Zadaci za zadaću

- Napisati program koji učita 10 nizova znakova duljine do 20 znakova. Svi unešeni znakovi su samo alfabetski standardni ASCII znakovi (mala i velika slova). Sortirati te nizove znakova po abecedi i ispisati ih. Uputa: prebaciti sve unešene znakove u velika ili mala slova prije sortiranja i tako ih sortirati i ispisati.

- Napisati program koji će upotrebom velikog broja generiranih parova slučajnih brojeva izračunati površinu kruga radijusa r s centom u središtu koordinatnog sustava (0,0). Usporediti dobiveni rezultat s izračunatom pravom vrijednošću površine kruga.

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);
        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);
}
```

Još primjera rekurzija

- Nekoliko jednostavnih primjera rekurzija s predavanja:
  - Rekurzivno traženje indeksa člana u polju
  - Rekurzivno traženje indeksa člana u polju s ograničivaćem
  - Rekurzivno traženje najvećeg člana polja
  - Rekurzivno traženje najvećeg člana polja – strukturirano
  - Primjer neispravne rekurzije

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXA 15

// ispis znaka c u zadanoj duljini n
void nznak (int c, int n) {
  while (--n >= 0) putchar(c);
}

// ispis polja
void ispisi(int A[], int n) {
      int i;
      printf("\n");
      for (i = 0; i < n; i++) printf(" A[%d]",i);
      printf("\n");
      for (i = 0; i < n; i++) printf("%5d", A[i]);
      printf("\n");
}
```

```
// Rekurzivno trazenje indeksa clana u polju
int trazi (int A[], int x, int n, int i) {     // A-polje, x-trazeni, i-indeks od kojeg se trazi
     int ret;
     nznak(' ', i*5); printf("^^^^^\n");
     if (i >= n)
              ret = -1;
     else if (A[i] == x)
              ret = i;
     else
             ret = trazi (A, x, n, i+1);
     nznak(' ', i*5); printf("%5d\n", ret);
  return ret;
}

// Rekurzivno trazenje indeksa clana u polju s ogranicivacem
int trazi1 (int A[], int x, int i){
     int ret;
     nznak(' ', i*5); printf("^^^^^\n");
     if(A[i] == x)
              ret = i;
     else
              ret= trazi1 (A, x, i+1);
  nznak(' ', i*5); printf("%5d\n", ret);
  return ret;
}
```

```
// Rekurzivno trazenje najveceg clana polja
int maxclan (int A[], int i, int n) {
  int imax;
  if (i >= n-1)
        return n-1;
  imax = maxclan (A, i + 1, n);
  if (A[i] > A[imax])
        return i;
  return imax;
}

// Rekurzivno trazenje najveceg clana polja - strukturirano
int maxclan1 (int A[], int i, int n) {
  int imax, ret;
     printf ("max(%d) -> ", i);
  if (i >= n-1) {
              printf ("\n");
              ret = n-1;
     } else {
           imax = maxclan1 (A, i + 1, n);
           if (A[i] > A[imax])
               ret = i;
           else
              ret = imax;  }
     printf ("<- max(%d)=%d ", i, ret);
  return ret; }
```

```
// macro naredba za vecu od dvije vrijednosti
#define maxof(a,b) ((a) > (b) ? (a) : (b))

// Funkcija s macro naredbom koja vraca vrijednost najveceg clana
int maxclan2 (int A[], int i, int n) {
      int m;
  if (i >= n-1) return A[i];
      m = maxclan2 (A, i + 1, n);
  return maxof(A[i], m);
}

// Primjer neispravne rekurzije
int los (int n, int *dubina) {
      int r;
      (*dubina)++;
      printf ("n = %d, dubina rekurzije = %d\n", n, *dubina);
      if (n == 0)
            r = 0;
      else
            r = los (n / 3 + 1, dubina) + n - 1;
  return r;
}
```

```
void main () {
  int A[MAXA], x, i, n, dubina;
  FILE *fi;
  fi = fopen ("UlazZaRekurzije.txt", "r");
  if (!fi) exit (1);
  n = 0;
  while (n < MAXA - 1 && fscanf (fi, "%d", &A[n]) != EOF)    n++;
  fclose (fi);
  ispisi (A, n);
     printf ("Upisite vrijednost za x =");  scanf ("%d", &x);
     printf ("\nRekurzivno trazenje indeksa clana\n");
     ispisi (A, n);
     if ((i = trazi (A, x, n, 0)) < 0) {
              printf ("Vrijednost %d ne postoji u polju\n", x);
     } else {
              printf ("A [%d] = %d\n", i, A [i]);          }
  printf ("\nRekurzivno trazenje ... s ogranicivacem\n");
     A [n] = x;  // postavljanje ogranicivaca
     ispisi (A, n+1);
     if ((i = trazi1 (A, x, 0)) == n) {
              printf ("Vrijednost %d ne postoji u polju", x);
     } else {
              printf ("A [%d] = %d\n", i, A [i]);          }
```

```
printf ("\nRekurzivno trazenje najveceg...\n");
    ispisi(A, n);
    if ((i = maxclan (A, 0, n)) != maxclan1 (A, 0, n)) {
        printf ("Pogreska: Strukturirana i nestrukturirana funkcija daju razlicite rezultate!\n");
        exit (0);
    }
  printf ("\nNajveci clan A [%d] = %d\n",i, A [i]);
printf ("Funkcija s macro naredbom je nasla najveci clan %d\n", maxclan2 (A, 0, n));

    printf ("\nPozivam neispravnu rekurziju\n");
        dubina = 0;
        printf ("Upisite vrijednost za n =");
        scanf ("%d", &n);
        i = los (n, &dubina);
        printf ("\ni = %d", i);
    exit(0);
}
```