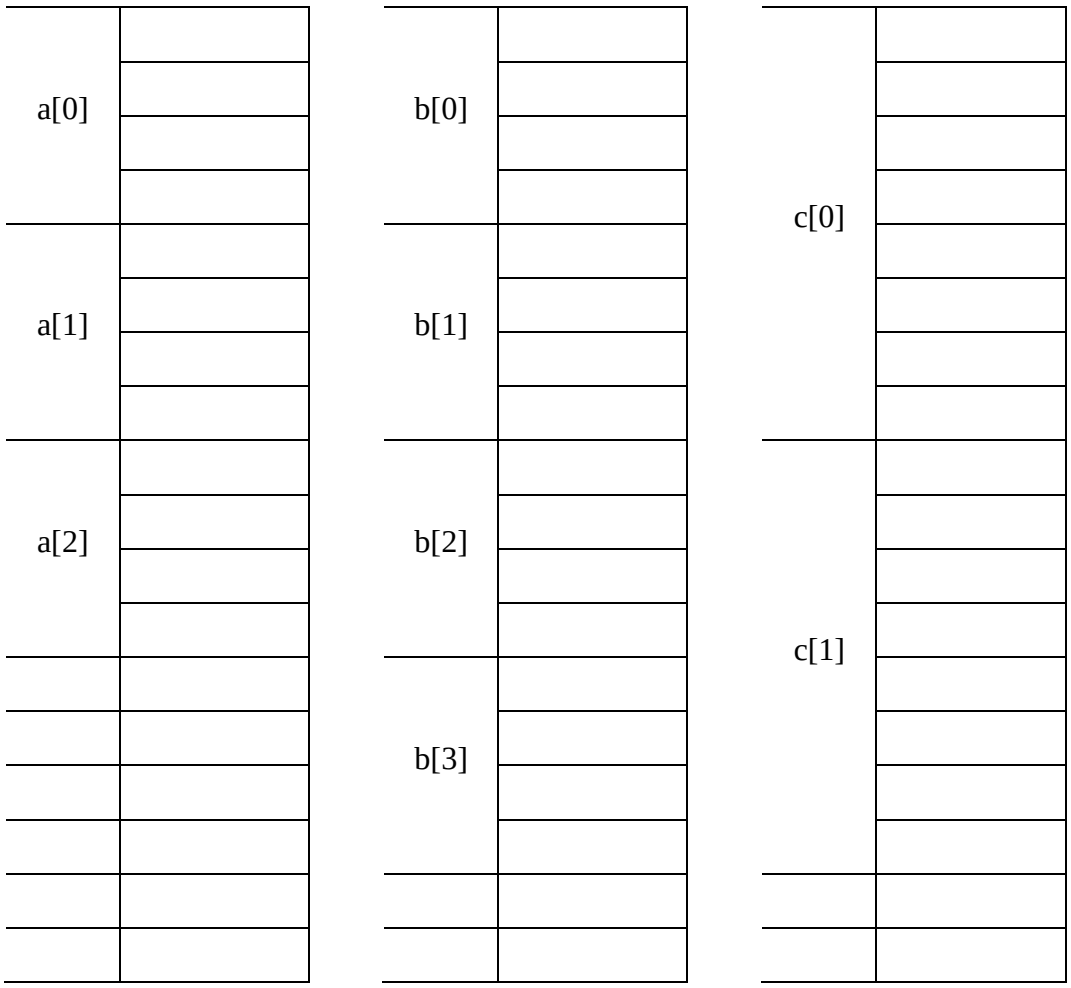
Низ је скуп узастопних меморијских локација које чувају податке истога типа.

У првом разреду смо се упознали са низовима целих бројева и низовима реалних бројева (једноструке и двоструке тачности).

Низ је декларисан именом и бројем елемената у низу.

```
int a[3];           /* Deklaracija celobrojnog niza cije je ime a i ima 3 elementa */
float b[4];         /* Deklaracija realnog niza cije je ime b i ima 4 elementa     */
double c[2];        /* Deklaracija realnog niza cije je ime c i ima 2 elementa     */
```

Приказ заузетих меморијских локација након декларисања низова, једно меморијско поље одговара једном бајту:



Претпоставили смо да је величина целобројне променљиве типа **int** 32 бита, односно 4 бајта. Величина променљивих типа **float** и **double** дефинисана је стандардом.

Ако желимо да приступимо неком елементу низа неопходно је да знамо два податка. То су **име** низа и **индекс** низа.

Индекс низа нам говори о позицији елемента унутар низа. Индекси низа почињу од **0** и иду до **n-1**, где је **n** укупан број елемената низа.

У наведеном примеру први елемент целобројног низа је означен са **a[0]**, други елемент са **a[1]**, а трећи елемент **са a[2]**.

Први елемент низа реалних бројева једноструке тачности означен је са **b[0]**, други елемент са **b[1]**, трећи елемент са **b[2]**, а четврти елемент са **b[3]**.

Први елемент низа реалних бројева двоструке тачности означен је са **c[0]**, а други елемент са **c[1]**.

Иницијализација елемената низа

Приликом декларације можемо да извршимо иницијализацију вредности елемената низа.

```
int a[5] = {11, 22, 55, 77, 66};
```

Ако изоставимо величину низа приликом иницијализације, тада ће преводилац сам да одреди број елемената у низу.

```
int b[] = {11, 22, 55, 77, 66};     /* isto kao da smo napisali int b[5] = ... */
```

Ако се обави делимична иницијализација, тада преводилац елементима којима нису наведене вредности при делимичној иницијализацији додељује вредност 0.

```
int c[12] = {11, 22, 55, 77, 66};

/* elementi niza imaju vrednosti 11, 22, 55, 77, 66, 0, 0, 0, 0, 0, 0, 0 */
```

Уколико се изостави иницијализација, тада „не знамо“ које су вредности елемената низа, већ их морамо иницијализовати (учитати) унутар програма, нпр. тражимо од корисника да унесе вредности за елементе низа!

Формирање низа

Написати програм који учитава елементе целобројног низа, а затим приказује вредности елемената низа. Максимални број елемената у низу је 50.

```
#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elemenata u nizu */

int main(void)
{
    int n, i;       /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Prikaz elemenata niza */
    printf("\n\nPrikaz vrednosti elemenata niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Претраживање низа

Претраживање низа представља испитивање елемената низа да ли се међу њима налази одређена вредност.

Пример:

Написати програм који учитава елементе низа реалних бројева двоструке тачности, а затим претражује елементе учитаног низа и испитује да ли се међу њима налази број који је корисник накнадно унео.

Уколико се пронађе тражени број, тада исписати у ком се елементу низа налази. Програм треба да пронађе сва места појављивања броја у низу. Уколико се не пронађе тражени број, тада исписати одговарајућу поруку.

Максимални број елемената у низу је 25.

```
#include <stdio.h>

#define MAX 25          /* Definisanje najveceg broja elemenata u nizu */

int main (void)
{
    int i, n, p=0;      /* Promenljive u vezi sa indeksima niza. */
                        /* Ove promenljive su uvek celobrojne!!! */

    double a[MAX], b;   /* Promenljive u vezi sa nizom.          */
                        /* Tipovi ovih promenljivih zavise       */
                        /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\nUnesite broj elemenata niza: ");
        scanf("%d", &n);      /* uvek je %d konverzija */
    }while(n<1 || n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    for(i=0; i<n; i++){
        printf("\n a[%d]= ",i);     /* uvek je %d konverzija */
        scanf("%lf", &a[i]);        /* konverzija zavisi od tipa elementa niza */
    }

    /* koju vrednost zelimo da potrazimo u nizu */
    printf("\nKoji broj zelite da pronadjete u nizu ");
    scanf("%lf",&b);          /* konverzija zavisi od tipa elementa niza */

    for(i=0; i<n; i++)
        if(a[i]==b){
            printf("\nTrazeni broj se nalazi u a[%d]\n",i);
            p=1;        /* pomocna promenljiva za pretrazivanje          */
                        /* ako je p jednako 1 broj je pronadjen u nizu   */
                        /* ako je p jednako 0 broj nije pronadjen u nizu */
        }

    if(p==0)
        printf("\nTrazeni broj se ne nalazi u nizu\n");

    return 0;
}
```

У даљем тексту бавићемо се различитим варијантама претраживања. Анализираћемо само део програмског кода који се односи на претраживање.

Уколико нас интересује само да ли се неки број налази или не у низу, тада можемо да користимо следећи код:

```
int brojJePronadjen = 0;
int/float/double/... broj;    /* Tip zavisi od tipa elemenata niza */
...

for(i=0; i<n; i++)
    if(a[i] == broj)
        brojJePronadjen = 1;

if(brojJePronadjen == 1)
    printf("\nTrazeni broj se nalazi u nizu");
else
    printf("\nTrazeni broj se ne nalazi u nizu");
```

Претраживање можемо да убрзамо, тако што ћемо прекинути даље претраживање у случају када пронађемо број. Прекидање **for** петље обавићемо помоћу наредбе **break;**

```
int brojJePronadjen = 0;
int/float/double/... broj;    /* Tip zavisi od tipa elemenata niza */
...

for(i=0;i<n;i++)
    if(a[i] == broj){
        brojJePronadjen = 1;
        break;
    }

if(brojJePronadjen == 1)
    printf("\nTrazeni broj se nalazi u nizu\n");
else
    printf("\nTrazeni broj se ne nalazi u nizu\n");
```

Уколико нас интересује индекс првог појављивања броја у низу:

```
int brojJePronadjen = 0;
int/float/double/... broj;    /* Tip zavisi od tipa elemenata niza */
...

for(i=0;i<n;i++)
    if(a[i] == broj){
        brojJePronadjen = 1;
        break;
    }

if(brojJePronadjen == 1)
    printf("\nTrazeni broj se nalazi u a[%d]\n", i);
else
    printf("\nTrazeni broj se ne nalazi u nizu"\n);
```

Уколико нас интересује индекс последњег појављивања броја у низу:

```
int brojJePronadjen = 0;
int indeksPoslednjegPojavljivanja;
int/float/double/... broj;    /* Tip zavisi od tipa elemenata niza */
...

for(i=0;i<n;i++)
    if(a[i] == broj){
        brojJePronadjen = 1;
        indeksPoslednjegPojavljivanja = i;
    }

if(brojJePronadjen == 1)
    printf("\nTrazeni broj se nalazi u a[%d]\n", indeksPoslednjegPojavljivanja);
else
    printf("\nTrazeni broj se ne nalazi u nizu"\n);
```

Уколико нас интересује број појављивања траженог броја у низу:

```
int brojPojavljivanja = 0;
int/float/double/... broj;    /* Tip zavisi od tipa elemenata niza */
...

for(i=0;i<n;i++)
    if(a[i] == broj)
        brojPojavljivanja++;

if(brojPojavljivanja > 0)
    printf("\nBroj pojavljivanja je %d\n", brojPojavljivanja);
else
    printf("\nTrazeni broj se ne nalazi u nizu"\n);
```

Ротирање елемената низа удесно

Ротирање елемената низа подразумева померање свих елемената низа за по једно место удесно, при чему последњи елемент низа који „испада“ долази на „упражњено“ место.

Пример: Дат је низ од 5 елемената и једна помоћна променљива

a[0]	a[1]	a[2]	a[3]	a[4]		pomoc
11	22	33	44	55		

Поступак ротирања елемената низа за 1 место удесно почиње тако што се елемент са највећим индексом у низу (крајњи десни елемент) пребаци (ископира) у помоћну променљиву.

Напомена: Да би смо боље видели сам процес ротирања, вредности које су пребачене нећемо приказати на полазним пољима (иако оне ту постоје).

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
11	22	33	44		pomoc = a[4];	55

Помоћу наредби циклуса (for, while, ...) померамо један по један елемент, почевши од елемената са највећим индексом.

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
11	22	33		44	a[4] = a[3];	55

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
11	22		33	44	a[3] = a[2];	55

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
11		22	33	44	a[2] = a[1];	55

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
	11	22	33	44	a[1] = a[0];	55

Након наредби циклуса сада у елемент са индексом 0 (крајњи леви елемент) треба уписати вредност која је сачувана у помоћној променљивој.

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
55	11	22	33	44	a[0] = pomoc;	

Програмски код који демонстрира процес ротирања елемената низа удесно за једно место:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

pomoc = a[n-1];

for(i = n-1; i>0; i--)
    a[i] = a[i-1];

a[0] = pomoc;
```

Програмски код који демонстрира процес ротирања елемената низа удесно за **k** места:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

for(j = 0; j<k; j++){         /* k puta ponovi rotiranje za 1 mesto */
    pomoc = a[n-1];
    for(i = n-1; i>0; i--)
        a[i] = a[i-1];
    a[0] = pomoc;
}
```

Померање се разликује од **ротирања** по томе што се на „упражњено“ место не уписује податак који је „испао“ из низа, већ неки други број. Тај други број може бити 0 или број који уноси корисник или ... .

Програмски код који демонстрира процес померања елемената низа удесно за једно место:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

pomoc = a[n-1];               /* Moze da se sacuva, ali ne mora    */

for(i = n-1; i>0; i--)
    a[i] = a[i-1];

a[0] = 0;
```

Програмски код који демонстрира процес померања елемената низа удесно за **k** места:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza  */
...

for(j = 0; j<k; j++){         /* k puta ponovi pomeranje za 1 mesto */
    pomoc = a[n-1];           /* Moze da se sacuva, ali ne mora     */
    for(i = n-1; i>0; i--)
        a[i] = a[i-1];
    a[0] = 0;
}
```

Написати програм који елементе целобројног низа ротира удесно за 1 место.

Максимални број елемената у низу је 50.

```
#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i;       /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Rotiranje elemenata niza udesno za 1 mesto */
    p = a[n-1];
    for(i=n-1; i>0; i--)
        a[i] = a[i-1];

    a[0] = p;


    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Ротирање елемената низа улево

Ротирање елемената низа подразумева померање свих елемената низа за по једно место улево, при чему први елемент низа који „испада“ долази на „упражњено“ место.

Пример: Дат је низ од 5 елемената и једна помоћна променљива

a[0]	a[1]	a[2]	a[3]	a[4]		pomoc
11	22	33	44	55		

Поступак ротирања елемената низа за 1 место улево почиње тако што се елемент са најмањим индексом у низу (крајњи леви елемент) пребаци (ископира) у помоћну променљиву.

Напомена: Да би смо боље видели сам процес ротирања, вредности које су пребачене нећемо приказати на полазним пољима (иако оне ту постоје).

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
	22	33	44	55	pomoc = a[0];	11

Помоћу наредби циклуса (for, while, ...) померамо један по један елемент, почевши од елемената са најмањим индексом.

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
22		33	44	55	a[0] = a[1];	11

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
22	33		44	55	a[1] = a[2];	11

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
22	33	44		55	a[2] = a[3];	11

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
22	33	44	55		a[3] = a[4];	11

Након наредби циклуса сада у елемент са индексом n-1 (крајњи десни елемент) треба уписати вредност која је сачувана у помоћној променљивој.

a[0]	a[1]	a[2]	a[3]	a[4]	наредба	pomoc
22	33	44	55	11	a[4] = pomoc;	

Програмски код који демонстрира процес ротирања елемената низа улево за једно место:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

pomoc = a[0];

for(i = 0; i<n-1; i++)
    a[i] = a[i+1];

a[n-1] = pomoc;
```

Програмски код који демонстрира процес ротирања елемената низа улево за **k** места:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

for(j = 0; j<k; j++){         /* k puta ponovi rotiranje za 1 mesto */
    pomoc = a[0];
    for(i = 0; i<n-1; i++)
        a[i] = a[i+1];
    a[n-1] = pomoc;
}
```

Померање се разликује од **ротирања** по томе што се на „упражњено“ место не уписује податак који је „испао“ из низа, већ неки други број. Тај други број може бити 0 или број који уноси корисник или ... .

Програмски код који демонстрира процес померања елемената низа улево за једно место:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza */
...

pomoc = a[0];                 /* Moze da se sacuva, ali ne mora    */

for(i = 0; i<n-1; i++)
    a[i] = a[i+1];

a[n-1] = 0;
```

Програмски код који демонстрира процес померања елемената низа улево за **k** места:

```
int/float/double/... pomoc;   /* Tip zavisi od tipa elemenata niza  */
...

for(j = 0; j<k; j++){         /* k puta ponovi pomeranje za 1 mesto */
    pomoc = a[0];             /* Moze da se sacuva, ali ne mora     */
    for(i = 0; i<n-1; i++)
        a[i] = a[i+1];
    a[n-1] = 0;
}
```

Написати програм који елементе целобројног низа ротира улево за 1 место.

Максимални број елемената у низу је 50.

```
#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i;       /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Pomeranje elemenata niza ulevo za 1 mesto */
    p = a[0];
    for(i=0; i<n-1; i++)
        a[i] = a[i+1];

    a[n-1] = p;


    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

ДОДАТНА НАСТАВА

Ротирање елемената низа удесно за k места

Уколико нас интересује само крајњи резултат ротирања елемената низа за k места, тада можемо знатно да убрзамо сам процес.

Идеја се састоји у томе да се уведе помоћни низ.

Прва фаза се састоји у пребацивању k елемената са десног краја низа у помоћни низ.

Елемент a[n-1] се пребацује у p[k-1].

Елемент a[n-2] се пребацује у p[k-2].

...

Елемент a[n-k] се пребацује у p[0].

Друга фаза се састоји у померању преосталих n-k елемената низа за k места удесно.

Елемент a[n-k-1] се пребацује у a[n-1].

Елемент a[n-k-2] се пребацује у a[n-2].

...

Елемент a[0] се пребацује у a[k].

Трећа фаза се састоји у пребацивању k елемената из помоћног низа на првих k места низа.

Елемент p[k-1] се пребацује у a[k-1].

Елемент p[k-2] се пребацује у a[k-2].

...

Елемент p[0] се пребацује у a[0].

Укупан број пребацивања података приликом ротирања овом методом је:

k + (n - k) + k = **n + k** пребацивања!

Овакав начин ротирања елемената низа је знатно ефикаснији него раније описан поступак.

Раније описан поступак ротирања помоћу два циклуса захтева следећи број пребацивања података:

k * (n + 1) пребацивања!

Написати програм који елементе целобројног низа ротира удесно за k место.

Максимални број елемената у низу је 50.

```
/* Pomeranje elemenata niza za k mesta udesno */
/*                                            */

#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, j, k; /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], pomoc[MAX];  /* Promenljive u vezi sa nizom.          */
                             /* Tipovi ovih promenljivih zavise       */
                             /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Odredjivanje broja pomeranja elemenata u nizu */
    do{
        printf("\n\nUnesite za koliko mesta pomerate elemente niza udesno: ");
        scanf("%d", &k);            /* uvek je %d konverzija */
    }while(k<1);

    k %= n;                         /* k je u opsegu od 0 do n-1

    /* Pomeranje elemenata niza za k mesta */
    for(i=n-1, j=k-1 ; j>=0; i--, j--)  // izdvajanje k elemenata
        pomoc[j] = a[i];                // sa desne strane niza

    for(i=n-1; i>=k; i--)               // pomeranje elemenata za k mesta
        a[i] = a[i-k];

    for(i=0; i<k; i++)                  // ubacivanje izdvojenih k elemenata
        a[i] = pomoc[i];                // na pocetak niza

    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);   /* prva konverzija je uvek %d, */
                                           /* druga konverzija zavisi od  */
                                           /* tipa elementa niza          */

    return 0;
}
```

ДОДАТНА НАСТАВА

Ротирање елемената низа улево за k места

Уколико нас интересује само крајњи резултат ротирања елемената низа за k места, тада можемо знатно да убрзамо сам процес.

Идеја се састоји у томе да се уведе помоћни низ.

Прва фаза се састоји у пребацивању k елемената са левог краја низа у помоћни низ.

Елемент a[k-1] се пребацује у p[k-1].

Елемент a[k-2] се пребацује у p[k-2].

...

Елемент a[0] се пребацује у p[0].

Друга фаза се састоји у померању преосталих n-k елемената низа за k места улево.

Елемент a[k] се пребацује у a[0].

Елемент a[k+1] се пребацује у a[1].

...

Елемент a[n-1] се пребацује у a[n-k-1].

Трећа фаза се састоји у пребацивању k елемената из помоћног низа на првих k места низа.

Елемент p[k-1] се пребацује у a[n-1].

Елемент p[k-2] се пребацује у a[n-2].

...

Елемент p[0] се пребацује у a[n-k].

Укупан број пребацивања података приликом ротирања овом методом је:

k + (n - k) + k = **n + k** пребацивања!

Овакав начин ротирања елемената низа је знатно ефикаснији него раније описан поступак.

Раније описан поступак ротирања помоћу два циклуса захтева следећи број пребацивања података:

k * (n + 1) пребацивања!

Написати програм који елементе целобројног низа ротира улево за k место.

Максимални број елемената у низу је 50.

```
/* Pomeranje elemenata niza za k mesta ulevo */

#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, j, k; /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], pomoc[MAX];  /* Promenljive u vezi sa nizom.          */
                             /* Tipovi ovih promenljivih zavise       */
                             /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Odredjivanje broja pomeranja elemenata u nizu */
    do{
        printf("\n\nUnesite za koliko mesta pomerate elemente niza ulevo: ");
        scanf("%d", &k);            /* uvek je %d konverzija */
    }while(k<1);

    k %= n;                         /* k je u opsegu od 0 do n-1          */

    /* Pomeranje elemenata niza za k mesta ulevo                          */
    for(i=0; i<k; i++)              // izdvajanje k elemenata
        pomoc[i] = a[i];            // sa leve strane niza

    for(i=0; i<n-k; i++)            // pomeranje elemenata za k mesta
        a[i] = a[i+k];

    for(i=n-k, j=0; i<n; i++, j++)  // ubacivanje izdvojenih k elemenata
        a[i] = pomoc[j];          // na kraj niza

    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);   /* prva konverzija je uvek %d, */
                                           /* druga konverzija zavisi od  */
                                           /* tipa elementa niza          */

    return 0;
}
```

ДОДАТНА НАСТАВА

Ротирање елемената низа удесно/улево за k места – оптимална верзија

У овом делу разматраћемо како можемо да смањимо број основних операција померања елемената низа, а да при томе добијемо исти резултат.

Корисник може задати било коју вредност за **k**.

На пример, заротирати низ од 12 елемената за 111 места у десну страну.

Потребно је прво да одредимо колики је остатак целобројног дељења када се **k** подели са **n**.

```
k %= n;          /* k je u opsegu od 0 do n-1 */
```

12%111 даје број 3.

Заротирати низ од 12 елемената за 111 места у десну страну, исто је као и заротирати исти низ за 3 место у десну страну. Ако заротирамо низ од 12 елемената за 108 места у десну страну добићемо исти тај низ, само смо 9 пута заротирали све његове чланове у круг.

Број **k** треба да се сведе на опсег од **0** до **(n – 1) !**.

Даље смањивање броја основних операција померања елемената низа ако број места за ротирање елемената низа буде мањи од или једнак броју **n/2**, где је n број елемената низа.

Уколико неки низ ротирамо у десну страну за **k** места, тада се добија исти резултат као да смо низ ротирали за **(n – k)** места у леву страну!

Исто важи ако низ ротирамо у леву страну за **k** места, тада се добија исти резултат као да смо низ ротирали за **(n – k)** места у десну страну!

Ову особину можемо да искористимо ако желимо да смањимо број основних операција померања елемената низа.

Ако је **k** веће од **(n – k),** онда ћемо дати низ ротирати у супротну страну за **(n – k)** места!

Ако је **k** мање од **(n – k),** онда ћемо дати низ ротирати у супротну страну за **k** места!

Ако је **k** једнако **(n – k),** онда добијамо исти број основних операција померања елемената низа у обе стране. У овом случају ћемо обавити задато ротирање за **k** места!

Из горе наведеног следи да је највећи број места ротирања свих елемената низа **n/2**!

Последица тога је и величина помоћног низа, која сада може да буде дупло мања од максималне величине низа који се ротира!

Написати програм који елементе целобројног низа ротира за k место улево, односно удесно, према захтевима корисника.

Програм треба да оствари минималан број померања елемената низа.

Максимални број елемената у низу је 50.

```
/* Pomeranje elemenata niza za k mesta - univerzalno */
/*                                                   */

#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, j, k, smer; /* Promenljive u vezi sa indeksima niza. */
                          /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], pomoc[MAX];  /* Promenljive u vezi sa nizom.          */
                             /* Tipovi ovih promenljivih zavise       */
                             /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }


    /* Odredjivanje smera pomeranja elemenata u nizu */
    do{
        printf("\n\nUnesite smer pomeranja (1 - levo, 2 - desno): ");
        scanf("%d", &smer);            /* uvek je %d konverzija */
    }while(smer!=1 && smer!=2);


    /* Odredjivanje broja pomeranja elemenata u nizu */
    do{
        printf("\n\nUnesite za koliko mesta pomerate elemente niza: ");
        scanf("%d", &k);            /* uvek je %d konverzija */
    }while(k<1);

    k %= n;                         /* k je u opsegu od 0 do n-1          */

    if(k>n/2){
        smer = (smer==1)?2:1;       // Odredjivanje u koju stranu je manji
        k = n - k;                  // broj pomeranja i izracunavanje istog
    }
```

```
    if(smer == 1)    {
        /* Pomeranje elemenata niza za k mesta ulevo                          */
        for(i=0; i<k; i++)              // izdvajanje k elemenata
            pomoc[i] = a[i];            // sa leve strane niza

        for(i=0; i<n-k; i++)            // pomeranje elemenata za k mesta
            a[i] = a[i+k];

        for(i=n-k, j=0; i<n; i++, j++)  // ubacivanje izdvojenih k elemenata
            a[i] = pomoc[j];            // na kraj niza
    }
    else
    {
        /* Pomeranje elemenata niza za k mesta udesno */
        for(i=n-1, j=k-1 ; j>=0; i--, j--)  // izdvajanje k elemenata
            pomoc[j] = a[i];                // sa desne strane niza

        for(i=n-1; i>=k; i--)               // pomeranje elemenata za k mesta
            a[i] = a[i-k];

        for(i=0; i<k; i++)                  // ubacivanje izdvojenih k elemenata
            a[i] = pomoc[i];                // na pocetak niza
    }



    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);   /* prva konverzija je uvek %d, */
                                           /* druga konverzija zavisi od  */
                                           /* tipa elementa niza          */

    return 0;
}
```

Инвертовање елемената низа

Инвертовање елемената низа подразумева замену места елеменатима унутар низа, и то тако да први и последњи елемент замене места, други и предпоследњи елемент, и тако даље.

a[0] <---->a[n-1] међусобна замена места елемената са индексима 0 и n-1

a[1] <---->a[n-2] међусобна замена места елемената са индексима 1 и n-2

a[2] <---->a[n-3] међусобна замена места елемената са индексима 2 и n-3

a[3] <---->a[n-4] међусобна замена места елемената са индексима 3 и n-4

...

Пример: Дат је низ од 6 елемената и једна помоћна променљива (користи се у процесу замене места, односно вредности елемената)

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	
11	22	33	44	55	66	

Помоћу наредби циклуса (for, while, ...) размењујемо вредности једног по једног пара елемената.

У првом циклусу места/вредности ће заменити елементи a[0] и a[5]. Након обављене замене низ добија следећи изглед:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	наредба

66	22	33	44	55	**11**	`pomoc = a[0];`
						`a[0] = a[5];`
						`a[5] = pomoc;`

У другом циклусу места/вредности ће заменити елементи a[1] и a[4]. Изглед низа:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	наредба

66	**55**	33	44	**22**	11	`pomoc = a[1];`
						`a[1] = a[4];`
						`a[4] = pomoc;`

У трећем циклусу места/вредности ће заменити елементи a[2] и a[3]. Изглед низа:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	наредба

66	55	**44**	**33**	22	11	`pomoc = a[2];`
						`a[2] = a[3];`
						`a[3] = pomoc;`

За низ од 6 елемената потребно нам је 3 циклуса за инвертовање низа (6/2).

Пример: Дат је низ од 7 елемената и једна помоћна променљива (користи се у процесу замене места, односно вредности елемената)

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	
11	22	33	44	55	66	77	

Помоћу наредби циклуса (for, while, ...) размењујемо вредности једног по једног пара елемената.

У првом циклусу места/вредности ће заменити елементи a[0] и a[6]. Након обављене замене низ добија следећи изглед:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	наредба

77	22	33	44	55	66	**11**	`pomoc = a[0];`
							`a[0] = a[6];`
							`a[6] = pomoc;`

У другом циклусу места/вредности ће заменити елементи a[1] и a[5]. Након обављене замене низ добија следећи изглед:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	наредба

77	**66**	33	44	55	**22**	11	`pomoc = a[1];`
							`a[1] = a[5];`
							`a[5] = pomoc;`

У трећем циклусу места/вредности ће заменити елементи a[2] и a[4]. Након обављене замене низ добија следећи изглед:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	наредба

77	66	**55**	44	**33**	22	11	`pomoc = a[2];`
							`a[2] = a[4];`
							`a[4] = pomoc;`

Елемент a[3] остаје на месту на коме се налази, тј. на „средини“ низа. Над елементом a[3] се не примењују никакве наредбе у процесу инвертовања.

За низ од 7 елемената потребно нам је 3 циклуса да инвертујемо низ (7/2 – целобројно дељење).

Напомена: Гледајући наредбе које обављају инвертовање, примећује се да је у сваком циклусу збир индекса елемената који учествују у замени места/вредности увек једнак **n-1**!

У замени учествују елементи **a[i]** и **a[n-1-i]**.

Програмски код који обавља инвертовање елемената низа:

```
/* Zamena mesta elementima niza  - invertovanje */
    for(i=0; i<n/2; i++){
        p        = a[i];
        a[i]     = a[n-1-i];
        a[n-1-i] = p;
    }
```

Написати програм који инвертује елементе целобројног низа.

Максимални број елемената у низу је 50.

```
#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i;       /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Zamena mesta elementima niza */
    for(i=0; i<n/2; i++){
        p        = a[i];
        a[i]     = a[n-1-i];
        a[n-1-i] = p;
    }

    /* Prikaz invertovanog niza */
    printf("\n\nPrikaz invertovanog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Убацивање новог елемента у низ на k-ту позицију

Приликом убацивања новог елемента у низ на k-ту позицију могу да настану две ситуације.

Прва ситуација је да је тренутни број елемената у низу мањи од максималног броја. У том случају се прво помере сви елементи са индексима од (n-1) до k, а затим се на k-ту позицију упише нови елемент. Након тога се тренутни број елемената увећа за 1. У овој ситуацији долази до проширења низа.

Пример:

Низ има места за MAX = 10 елемената.

У низ је уписано n = 5 елемената.

Желимо да на позицију k = 3 убацимо broj = 66.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
11	22	33	44	55					

Прво се обави поступак померања елемената низа:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
11	22	33	44		55				

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
11	22	33		44	55				

Затим се упише број 66 у поље a[3]:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
11	22	33	66	44	55				

Број елемената у низу се увећава за 1 и постаје n = 6.

Код који обавља овакво убацивање новог елемента у низ гласи:

```
/* Ubacivanje broja u niz na k-tu poziciju */
for(i=n; i>k; i--)                // pomeranje elemenata za 1 mesto
    a[i] = a[i-1];
a[k] = broj;                // upisivanje novog elementa na k-tu poziciju
n++;                        // azuriranje broja elemenata u nizu
```

Друга ситуација која може да настане је да су сва поља низа попуњена.

Тада не можемо да убацимо нови елемент и повећамо број елемената у низу. Број елемената у низу остаје непромењен, крајњи десни елемент „нестаје“, остали елементи се померају за по 1 место удесно, све до позиције k. Затим се на позицију k упише нови елемент.

Пример:

Низ има места за MAX = 10 елемената.

У низ је уписано n = 10 елемената.

Желимо да на позицију k = 7 убацимо broj = 1212.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
00	11	22	33	44	55	66	77	88	99

Прво се обави поступак померања елемената низа:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
00	11	22	33	44	55	66	77		88

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
00	11	22	33	44	55	66		77	88

Затим се упише број 1212 у поље a[7].

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
00	11	22	33	44	55	66	1212	77	88

Број елемената у низу је остао непромењен!

Код који се може користити у обе ситуације гласи:

```
if(n<MAX) n++;                  /* da li ima mesta u nizu za ubacivanje */

/* Ubacivanje broja u niz na k-tu poziciju */
for(i=n-1; i>k; i--)           // pomeranje elemenata za 1 mesto
    a[i] = a[i-1];

a[k] = broj;                   // ubacivanje broja na k-tu poziciju
```

Написати програм који убацује нови елемент у целобројни низ на k-ту позицију у низу. Максимални број елемената у низу је 50.

```
#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, k, pozMaks;    /* Promenljive u vezi sa indeksima niza. */
                             /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], broj;        /* Promenljive u vezi sa nizom.          */
                             /* Tipovi ovih promenljivih zavise       */
                             /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Odredjivanje koji se broj ubacuje u niz */
    printf("\n\nUnesite broj koji ubacujete u niz: ");
    scanf("%d", &broj);            /* konverzija zavisi od tipa elementa niza */

    /* Odredjivanje poziciju ubacivanja elementa u niz */
    pozMaks = (n<MAX)?n:(n-1);
    do{
        printf("\nBroj mozete ubaciti na pozicije od 0 do %d.\n\n", pozMaks);
        printf("\n\nUnesite na koju poziciju ubacujete uneti broj u niz: ");
        scanf("%d", &k);            /* uvek je %d konverzija */
    }while(k<0 || k>pozMaks);
```

```
    if(n<MAX) n++;                  /* da li ima mesta u nizu za ubacivanje */

    /* Ubacivanje broja u niz na k-tu poziciju */
    for(i=n-1; i>k; i--)               // pomeranje elemenata za 1 mesto
        a[i] = a[i-1];

    a[k] = broj;                   // ubacivanje broja na k-tu poziciju

    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);   /* prva konverzija je uvek %d, */
                                           /* druga konverzija zavisi od  */
                                           /* tipa elementa niza          */

    return 0;
}
```

Ова верзија програма подржава све могуће ситуацију које могу да настану приликом убацивања новог елемента у низ.

Избацивање елемента из низа са k-те позиције (сажимање низа)

Приликом избацивања елемента из низа са k-те позиције долази до смањења низа.

Прво треба померити све елементе почев од позиције (индекса) k+1 па све до позиције n-1 за по 1 место улево.

Након тога број елемената у низу смањити за 1.

Пример:

Низ има места за MAX = 10 елемената.

У низ је уписано n = 6 елемената.

Желимо да избацимо елемент са позицију k = 3.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	
11	22	33	44	55	66	

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	
11	22	33	55		66	`a[3] = a[4];`

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	
11	22	33	55	66		`a[4] = a[5];`

a[0]	a[1]	a[2]	a[3]	a[4]		
11	22	33	55	66		n--;

Код који обавља избацивање елемента из низа гласи:

```
/* Izbacivanje broja iz niza sa k-te pozicije */
for(i=k; i<n-1; i++)                // pomeranje elemenata za 1 mesto
    a[i] = a[i+1];

n--;                    // azuriranje broja elemenata u nizu
```

Написати програм који избацује елемент из целобројног низа са k-те позиције у низу. Максимални број елемената у низу је 50.

```
/* Izbacivanje elementa sa k-te pozicije u nizu */

#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, k;             /* Promenljive u vezi sa indeksima niza. */
                             /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], broj;        /* Promenljive u vezi sa nizom.          */
                             /* Tipovi ovih promenljivih zavise       */
                             /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Odredjivanje pozicije izbacivanja elementa iz niza */
    do{
        printf("\nBroj mozete izbaciti sa pozicije od 0 do %d.\n\n", n-1);
        printf("\n\nUnesite sa koje pozicije u nizu izbacujete element: ");
        scanf("%d", &k);            /* uvek je %d konverzija */
    }while(k<0 || k>n-1);

    /* Izbacivanje broja iz niza sa k-te pozicije */
    for(i=k; i<n-1; i++)               // pomeranje elemenata za 1 mesto
        a[i] = a[i+1];

    n--;                  // azuriranje broja elemenata u nizu

    /* Prikaz novonastalog niza */
    printf("\n\nPrikaz novonastalog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);   /* prva konverzija je uvek %d, */
                                           /* druga konverzija zavisi od  */
                                           /* tipa elementa niza          */

    return 0;
}
```

Задаци за самосталан рад

Написати програм који претражује/ротира у леву страну за 1 или више места/ротира у десну страну за 1 или више места/инвертује елементе низа/убацује нови елемент на k-ту позицију у низу/избацује елемент са k-те позиције (најмање 8 могућности) елементе целобројног низа/низа реалних бројева једноструке тачности/низа реалних бројева двоструке тачности (3 могућности).

На располагању имате најмање **8 x 3 = 24** могућих варијанти задатака.

За максимални број елемената у низу узимајте различите вредности.

рб	**елементи низа**	**тип задатка**
1	int	претраживање низа и позиција свих појављивања
2	float	претраживање низа и позиција свих појављивања
3	double	претраживање низа и позиција свих појављивања
4	int	претраживање низа и позиција првог појављивања
5	float	претраживање низа и позиција првог појављивања
6	double	претраживање низа и позиција првог појављивања
7	int	претраживање низа и позиција последњег појављивања
8	float	претраживање низа и позиција последњег појављивања
9	double	претраживање низа и позиција последњег појављивања
10	int	ротирање удесно за 1 место
11	float	ротирање удесно за 1 место
12	double	ротирање удесно за 1 место
13	int	ротирање улево за 1 место
14	float	ротирање улево за 1 место
15	double	ротирање улево за 1 место
16	int	ротирање удесно за k места
17	float	ротирање удесно за k места
18	double	ротирање удесно за k места
19	int	ротирање улево за k места
20	float	ротирање улево за k места
21	double	ротирање улево за k места
22	int	инвертовање елемената низа
23	float	инвертовање елемената низа
24	double	инвертовање елемената низа
25	int	убацивање новог елемента на k-ту позицију (индекс)
26	float	убацивање новог елемента на k-ту позицију (индекс)
27	double	убацивање новог елемента на k-ту позицију (индекс)
28	int	избацивање елемента са k-те позиције (индекс)
29	float	избацивање елемента са k-те позиције (индекс)
30	double	избацивање елемента са k-те позиције (индекс)

Методе сортирања једнодимензионалних низова (вектора)

Циљ сортирања елемената низа јесте да се добије уређени распоред елемената низа.

Низ је растући, односно неопадајући, ако је **a[0] <= a[1] <= a[2] <= ... <= a[n-2] <= a[n-1]**.

Низ је опадајући, односно нерастући, ако је **a[0] >= a[1] >= a[2] >= ... >= a[n-2] >= a[n-1]**.

Постоји више метода сортирања, које се међусобно разликују по ефикасности, односно потребном броју наредби да се обави сортирање елемената низа.

Упознаћемо неке од метода за сортирање, и то:

метода избора, селекције (selection sort)

метода избора, селекције (selection sort) – побољшана метода

метода уметања (insertion sort)

метода уметања (insertion sort) - побољшана метода

метода замене суседа (bubble sort)

метода поделе (реализује се као рекурзивна функција)

selection sort

```
/* Sortiranje elemenata niza */
    for(i=0; i<n-1; i++)
        for(j=i+1; j<n; j++)
            if(a[i]>a[j])           /* znak > u pitanju se postavlja za */
            {                       /* sortiranje po rastucem redosledu */
                p    = a[i];
                a[i] = a[j];
                a[j] = p;
            }
```

Метода селекције заснива се на следећем принципу. Одабрати најмањи (највећи) елемент и довести га на прво место (индекс 0), затим други најмањи (највећи) елемент на друго место (индекс 1), трећи најмањи (највећи) елемент на треће место (индекс 2), и тако даље до краја низа.

Објашњење кода дато је за сортирање по растућем редоследу.

`for(i=0; i<n-1; i++)`	индекс места у низу на који желимо да доведемо одговарајући елемент
`for(j=i+1; j<n; j++)`	индекси елемената у преосталом делу низа
`if(a[i]>a[j])`	уколико се пронађе елемент који који је мањи од a[i] онда
`{`	

`p = a[i];`
`a[i] = a[j];`
`a[j] = p;`
	заменити места елементима низа a[i] и a[j]

`}`	

За сортирање по опадајућем редоследу у наредби **if** треба окренути знак за поређење:

```
            if(a[i]<a[j])
```

selection sort – побољшана метода

```
/* Sortiranje elemenata niza */
    for(i=0; i<n-1; i++){
        m=i;
        for(j=i+1; j<n; j++)
            if(a[m]>a[j])       /* znak > u pitanju se postavlja za */
                m = j;          /* sortiranje po rastucem redosledu */
        if(m != i){             /* ako smo pronasli manji element,   */
            p    = a[i];        /* tek onda menjamo mesta elementima */
            a[i] = a[m];
            a[m] = p;
        }
    }
```

За сортирање по опадајућем редоследу у наредби **if** треба окренути знак за поређење:

```
            if(a[m]>a[j])
```

Побољшање метода селекције огледа се у томе да се уводи помоћна променљива m која има задатак да чува индекс најмањег елемента у преосталом делу низа.

На почетку спољашњег циклуса променљива **m** добија вредност **i**.

Затим се у преосталом делу низа испитује да ли постоји елемент **a[j]** који је мањи (већи) од **a[m]**.

Уколико пронађемо такав елемент онда се у променљиву **m** уписује вредност индекса тог елемента (**m = j**).

Уколико смо пронашли у преосталом делу низа мањи (већи) елемент онда се вредност променљиве **m** променила и тада ћемо заменити вредности елементима низа. У једном пролазу (спољашња петља) може се само једном заменити вредности елементима низа.

Уколико нисмо пронашли у преосталом делу низа мањи (већи) елемент онда вредност променљиве m остаје иста и тада нема замене вредности елементима низа.

Прелази се на нову вредност променљиве **i**, односно тражење следећег најмањег (највећег) елемента низа.

insertion sort

```
/* Sortiranje elemenata niza */
    for(i=1; i<n; i++)
        for(j=i-1; j>=0 && a[j]>a[j+1]; j--){
            p    = a[j];          /* znak > u pitanju se postavlja za */
            a[j] = a[j+1];        /* sortiranje po rastucem redosledu */
            a[j+1] = p;
        }
```

За сортирање по опадајућем редоследу у наредби **for** треба окренути знак за поређење:

```
                        a[j]<a[j+1]
```

Метода уметања заснива се на идеји да се елементи низа умећу један по један, тако да оформљени део низа сваког момента буде уређен.

Унутрашњи циклус не траје увек до крајњих граница. Циклус може да се заврши и пре времена, ако се утврди да је управо испитана вредност стигла на своје место у раније већ уређеном делу низа.

insertion sort - побољшана метода

```
/* Sortiranje elemenata niza */
    for(i=1; i<n; i++){
        p = a[i];
        for(j=i-1; j>=0 && a[j]>p; j--)
            a[j+1] = a[j];           /* znak > u pitanju se postavlja za */
                                     /* sortiranje po rastucem redosledu */
        a[j+1] = p;
    }
```

За сортирање по опадајућем редоследу у наредби **for** треба окренути знак за поређење:

```
a[j]<p
```

Побољшање методе уметања огледа се у томе што се елемент **a[i]** у помоћну променљиву **p**. Тиме смо постигли у унутрашњој петљи да није потребно међусобно замењивање два по два суседна елемента, већ само премештање елемената **a[j+1] = a[j]**, све докле је **a[j]>p**. Након завршетка унутрашње петље вредност помоћне променљиве се уписује на одговарајуће место **a[j+1] = p**.

bubble sort

```
/* Sortiranje elemenata niza */
    for(dalje=1, i=0; i<n-1 && dalje; i++){
        for(dalje=0, j=n-1; j>i; j--)
            if(a[j-1]>a[j]){          /* znak > u pitanju se postavlja za */
                p      = a[j-1];      /* sortiranje po rastucem redosledu */
                a[j-1] = a[j];
                a[j]   = p;
                dalje  = 1;
            }
    }
```

За сортирање по опадајућем редоследу у наредби **if** треба окренути знак за поређење:

```
            if(a[j-1]<a[j])
```

Постоји више варијанти методе замене суседа. Заједничко за све методе је:

Пролази се кроз низ са једног краја на други. Упоређују се два по два суседна елемента. Уколико два суседна елемента нису у одговарајућем поретку, обавља се њихова замена.

Овај поступак се понавља све док у једном пролазу кроз низ више не треба замењивати ни један пар суседа.

Уводи се променљива **dalje** која ће нам рећи да ли треба спољашњи циклус наставити још једном.

На почетку сваког унутрашњег циклуса променљива **dalje** добија вредност **0**. Уколико се обави макар једна замена током унутрашњег циклуса, тада променљива **dalje** добија вредност **1**, што је знак да се спољашњи циклус још једном покрене.

Уколико се не обави нити једна замена током унутрашњег циклуса, променљива **dalje** задржава вредност **0** и самим тим неће се започети нови спољашњи циклус, већ се процес сортирања зауставља.

Написати програм који сортира методом селекције елементе целобројног низа.

Максимални број елемената у низу је 50.

```
/* Primenom metode izbora (selekcije) sortirati elemente */
/* unetog celobrojnog niza po rastucem redosledu         */
/*                    selection sort                     */


#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, j;    /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Sortiranje elemenata niza */
    for(i=0; i<n-1; i++)
        for(j=i+1; j<n; j++)
            if(a[i]>a[j])           /* znak > u pitanju se postavlja za */
            {                       /* sortiranje po rastucem redosledu */
                p    = a[i];
                a[i] = a[j];
                a[j] = p;
            }

    /* Prikaz sortiranog niza */
    printf("\n\nPrikaz sortiranog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Написати програм који сортира побољшаном методом селекције елементе целобројног низа. Максимални број елемената у низу је 50.

```
/* Primenom metode izbora (selekcije) sortirati elemente */
/* unetog celobrojnog niza po rastucem redosledu.        */
/* Izmenjena varijanta metoda izbora (selekcije)         */
/*                    selection sort                     */


#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, m, i, j; /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Sortiranje elemenata niza */
    for(i=0; i<n-1; i++){
        m=i;
        for(j=i+1; j<n; j++)
            if(a[m]>a[j])           /* znak > u pitanju se postavlja za */
                m = j;              /* sortiranje po rastucem redosledu */
            if(m != i){             /* ako smo pronasli manji element,   */
                p    = a[i];        /* tek onda menjamo mesta elementima */
                a[i] = a[m];
                a[m] = p;
            }
    }

    /* Prikaz sortiranog niza */
    printf("\n\nPrikaz sortiranog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Написати програм који сортира методом уметања елементе целобројног низа.

Максимални број елемената у низу је 50.

```
/* Primenom metode umetanja (insertion) sortirati elemente */
/* unetog celobrojnog niza po rastucem redosledu.          */
/*                    insertion sort                       */


#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, m, i, j; /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Sortiranje elemenata niza */
    for(i=1; i<n; i++)
        for(j=i-1; j>=0 && a[j]>a[j+1]; j--){
            p    = a[j];           /* znak > u pitanju se postavlja za */
            a[j] = a[j+1];         /* sortiranje po rastucem redosledu */
            a[j+1] = p;
            }

    /* Prikaz sortiranog niza */
    printf("\n\nPrikaz sortiranog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Написати програм који сортира побољшаном методом уметања елементе целобројног низа. Максимални број елемената у низу је 50.

```
/* Primenom poboljsane metode umetanja (insertion) */
/* sortirati elemente unetog celobrojnog niza      */
/* po rastucem redosledu.                          */
/*                insertion sort                   */


#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, m, i, j; /* Promenljive u vezi sa indeksima niza. */
                    /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Sortiranje elemenata niza */
    for(i=1; i<n; i++){
        p = a[i];
        for(j=i-1; j>=0 && a[j]>p; j--)
            a[j+1] = a[j];           /* znak > u pitanju se postavlja za */
                                     /* sortiranje po rastucem redosledu */
        a[j+1] = p;
    }

    /* Prikaz sortiranog niza */
    printf("\n\nPrikaz sortiranog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Написати програм који сортира методом замене суседа елементе целобројног низа.

Максимални број елемената у низу је 50.

```
/* Primenom metode zamene suseda sortirati elemente */
/* unetog celobrojnog niza po rastucem redosledu.   */
/*                   bubble sort                    */


#include <stdio.h>

#define MAX 50      /* Definisanje najveceg broja elementa u nizu */

int main(void)
{
    int n, i, j, dalje; /* Promenljive u vezi sa indeksima niza. */
                        /* Ove promenljive su uvek celobrojne!!! */

    int a[MAX], p;  /* Promenljive u vezi sa nizom.          */
                    /* Tipovi ovih promenljivih zavise       */
                    /* od tipa elemenata niza                */

    /* Odredjivanje broja elemenata u nizu */
    do{
        printf("\n\nUnesite velicinu niza (max = %d): ", MAX);
        scanf("%d", &n);            /* uvek je %d konverzija */
    }while(n<1||n>MAX);

    /* Ucitavanje elemenata niza, jedan po jedan */
    printf("\n\nUnos elemenata niza:");
    for(i=0; i<n; i++){
        printf("\na[%d] = ", i);    /* uvek je %d konverzija */
        scanf("%d", &a[i]);         /* konverzija zavisi od tipa elementa niza */
    }

    /* Sortiranje elemenata niza */
    for(dalje=1, i=0; i<n-1 && dalje; i++){
        for(dalje=0, j=n-1; j>i; j--)
            if(a[j-1]>a[j]){          /* znak > u pitanju se postavlja za */
                p      = a[j-1];      /* sortiranje po rastucem redosledu */
                a[j-1] = a[j];
                a[j]   = p;
                dalje  = 1;
            }
    }

    /* Prikaz sortiranog niza */
    printf("\n\nPrikaz sortiranog niza:\n");
    for(i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);    /* prva konverzija je uvek %d, */
                                            /* druga konverzija zavisi od  */
                                            /* tipa elementa niza          */

    return 0;
}
```

Задаци за самосталан рад

Написати програм који сортира по растућем/опадајућем редоследу (2 могућности) елементе целобројног низа/низа реалних бројева једноструке тачности/низа реалних бројева двоструке тачности (3 могућности) помоћу методе селекције/побољшане методе селекције/методе уметања/побољшане методе уметања/методе замене суседа (5 могућности).

На располагању имате **2 x 3 x 5 = 30** могућих варијанти задатака.

За максимални број елемената у низу узимајте различите вредности.

рб	**редослед**	**елементи низа**	**метод сортирања**
1	растући	int	метода селекције
2	опадајући	int	метода селекције
3	растући	float	метода селекције
4	опадајући	float	метода селекције
5	растући	double	метода селекције
6	опадајући	double	метода селекције
7	растући	int	побољшана метода селекције
8	опадајући	int	побољшана метода селекције
9	растући	float	побољшана метода селекције
10	опадајући	float	побољшана метода селекције
11	растући	double	побољшана метода селекције
12	опадајући	double	побољшана метода селекције
13	растући	int	метода уметања
14	опадајући	int	метода уметања
15	растући	float	метода уметања
16	опадајући	float	метода уметања
17	растући	double	метода уметања
18	опадајући	double	метода уметања
19	растући	int	побољшана метода уметања
20	опадајући	int	побољшана метода уметања
21	растући	float	побољшана метода уметања
22	опадајући	float	побољшана метода уметања
23	растући	double	побољшана метода уметања
24	опадајући	double	побољшана метода уметања
25	растући	int	метода замене суседа
26	опадајући	int	метода замене суседа
27	растући	float	метода замене суседа
28	опадајући	float	метода замене суседа
29	растући	double	метода замене суседа
30	опадајући	double	метода замене суседа