Rad s nizom znakova

- Napravite niz duljine do 80 znakova koje unesete, upišete znak koji tražite u nizu i napišite program koji pretražuje niz znakova i javlja poziciju na kojoj se traženi znak nalazi, ukoliko se traženi znak ne nalazi u nizu, obavještava Vas o tome

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define L 80
int traziznak(char *niz, char *znak, int nizl);

int main(){
    char *tekst;
    char *ctrazeni;
    int poz=0;
    int nizl;
    if ((tekst = (char *)malloc(L+1)) == NULL)
        {printf("Greska u alociranju memorije !");
      exit (1); }
if ((ctrazeni = (char *)malloc(2)) == NULL)
        {printf("Greska u alociranju memorije !");
      exit (1); }
```

```
    printf("Unesi do %d znaka:\n",L);
    gets(tekst);
    printf("Unesi znak koji trazis:\n");
    gets(ctrazeni);
    nizl=strlen(tekst);
    printf("Uneseni niz ima %d znakova\n",nizl);

    poz = traziznak(tekst,ctrazeni,nizl);
    if (!poz)
            printf("Znak %s se ne nalazi u nizu %s\n",ctrazeni,tekst);
            else
    printf ("Znak %s je na mjestu %d u nizu %s\n",ctrazeni,poz,tekst);
    system("PAUSE");
    return 0;
}

int traziznak(char *niz, char *znak, int nizl){
    int ind;
    for (ind=0; ind <= nizl; ind++)
            if (niz[ind] == *znak) return ind+1;
    return 0;
}
```

Dvodimenzionalna i višedimenzionalna polja kao argumenti funkcije

- Polje u funkciji treba prihvatiti kao jednodimenzionalno polje ili pokazivač.
- Primjer:

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

u memoriji računala spremljeno je kao

tj. jednako kao jednodimenzionalno polje od 12 elemenata.

- Razlikuju se fizičke dimenzije polja od logičkih, koje mogu biti i manje. Za pozicioniranje je potrebno znati fizički broj stupaca `(MAX_broj_stupaca)`.
- Za dohvat elementa iz *i*-tog retka treba preskočiti `i-1` punih redaka.
- Kako prvi redak ima indeks 0, drugi 1, treći 2 itd., za dohvat elemenata retka s indeksom *i* treba preskočiti `i*MAX_broj_stupaca` članova polja.
- Općenito:

```
dvodim_polje[i][j] ≡ jednodim_polje[i * MAX_broj_stupaca + j]
```

Zbrajanje i množenje matrica

- Napišite program koji će rezervirati u memoriji mjesto za tri dvodimenzionalna polja (matrice) A, B i C dimenzija maxstu. Polja A i B popuniti slučajno generiranim cijelim brojevima u rasponu od 0 do maxcl. Napisati funkciju koja zbroji matrice A i B i rezultat pospremi u polje C. Napisati funkciju koja umnožak matrica A*B pospremi u polje C. Ispisati sva polja.

- Suma: $c_{ij} = a_{ij} + b_{ij}$

- Produkt: $$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#define maxstu 4
#define maxcl 5

void prikazi(int *M){
int i,j;

for (i=0;i<maxstu;i++)  {
   for (j=0;j<maxstu;j++)
       printf("%d ",*(M+i*maxstu+j));
   printf("\n");
  }
printf("\nPritisnite Enter za nastavak!\n");
scanf("%*c");
}

void zbroji(int *A,int *B,int *C){
int i,j;

 for (i=0;i<maxstu;i++)
    for (j=0;j<maxstu;j++)
     *(C+i*maxstu+j)=*(A+i*maxstu+j)+*(B+i*maxstu+j);
}
```

```
void mnozi(int *A,int *B,int *C){
int i,k,j;

 for (i=0;i<maxstu;i++)
    for (j=0;j<maxstu;j++)
       for (k=0;k<maxstu;k++)
     *(C+i*maxstu+j)+=*(A+i*maxstu+k) * *(B+k*maxstu+j);
}

int main(){
int i,j;
int A[maxstu][maxstu];
int B[maxstu][maxstu];
int C[maxstu][maxstu]={0};

srand(time(NULL));
for (i=0;i<maxstu;i++)
   for (j=0;j<maxstu;j++)   {
      srand(time(NULL) * (float) rand() / (RAND_MAX+1));
      A[i][j]=maxcl * ((float) rand() / (RAND_MAX+1));
      B[i][j]=maxcl * ((float) rand() / (RAND_MAX+1));
   }
```

```
printf("\n------------------A------------------\n\n");
prikazi(&A[0][0]);
printf("\n------------------B------------------\n\n");
prikazi(&B[0][0]);
printf("\n------------------C------------------\n\n");
prikazi(&C[0][0]);

zbroji(&A[0][0],&B[0][0],&C[0][0]);
printf("\n----------------C=A+B-------------------\n\n");
prikazi(&C[0][0]);

for (i=0;i<maxstu;i++)
  for (j=0;j<maxstu;j++)
      C[i][j]=0;

mnozi(&A[0][0],&B[0][0],&C[0][0]);
printf("\n----------------C=A*B-------------------\n\n");
prikazi(&C[0][0]);

return 0;
}
```

Maksimalna granična vrijednost: primjer s kompleksnim brojevima

- Međurezultati množenja u nekim izrazima mogu biti veći od maksimalne granične vrijednosti za neki tip podataka pa dolazi do prelijevanja (overflow) znamenki, iako je konačni rezultat unutar granica koje se mogu predstaviti tim tipom podataka
- primjer množenja kompleksnih brojeva: do prelijevanja će doći samo kada je i krajnji rezultat blizu maksimalne granične vrijednosti
- Dobar primjer je algoritam za određivanje modula kompleksnog broja dan izrazom:

$$|a + ib| = \sqrt{a^2 + b^2}$$

U rješavanju gornjeg izraza često će međurezultat (kvadrat) biti veći od graničnog.

Ispravan način na koji ovaj račun ostaje unutar granica je:

$$|a + ib| = \begin{cases} |a|\sqrt{1 + (b/a)^2} & |a| \geq |b| \\ |b|\sqrt{1 + (a/b)^2} & |a| < |b| \end{cases}$$

Isti problem se javlja i kod dijeljenja kompleksnih brojeva:

$$\frac{a+ib}{c+id} = \begin{cases} \dfrac{[a+b(d/c)]+i[b-a(d/c)]}{c+d(d/c)} & |c| \geq |d| \\ \dfrac{[a(c/d)+b]+i[b(c/d)-a]}{c(c/d)+d} & |c| < |d| \end{cases}$$

Ispravan postupak računanja je:

$$w \equiv \begin{cases} 0 & c = d = 0 \\ \sqrt{|c|}\sqrt{\dfrac{1+\sqrt{1+(d/c)^2}}{2}} & |c| \geq |d| \\ \sqrt{|d|}\sqrt{\dfrac{|c/d|+\sqrt{1+(c/d)^2}}{2}} & |c| < |d| \end{cases}$$

$$\sqrt{c+id} = \begin{cases} 0 & w = 0 \\ w + i\left(\dfrac{d}{2w}\right) & w \neq 0, c \geq 0 \\ \dfrac{|d|}{2w} + iw & w \neq 0, c < 0, d \geq 0 \\ \dfrac{|d|}{2w} - iw & w \neq 0, c < 0, d < 0 \end{cases}$$

Linearno pretraživanje polja

- Napiši funkciju Search koja uzima sljedeće ulazne parametre:
  - polje cijelih brojeva a[]
  - duljinu length polja a[]
  - cijeli broj x

  Funkcija kao rezultat svog rada treba vratiti indeks onog elementa polja a[] u kojemu se nalazi broj x, odnosno -1 ukoliko se broj x ne nalazi u polju a[].

- Pretpostavimo da imamo polje **nesortiranih** elemenata i da želimo znati nalazi li se ili ne nalazi određeni zadani element u tom polju i ako se nalazi, želimo znati njegovu poziciju (indeks) u tom polju. Obzirom na to da polje nije sortirano, ne preostaje nam ništa drugo nego da "obiđemo" polje element po element i uspoređujemo vrijednost svakog elementa s brojem koji tražimo. Pri tom obilasku mogu nam se dogoditi 2 stvari:

  u nekom trenutku možemo naići na element u polju koji ima vrijednost koju tražimo. Tada zapamtimo njegov indeks i prestajemo s radom, ili

  došli smo do kraja polja i nigdje u njemu nismo našli traženi broj. Tada na odgovarajući način signaliziramo da traženi broj nije nađen i prestajemo s radom.

  Mana ovakvog pretraživanja je da za pronalaženje elementa moramo u najgorem slučaju obići čitavo polje, što može biti sporo za jako velika polja.

```
#include <stdio.h>
#include <stdlib.h>
int Search(int a[], int length, int key);

int main () {
      int N, i, trazeni, rez;
      int *polje;
      printf("Unesi duljinu polja:\n");
       scanf("%d",&N);
              if ((polje = (int *)malloc(N*sizeof(int))) == NULL){
                printf("Greska u alociranju memorije !");
               exit (1); }
       for (i = 0; i<N; i++)        {
              printf("Unesi %d. broj:",i);
              scanf("%d", &polje[i]);  }
         printf("\n");
         printf("Unesi vrijednost koju trazis:");
         scanf("%d", &trazeni);
        rez = Search(polje, N, trazeni);
        if (rez != -1)
            printf("Broj %d se nalazi u polju na poziciji %d\n", trazeni,rez);
        else
            printf("Broj %d se ne nalazi u polju\n", trazeni);
        system("PAUSE"); return 0;
        }
```

```
int Search(int a[], int lenght, int key)   {
            for (int i = 0; i < lenght; i++)
                if (a[i] == key) return i;
                        return -1;
        }
```

Primjeri rekurzija

- Primjeri jednostavnih rekurzivnih funkcija za koje treba odrediti što će ispisati i koju će vrijednost sadržavati varijabla nula nakon povratka u glavnu proceduru. Također odgovoriti na pitanja:
  a) Kakav tip podatka sadrži *broj?
  b) Kakav tip podatka sadrži broj?
  c) Kakav tip podatka sadrži &nula?

```
#include <stdio.h>
#include <stdlib.h>
void pisi1 (int broj, int n) {
  broj++;
  if (broj > n) return;
  pisi1 (broj, n);
  printf( "%d ", broj);
}
void pisi2 (int broj, int n) {
  broj++;
  if (broj > n) return;
  printf( "%d ", broj);
  pisi2 (broj, n);
}
void pisi3 (int *broj, int n) {
  (*broj)++;
  if (*broj > n) return;
  pisi3 (broj, n);
  printf( "%d ", *broj);
}
void pisi4 (int *broj, int n) {
  (*broj)++;
  if (*broj > n) return;
  printf( "%d ", *broj);
  pisi4 (broj, n);
}
```

```
void pisi5 (int *broj, int n) {
  (*broj)--;
  if (*broj < 0) return;
  pisi5 (broj, n);
  printf( "%d ", *broj);
}

void main (void) {
  int nula;
      nula = 0; pisi1 (nula, 10);
      printf(" Nakon pisi1 nula = %d\n", nula);
      nula = 0; pisi2 (nula, 10);
      printf(" Nakon pisi2 nula = %d\n", nula);
       nula = 0; pisi3 (&nula, 10);
      printf(" Nakon pisi3 nula = %d\n", nula);
      nula = 0; pisi4 (&nula, 10);
      printf(" Nakon pisi4 nula = %d\n", nula);
      nula=10; pisi5 (&nula, 10);
      printf(" Nakon pisi5 nula = %d\n", nula);
     system("PAUSE");
      exit (0);
}
```