Pretraživanje i sortiranje polja

Ciljevi:

- upoznati se s pojmom pretraživanja polja
- proučiti metode pretraživanja polja
- upoznati se s pojmom sortiranja polja
- proučiti metode sortiranja polja

Pregled lekcije

Definicija pretraživanja polja

Pretraživanje polja je postupak traženja neke vrijednosti u skupu svih vrijednosti tj. elemenata polja. Pretraživanje polja je, uz sortiranje, jedan od najvažnijih algoritama u računalstvu uopće.

Općenito pretraživanje polja možemo definirati kao postupak sa sljedećim ulazom i izlazom:

ULAZ: Niz vrijednosti A_1, ..., A_N i vrijednost K
IZLAZ: Ako se K nalazi u polju, vratiti indeks prvog elementa polja koji sadrži vrijednost jednaku vrijednosti varijable K, a inače vratiti -1.

Postoji više načina pretraživanja no mi ćemo u ovoj lekciji obraditi samo ono najjednostavnije.

Najjednostavniji algoritam pretraživanja je slijedno pretraživanje. Kod slijednog pretraživanja elementi polja A se uspoređuju redom jedan za drugim sa traženom vrijednošću K. Postupak se nastavlja sve dok se ne pronađe tražena vrijednost ili dok se ne dođe do kraja svih elemenata polja bez da je tražena vrijednost nađena.

Slijedno pretraživanje polja

Sada kada smo se upoznali sa pojmom slijednog pretraživanja polja, prikazat ćemo uporabu ove metode na primjeru, no prvo ćemo dati općenite korake ovog algoritma.

Slijedno pretraživanje:

1. Učitaj N
2. Učitaj niz A
3. Učitaj K
4. I = 0
5. Nadjen = false
6. Sve dok je Nadjen = false i I <= N
7. Ako je A[I] == K onda nadjen = true
8. inače I = I+1

9. Ako je nadjen == true onda
10. vrati I
11. inače
12. vrati -1

Općeniti algoritam slijednog pretraživanja, prikazan u samom programskom kodu izgleda ovako:

```
#include <iostream>
using namespace std;
int main () {
     int N;
     float A[1000], K;
     do {
        cout << "N = ";
        cin >> N;
     } while (N < 1 || N > 1000);
     for (int i = 0; i < N; i++) {
         cout << "A[" << i << "] = ";
         cin >> A[i];
     }
     cout << "K = ";
     cin >> K;
     i = 0;
     bool nadjen = false;
     while (!nadjen && i <= N)
           if (A[i] == K) nadjen = true;
           else i++;
if (nadjen) cout << i;
     else cout << -1;
     return 0;
}
```

Da bismo prikazali ovaj algoritam na konkretnom primjeru postavit ćemo sljedeći zadatak. Neka je zadano polje veličine 10 elemenata. Korisnik unosi vrijednosti u polje. Vrijednosti mogu biti cijeli i decimalni brojevi. Nakon toga potrebno je ispisati cijelo polje i pronaći u polju element s vrijednosti K ukoliko postoji i ispisati njegov indeks. Ukoliko K ne postoji ispisuje se također odgovarajuća poruka.

Primjer 1:

ULAZ: Unijeti elemente polja A
IZLAZ: Naći indeks traženog elementa K

1. Učitaj elemente polja A
2. Ispiši polje A
3. Prođi kroz cijelo polje A
   Ukoliko je vrijednost K nađena ispiši njen indeks
   Ukoliko vrijednost K nije nađena ispiši poruku

Primjer 1

```
#include <iostream>
using namespace std;
int main () {
     int N = 0;
     float A[10], K;
     do {
        cout << "Unesite vrijednost = ";
        cin >> A[N];
        N++;
     } while (N < 10);
     for (int i = 0; i < 10; i++) {
         cout << "A[" << i << "] = " << A[i] << endl;
     }
     cout << "K = ";
     cin >> K;
     i = 0;
     bool nadjen = false;
     while (!nadjen && i < 10)
           if (A[i] == K) nadjen = true;
           else i++;
if (nadjen) cout << i << endl;
     else cout << "Trazena vrijednost ne postoji" << endl;
     system ("PAUSE");
     return 0;
}
```

U prethodnom primjeru smo deklarirali polje A veličine 10 elemenata te varijablu K decimalnog tipa podataka. Deklarirali smo također cjelobrojnu varijablu N koja će nam služiti kao brojač. U do...while iteraciji unosimo redom svih 10 elemenata polja. Nakon toga koristeći for iteraciju i brojač i koji ide od 0 do 9 (indeksi elemenata polja), ispisujemo redom sve elemente polja A. Nakon toga unosimo u varijablu K traženu vrijednost te pomoću while iteracije i if selekcije prolazimo kroz cijelo polje i tražimo vrijednost K. Ukoliko je vrijednost K nađena ispisuje se vrijednost varijable i, koju smo prije iteracije while postavili n 0 i koja predstavlja vrijednost indeksa tekućeg elementa. Ukoliko vrijednost nije nađena a vrijednost indeksa i dosegne 9 to znači da smo došli do kraja polja bez da smo pronašli traženu vrijednost elemenata pa se u skladu s tim ispisuje i odgovarajuća poruka.

Definicija sortiranja polja

Sortiranje polja predstavlja pretvrobu niza elemenata polja iz početnog oblika u oblik u kojem je niz poredan prema željenom kriteriju (uzlazno, silazno, abecedno, itd.). U tipičnom slučaju radi se o soritranju vrijednosti od najmanje do najveće vrijednosti.

Općenito soritranje polja možemo definirati kao postupak sa sljedećim ulazom i izlazom:

ULAZ: Niz vrijednosti A_1, ..., A_N
IZLAZ: Niz vrijednosti B_1, ..., B_N, koji se sastoji od vrijednosti koje se nalaze u nizu A_1, ..., A_N, ali poredanih od najmanje prema najvećoj.

Postoji više algoritama sortiranja no mi ćemo u ovoj lekciji objasniti 4 algoritma:

- *Sortiranje izborom (Selection sort)*
- *Sortiranje zamjenom (Exchange sort)*
- *Mjehuričasto sortiranje (Bubble sort)*
- *Sortiranje umetanjem (Insertion sort)*

Sortiranje izborom (Selection sort)

Algoritam sortiranja izborom je jedan od najjednostavnijih algoritama sortiranja. Algoritam funkcionira tako da se u polju pronađe element s najvećom vrijednosti i zatim se vrijednost tog elementa zamjeni s vrijednosti posljednjeg elementa u polju. Nakon toga se postupak ponavlja za sve elemente osim poljednjeg. Nakon toga se postupak ponavlja za sve elemente osim posljednja dva, itd. sve dok se ne dođe do samo jednog elementa. Tu se staje jer naravno jedan element više nema biti s čime uspoređen pa nikakva zamjena niti ne dolazi u obzir.

Sada kada smo se upoznali sa pojmom sortiranja izborom, prikazat ćemo uporabu ove metode na primjeru, no prvo ćemo dati općenite korake ovog algoritma.

Sortiranje izborom:

1. Učitaj N
2. Učitaj niz A
3. Za i = N..2
4. Max = 1
5. Za j = 1..i
6. Ako je A[j] >= A[Max] onda Max = j
7. pom = A[i-1]
8. A[i-1] = A[j]
9. A[j] = pom
10. Ispiši niz A

Općeniti algoritam sortiranja izborom, prikazan u samom programskom kodu izgleda ovako:

```
#include <iostream>
using namespace std;
int main () {
    int N;
    float A[1000];
    do {
       cout << "N = ";
       cin >> N;
    } while (N < 2 || N > 1000);
    for (int i = 0; i < N; i++) {
        cout << "A[" << i << "] = ";
        cin >> A[i];
    }
for (int i = N; i > 1; i--) {
        int Max = 0;
        for (int j = 1; j < i; j++)
```

```
            if (A[j] > A[Max]) Max  = j;
        float pom = A[i-1];
        A[i-1] = A[Max];
        A[Max] = pom;
    }
for (int i = 0; i < N; i++)
        cout << A[i] << " ";
    cout << endl;
    return 0;
}
```

Da bismo prikazali ovaj algoritam na konkretnom primjeru postavit ćemo sljedeći zadatak. Neka je zadano polje veličine 10 elemenata. Korisnik unosi vrijednosti u polje. Vrijednosti mogu biti cijeli i decimalni brojevi. Nakon toga potrebno je ispisati cijelo polje i sortirati ga uzlazno. Nakon sortiranja potrebno je ponovo ispisati sadržaj polja.

Primjer 2:

ULAZ: Unijeti elemente polja A

IZLAZ: Ispisati elemente polja A u uzlazno sortiranom obliku

1. Učitaj elemente polja A
2. Ispiši polje A
3. Za i = N..2
4. Max = 1
5. Za j = 1..i
6. Ako je A[j] >= A[Max] onda Max = j
7. pom = A[i-1]
8. A[i-1] = A[j]
9. A[j] = pom
10. Ispiši polje A

```
#include <iostream>
using namespace std;
int main () {
    int N = 0;
    float A[10];
    do {
       cout << "Unesite vrijednost = ";
       cin >> A[N];
       N++;
    } while (N < 10);
    for (int i = 0; i < 10; i++) {
        cout << "A[" << i << "] = " << A[i] << endl;
    }
for (int i = 10; i > 1; i--) {
        int Max = 0;
        for (int j = 1; j < i; j++)
            if (A[j] > A[Max]) Max  = j;
        float pom = A[i-1];
        A[i-1] = A[Max];
        A[Max] = pom;
    }
for (int i = 0; i < 10; i++)
```

```
        cout << A[i] << " ";
    cout << endl;
    system ("PAUSE");
    return 0;
}
```

U prethodnom primjeru smo deklarirali polje A veličine 10 elemenata. Nakon toga smo u do...while iteraciji unijeli sve elemente polja. Koristeći for iteraciju ispisali smo polje u originalnom obliku. Zatim smo koristeći 2 for iteracije (ugnježđene iteracije) izveli postupak sortiranja. Vanjska for iteracija ide od kraja polja i smanjuje se za 1 u svakom koraku je će unutarnja for iteracija u svakom koraku vanjske for iteracije pronaći najveći element u polju i staviti ga na kraj. Smanjenjem vrijednosti brojača vanjske for petlje za 1 u svakom koraku, smanjujemo broj elemenata polja koje će unutarnja for iteracija pretraživati u potrazi za najvećim elementom a ujedno ćemo pomicati prema naprijed poziciju na koju će biti smješten najveći element polja nakon što je pronađen. Prilikom svake zamjene koristimo pomoćnu varijablu pom. Nakon sortiranja polja isto ispisujemo koristeći opet for iteraciju.

Sortiranje zamjenom (Exchange sort)

Sortiranje zamjenom je također kao i sortiranje izborom jednostavniji algoritam sortiranja. U ovom algoritmu se vrijednost posljednjeg elementa uspoređuje sa vrijednostima svih prethodnih elemenata. Ukoliko se naiđe na element sa većom vrijednosti po posljednjeg, vrši se njihova zamjena itd. što će rezultirati time da će na kraju prvog prolaza na kraju polja biti element s najvećom vrijednosti. Cijeli postupak se zatim ponavlja za pretposljednji element i tako dalje, sve dok ne preostane samo jedan element.

Sada kada smo se upoznali sa pojmom sortiranja zamjenom, prikazat ćemo uporabu ove metode na primjeru, no prvo ćemo dati općenite korake ovog algoritma.

Sortiranje zamjenom:

1. Učitaj N
2. Učitaj niz A
3. Za i = N..2
4. Za j = 1..i
5. Ako je A[j] >= A[i] onda
6. pom = A[j]
7. A[j] = A[i]
8. A[i] = pom
9. Ispiši niz A

Općeniti algoritam sortiranja zamjenom, prikazan u samom programskom kodu izgleda ovako:

```
#include <iostream>
using namespace std;
int main () {
    int N;
    float A[1000];
    do {
       cout << "N = ";
```

```
        cin >> N;
    } while (N < 2 && N > 1000);
    for (int i = 0; i < N; i++) {
        cout << "A[" << i << "] = ";
        cin >> A[i];
    }
for (int i = N-1; i > 0; i--) {
        for (int j = 0; j < i; j++)
            if (A[j] > A[i]) {
                float pom = A[j];
                A[j] = A[i];
                A[i] = pom;
            }
    }
for (int i = 0; i < N; i++)
        cout << A[i] << " ";
    cout << endl;
    return 0;
}
```

Da bismo prikazali ovaj algoritam na konkretnom primjeru postavit ćemo sljedeći zadatak. Neka je zadano polje veličine 5 elemenata. Korisnik unosi vrijednosti u polje. Vrijednosti mogu biti cijeli i decimalni brojevi. Nakon toga potrebno je ispisati cijelo polje i sortirati ga uzlazno. Nakon sortiranja potrebno je ponovo ispisati sadržaj polja.

Primjer 3:

ULAZ: Unijeti elemente polja A

IZLAZ: Ispisati elemente polja A u uzlazno sortiranom obliku

11. Učitaj elemente polja A
1. Ispiši polje A
2. Za i = N..2
3. Za j = 1..i
4. Ako je A[j] >= A[i] onda
5. pom = A[j]
6. A[j] = A[i]
7. A[i] = pom
8. Ispiši polje A

```
#include <iostream>
using namespace std;
int main () {
    int N = 0;
    float A[5];
    do {
        cout << "Unesite vrijednost = ";
        cin >> A[N];
        N++;
    } while (N < 5);
    for (int i = 0; i < 5; i++) {
        cout << "A[" << i << "] = " << A[i] << endl;
    }
```

```
for (int i = 4; i > 0; i--) {
        for (int j = 0; j < i; j++)
            if (A[j] > A[i]) {
                float pom = A[j];
                A[j] = A[i];
                A[i] = pom;
            }
    }
for (int i = 0; i < 5; i++)
        cout << A[i] << " ";
    cout << endl;
    system ("PAUSE");
    return 0;
}
```

U prethodnom primjeru smo deklarirali polje A veličine 10 elemenata. Nakon toga smo u do...while iteraciji unijeli sve elemente polja. Koristeći for iteraciju ispisali smo polje u originalnom obliku. Zatim smo koristeći 2 for iteracije (ugnježđene iteracije) izveli postupak sortiranja. Vanjska for iteracija ide od kraja polja i smanjuje se za 1 u svakom koraku. Unutarnja for iteracija je zadužena za provjeru svih elemenata polja do elementa s indeksom vrijednosti brojača vanjske for iteracije i zamjene vrijednosti elemenata ukoliko je bilo koji element polja veći od zadnjeg elementa tj. elementa s indeksom vrijednosti brojača vanjske for iteracije. Prilikom svake zamjene koristimo pomoćnu varijablu pom. Nakon sortiranja polja isto ispisujemo koristeći opet for iteraciju.

Mjehuričasto sortiranje (Bubble sort)

Mjehuričasto sortiranje je nešto efikasnija varijanta sortiranja zamjenom. Ovaj algoritam sortiranja funkcionira tako da se svaki element polja uspoređuje sa elementom koji slijedi i ukoliko je sljedbenik tekućeg elementa veći vrši se zamjena njihovih vrijednosti. Kao i kod prethodnih sortiranja nakon prvog prolaska element s najvećom vrijednosti će biti na kraju polja tako da se u sljedećem prolasku zadnji element polja više ne mora gledati. Postupak sortiranja završava kada preostane samo jedan element polja ili kada u cijelom prolasku kroz polje nema potrebe za niti jednom zamjenom.

Sada kada smo se upoznali sa pojmom mjehuričastog sortiranja, prikazat ćemo uporabu ove metode na primjeru, no prvo ćemo dati općenite korake ovog algoritma.

Mjehuričasto sortiranje:

1. Učitaj N
2. Učitaj niz A
3. Za i = N..2
4. Za j = 1..i
5. Ako je A[j] >= A[j+1] onda
6. pom = A[j]
7. A[j] = A[j+1]
8. A[j+1] = pom
9. Ispiši niz A

Općeniti algoritam mjehuričastog sortiranja, prikazan u samom programskom kodu izgleda ovako:

```
#include <iostream>
using namespace std;
int main () {
    int N;
    float A[1000];
    do {
       cout << "N = ";
       cin >> N;
    } while (N < 2 && N > 1000);
    for (int i = 0; i < N; i++) {
        cout << "A[" << i << "] = ";
        cin >> A[i];
    }
bool zamjena = true;
    for (int i = N-1; i > 0 && zamjena; i--) {
        zamjena = false;
        for (int j = 0; j < i; j++)
            if (A[j] > A[j+1]) {
               float pom = A[j];
               A[j] = A[j+1];
               A[j+1] = pom;
               zamjena = true;
            }
    }
for (int i = 0; i < N; i++)
        cout << A[i] << " ";
    cout << endl;
    return 0;
}
```

Da bismo prikazali ovaj algoritam na konkretnom primjeru postavit ćemo sljedeći zadatak. Neka je zadano polje veličine 5 elemenata. Korisnik unosi vrijednosti u polje. Vrijednosti mogu biti cijeli i decimalni brojevi. Nakon toga potrebno je ispisati cijelo polje i sortirati ga uzlazno. Nakon sortiranja potrebno je ponovo ispisati sadržaj polja.

Primjer 4:

ULAZ: Unijeti elemente polja A

IZLAZ: Ispisati elemente polja A u uzlazno sortiranom obliku

1. Učitaj elemente polja A
2. Ispiši polje A
3. Za i = N..2
4. Za j = 1..i
5. Ako je A[j] >= A[j+1] onda
6. pom = A[j]
7. A[j] = A[j+1]
8. A[j+1] = pom
9. Ispiši polje A

```
#include <iostream>
using namespace std;
int main () {
    int N = 0;
    float A[5];
    do {
       cout << "Unesite vrijednost = ";
       cin >> A[N];
       N++;
    } while (N < 5);
    for (int i = 0; i < 5; i++) {
        cout << "A[" << i << "] = " << A[i] << endl;
    }
bool zamjena = true;
    for (int i = 4; i > 0 && zamjena; i--) {
        zamjena = false;
        for (int j = 0; j < i; j++)
            if (A[j] > A[j+1]) {
               float pom = A[j];
               A[j] = A[j+1];
               A[j+1] = pom;
               zamjena = true;
            }
    }
for (int i = 0; i < 5; i++)
        cout << A[i] << " ";
    cout << endl;
    system ("PAUSE");
    return 0;
}
```

Jedino što u prethodnom primjeru možda nije jasno ili analogno prethodnim objašnjenima algoritama je da u vanjskoj for iteraciji prilikom sortiranja koristimo bool varijablu zamjena. Dakle, unutarnja for iteracija će varijablu zamjena postaviti na true samo ukoliko su zadovoljeni uvjeti za zamjenom, dakle ukoliko je tekući element veći od svog sljedbenika.

Sortiranje umetanjem (Insertion sort)

Ovo sortiranje je nešto složenije od prethodnih. Polje se kod ovog sortiranja dijeli na dva dijela: sortirani i nesortirani dio. Na početku se sortirani dio polja sastoji samo od prvog elementa polja. U svakom sljedećem koraku sortirani dio se smanjuje za jedan element a nesortirani povećava za jedan element prebacivanjem prvog elementa iz nesortiranog u sortirani dio polja. Prebacivanje se vrši tako da se promatrani element iz nesortiranog dijela uspoređuje s posljednjim elementom u soriranom dijelu, te ako je promatrani element manji, posljednji se element iz sortiranog dijela prebacuje za jedno mjesto dalje u polju. Nakon toga se promatrani element uspoređuje s pretposljednjim elementom u polju itd., sve dok se ne naiđe na element u sortiranom dijelu polja koji je manji od promatranog elementa ili dok se i prvi element sortiranog dijela polja ne pomakne za jedno mjesto dalje u polju. Na taj se način oslobodilo mjesto u polju na koje se umeće novi element i povećava sortirani dio polja za jedan element.

Sada kada smo se upoznali sa pojmom sortiranja umetanjem, prikazat ćemo uporabu ove metode na primjeru, no prvo ćemo dati općenite korake ovog algoritma.

Sortiranje umetanjem:

1. Učitaj N
2. Učitaj niz A
3. Za i = 1..N-1
4. j = i-1
5. pom = A[i]
6. Sve dok je j>=0 i A[j] > pom radi
7. A[j+1] = A[j]
8. j = j -1
9. A[j+1] = pom
10. Ispiši niz A

Općeniti algoritam sortiranja umetanjem, prikazan u samom programskom kodu izgleda ovako:

```
#include <iostream>
using namespace std;
int main () {
    int N;
    float A[1000];
    do {
       cout << "N = ";
       cin >> N;
    } while (N < 2 && N > 1000);
    for (int i = 0; i < N; i++) {
        cout << "A[" << i << "] = ";
        cin >> A[i];
    }
for (int i = 1; i < N; i++) {
        int j = i-1;
        float pom = A[i];
        while (j >= 0 && A[j] > pom)
              A[j+1] = A[j--];
        A[j+1] = pom;
    }
for (int i = 0; i < N; i++)
        cout << A[i] << " ";
    cout << endl;
     return 0;
}
```

Da bismo prikazali ovaj algoritam na konkretnom primjeru postavit ćemo sljedeći zadatak. Neka je zadano polje veličine 5 elemenata. Korisnik unosi vrijednosti u polje. Vrijednosti mogu biti cijeli i decimalni brojevi. Nakon toga potrebno je ispisati cijelo polje i sortirati ga uzlazno. Nakon sortiranja potrebno je ponovo ispisati sadržaj polja.

Primjer 5:

ULAZ: Unijeti elemente polja A

IZLAZ: Ispisati elemente polja A u uzlazno sortiranom obliku

10. Učitaj elemente polja A
11. Ispiši polje A
12. Za i = 1..N-1
13. j = i-1
14. pom = A[i]
15. Sve dok je j>=0 i A[j] > pom radi
16. A[j+1] = A[j]
17. j = j -1
18. A[j+1] = pom
19. Ispiši polje A

```
#include <iostream>
using namespace std;
int main () {
    int N = 0;
    float A[5];
    do {
       cout << "Unesite vrijednost = ";
       cin >> A[N];
       N++;
    } while (N < 5);
    for (int i = 0; i < 5; i++) {
        cout << "A[" << i << "] = " << A[i] << endl;
    }
for (int i = 1; i < 5; i++) {
        int j = i-1;
        float pom = A[i];
        while (j >= 0 && A[j] > pom)
              A[j+1] = A[j--];
        A[j+1] = pom;
    }
for (int i = 0; i < 5; i++)
        cout << A[i] << " ";
    cout << endl;
    system ("PAUSE");
    return 0;
}
```

Ono što možemo primijetiti u ovom primjeru je da za razliku od ostalih koristimo for iteraciju i unutar nje while iteraciju da bismo mogli realizirati logiku sortiranja iz opisa algoritma.

Sortiranje umetanjem je najefikasnije od svih algoritama koje smo obradili u ovoj lekciji. Osim toga, ovo sortiranje ima još jedno važno svojstvo koje drugi ovdje opisani aloritmi nemaju. Kada u polje koje je već sortirano treba umetnuti još jedan element, ali tako da polje ostane i dalje sortirano, može se koristiti jedan korak sortiranja umetanjem da se to učini.

Pogledajmo sljedeći zadatak.

Korisnik upisuje prirodni broj *N* i *N* decimalnih brojeva. Program treba nakon svakog upisa ispisati sve upisane decimalne brojeve, ali poredane po veličini.

Pseudokod za rješenje ovog problema izgledao bi ovako:

Primjer 6:
ULAZ: Prirodni broj *N* i *N* decimalnih brojeva
IZLAZ: Sortirani svi unešeni nizovi decimalnih brojeva

1. Unesi N
2. Za i = 1.. N radi
3. Unesi B
4. j = i
5. Sve dok je j >= 0 i A[i] > B
6. A[i+1] = A[i]
7. Ispiši polje A

Odnosno, u C++ programu

```
#include <iostream>
using namespace std;

int main () {
    int N;
    float A[1000], B;
    do {
        cout << "N = ";
        cin >> N;
    } while (N < 2 || N > 1000);
    for (int i = 0; i < N ; i++) {
        int j = i-1;
        cout << "A[" << i << "] = ";
        cin >> B;
        while (j >= 0 && A[j] > B)
              A[j+1] = A[j--];
        A[j+1] = B;
        for (int k = 0; k <= i; k++)
            cout << A[k] << "  ";
        cout << endl;
    }
    system("pause");
    return 0;
}
```

Zadaci za vježbu

1. Odgovorite na pitanje što je to pretraživanje.

2. Odgovorite na pitanje koja sortiranja poznajete.

3. Koje od sortiranja koje poznajete je najsloženije?

4. Izradite program u kojem se od korisnika traži unos 10 decimalnih brojeva koji se zatim sortiraju uzlazno i ispisuju na ekran. Koristite sortiranje po izboru.

Programski primjeri za laboratorijske vježbe

Zadatak 1.

Izradite program u kojem:

1. Se unosi 5 decimalnih brojeva u polje. Polje treba ispisati u originalnom obliku, zatim ga sortirati uzlazno koristeći sortiranje izborom i ispisati u sortiranom obliku na ekran. Polje treba također ispisati i obrnutim redoslijednom (od zadnjem elementa do prvog).

2. Korisnik unosi 10 brojeva u polje (mogu biti cijeli i decimalni). Potrebno je ispisati najmanji i najveći element polja (koristeći mjehuričasto sortiranje, ne koristiti pretraživanje) te prosječnu vrijednost svih elemenata polja. Potrebno je zatim zamjeniti mjesta u polju tako da prvih 5 elemenata bude zadnjih 5 a zadnjih 5 da bude prvih 5. Nakon toga je potrebno polje ispisati na ekran.

3. Korisnik unosi 10 različitih slova u polje. Ukoliko korisnik unese 2 puta isto slovo potrebno je unos poništiti i zatražiti ponovni unos slova. Nakon unosa svih slova, potrebno je ispisati slovo čiji ASCII kod je sedmi po veličini u odnosu na ASCII kodove svih slova u polju. Korisnik potom unosi jedno slovo a program mu ispisuje da li traženo slovo postoji i ako postoji na kojoj je poziciji u polju te koji mu je ASCII kod.

4. Korisnik unosi 10 slova (može biti i više istih). Potrebno je sortirati polje koristeći sortiranje umetanjem u silaznom obliku i ispisati polje na ekran. Zatim je potrebno ispisati broj pojavljivanja svakog pojedinog različitog slova u polju. Potrebno je zatim ispisati pozicije svakog suglasnika u polju.

5. Korisnik unosi 5 riječi u polje. Potrebno je riječi sortirati uzlazno te ispisati na ekran. Zatim je riječi potrebno sortirati silazno i ispisati na ekran. Korisnik zatim unosi jedno slovo a program mu ispisuje koliko se puta pojedino slovo javlja na svakoj od pozicija u polju tj. u svakoj pojedinoj riječi. Zatim je potrebno ispisati na ekran riječ u polju koja sadrži najviše i riječ koja sadrži najmanje samoglasnika.