Operacije s nizovima podataka

Operacije s nizovima podataka

- Nakon odslušanog bit ćete u stanju:
  - objasniti algoritam za
    - zbroj svih članova niza
    - pronalaženje najvećeg (najmanjeg) člana niza
  - analizirati i implementirati algoritme traženja
    - sekvencijalno
    - binarno
  - analizirati i implementirati algoritme sortiranja
    - traženjem ekstrema
    - zamjenom susjednih članova.

Zbroj svih članova niza

- Pretpostavka: $n \geq 0$

```
.....
zbroj = 0.0;

for(i = 0; i < n; i++)
      zbroj += a[i];
printf("Zbroj clanova niza je %f.\n", zbroj);
```

Najmanji (najveći) član niza

```
.....
min = x[0]; poz = 0;

for(i = 1; i < n; i++)
    if(x[i] < min){
            min = x[i];
            poz = i;
    };

printf("Najmanji clan niza: x[%d] = %f\n", poz, min);
```

Pretraživanje (traženje)

- Zadan je niz bilo kakvih objekata i još jedan objekt `a` istog tipa. Tražimo odgovor na pitanje pojavljuje li se objekt `a` među članovima zadanog niza.

  Primjer: sekvencijalno (slijedno) pretraživanje: a = 5

niz[0]	niz[1]	niz[2]	niz[3]	niz[4]
-2	4	1	5	3
-2	4	1	5	3
-2	4	1	5	3
-2	4	1	5	3

Sekvencijalno pretraživanje

```
.....
pronasli = i = 0;

while(!pronasli && i < n)
    if(niz[i] == a) pronasli = 1; else i++;

if(pronasli) printf("Elemenat a je pronadjen.\n");
else printf("Elemenat a nije pronadjen.\n");
```

Primjer: binarno pretraživanje

- a = 5

niz[0]	niz[1]	niz[2]	niz[3]	niz[4]	niz[5]	niz[6]	niz[7]	niz[8]	niz[9]
-3	-1	0	2	4	5	7	13	16	20
-3	-1	0	2	4	5	7	13	16	20
-3	-1	0	2	4	5	7	13	16	20
-3	-1	0	2	4	5	7	13	16	20
-3	-1	0	2	4	5	7	13	16	20
-3	-1	0	2	4	5	7	13	16	20

Binarno pretraživanje (2)

```
……
d = 0;
g = n - 1;
pronadjen = 0;

while(!pronadjen && d <= g){
      s = (d + g)/2;
      if(niz[s] == a) pronadjen = 1;
      else if(niz[s] < a) d = s + 1; else g = s - 1;
}

if(pronadjen) printf("Elemenat je pronadjen.\n");
else printf("Elemenat nije pronadjen.\n");
```

Binarno pretraživanje - funkcija

```
int binarno_pretrazivanje(int niz[], int n, int a){
   int s, d, g;
   d = 0;
   g = n - 1;

   while(d <= g){
        s = (d + g)/2;
        if(niz[s] == a) return 1;
        else if(niz[s] < a) d = s + 1; else g = s - 1;
   }
   return 0;
}

   ……
   if(binarno_pretrazivanje(niz, n, a)) ……
```

Binarno pretraživanje – funkcija (2)

```
int binarno_pretrazivanje(int niz[],int a, int d, int g){
   int s;

   if(d <= g){
        s = (d + g)/2;
        if(niz[s] == a)  return 1;
        else if(niz[s] < a) binarno_pretrazivanje(niz, a, s + 1, g);
                else binarno_pretrazivanje(niz, a, d, s - 1);}
   else return 0;
}

   ……
   if(binarno_pretrazivanje(niz, a, 0, n - 1)) ……
```

Sortiranje nizova

- Sortiramo da bismo brže pretraživali. Npr. na 1000000 podataka, maksimalni broj usporedbi za sekvencijalno pretraživanje je 10^6, a za binarno 21.
- Najjednostavniji algoritam – tzv. sortiranje traženjem ekstrema (Selection sort):

niz[0]	niz[1]	niz[2]	niz[3]	niz[4]
3	4	-2	5	0
-2	4	3	5	0
-2	0	3	5	4
-2	0	3	5	4
-2	0	3	4	5

Sortiranje traženjem ekstrema

```
void selection_sort(int *niz, int n){
   int i, j, min, poz, temp;
   for(i = 0; i < n - 1; i++){
        poz = i;
        min = niz[i];
        for(j = i + 1; j < n; j++){
                if(min > niz[j]){
                        poz = j;
                        min = niz[j];
                }
                if(poz != i){
                        temp = niz[i];
                        niz[i] = niz[poz];
                        niz[poz] = temp;}
        }
   }
}
```

Sortiranje traženjem ekstrema (2)

```
.....
selection_sort(niz, n);
for(i = 0; i < n; i++) printf("niz[%d] = %d, ", i ,niz[i]);
.....
```

- Sljedeći standardni algoritam za sortiranje je tzv. Bubble sort (*bubble* = mjehurić).
- Ovaj algoritam baziran je na zamjenama susjednih elemenata niza ukoliko su oni u „lošem” poretku – „sortiranje zamjenama susjeda”.
- Ako prilikom prolaska niza nema zamjena onda je niz sortiran.

Primjer:

niz[0]	niz[1]	niz[2]	niz[3]	niz[4]
3	4	-2	5	0
3	4	-2	5	0
3	-2	4	5	0
3	-2	4	5	0
3	-2	4	0	5
-2	3	4	0	5
-2	3	4	0	5
-2	3	0	4	5
-2	3	0	4	5
-2	0	3	4	5

Bubble sort

```
void bubble_sort(int *niz, int n){
   int i, j, temp, zamjena;
    j = n - 1;
   do{
        zamjena = 0;
        for(i = 0; i < j; i++){
                if(niz[i] > niz[i + 1]){
                        temp = niz[i];
                        niz[i] = niz[i + 1];
                        niz[i + 1] = temp;
                        zamjena = 1;
                }
        }
        j--;
   }
   while(zamjena);
}
```

Sortiranje prema zadanim kriterijima

Primjer:

- Napišite dio programa koji učitava prirodni broj `n` te niz `niz` s `n+1` cijelim brojem. Niz treba sortirati silazno prema vrijednosti predzadnje znamenke.
- Niz ograničite na `200` elemenata.
- Obavezno navedite deklaracije svih korištenih varijabli.
- Nije dozvoljeno korištenje funkcija iz `<math.h>` i dodatnih nizova.

Sortiranje prema zadanim kriterijima (2)

```
int predzadnja(int broj){
   int p_z;
   p_z = (broj - broj/100 * 100)/10;
   if(p_z < 0) return -p_z; else return p_z;
}

int main( )
{
   int n, niz[200];
   int i, j, temp, zamjena;

   printf("ucitaj n:");
   scanf("%d", &n);
```

Sortiranje prema zadanim kriterijima (3)

```
/* ucitavanje niza */
for(i = 0; i <= n; i++) scanf("%d", &niz[i]);

/* bubble sort */
 j = n;
do{
        zamjena = 0;
        for(i = 0; i < j; i++){
                if(predzadnja(niz[i]) < predzadnja(niz[i + 1])){
                        temp = niz[i];
                        niz[i] = niz[i + 1];
                        niz[i + 1] = temp;
```

Sortiranje prema zadanim kriterijima (4)

```
                    zamjena =1 ;
                }
            }
            j--;
        }
        while(zamjena);
        for(i = 0; i <= n; i++)
            printf("niz[%d ] = %d, \n", i, niz[i]);
        return 0;
}
```