3. Pretraživanje podataka

3.1. SEKVENCIJALNO PRETRAŽIVANJE

Pretraživanje je postupak traženja određenog podatka iz nekog skupa podataka, a u ovoj vježbi pretraživat će se polje cijelih brojeva. Sekvencijalno (slijedno, linearno) pretraživanje je postupak kada ispitujemo svaki element polja (po redu) od prvog do zadnjega ili dok se ne pronađe traženi podatak.

Primjer:

Imamo zadan niz od 5 cijelih brojeva te broj **a** koji se traži unutar niza.

niz[5]= {-2, 4, 1, 5, 3}, 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

Kod:

```
.....
pronasli=0; i=0;
....
while(!pronasli && i<n)
    if(niz[i]==a) pronasli=1; else i++;
if(pronasli) printf("Element je pronadjen.\n");
else printf("Element nije pronadjen.\n");
.....
```

3.2. BINARNO PRETRAŽIVANJE

Moć binarnog pretraživanja dolazi do punog izražaja u sortiranom polju. To je najbrži način pretraživanja polja, osobito bitan za polja s velikim brojem članova. Jednostavan primjer logike binarnog pretraživanja je dobro poznata igra pogađanja zamišljenog broja. U toj igri zapiše se neki broj npr. između 1 i 100. Da bi se pronašao zapisani (ili zamišljeni) broj potrebno je redom postavljati pitanja. Mora se znati jedino u kojem se intervalu nalazi zamišljeni broj (taj interval je [1,100]). Prvo je razumno podijeliti interval na polovicu i upitati je li broj veći od 50. Ako je odgovor da, onda je zapisani broj između 50 i 100. Ponovno se interval [50, 100] dijeli na polovcu (polovica je 75) i postavlja isto pitanje je li broj veći od 75. Svaki odgovor dopušta podjelu intervala u kojem se nalazi zapisani broj na polovicu. Interval se smanjuje i na koncu interval postaje samo jedan broj, a on je zapisani ili zamišljeni broj.

Primjer:

Zapisuje se 44. Kako binarnim pretraživanjem pronaći zapisani broj (44) prikazuje sljedeća tablica:

Broj koraka	Traženi broj	Rezultat	Interval mogućih vrijednosti
0			1 – 100
1	50	Broj je velik	1 – 49
2	25	Broj je mali	26 – 49
3	37	Broj je mali	38 – 49
4	43	Broj je mali	44 – 49
5	46	Broj je velik	44 – 45
6	44	Točno	

Tablica 1. Primjer binarnog pretraživanja

Zapisani broj (44) pronalazi se u šest koraka. Broj koraka može biti i manji ako bi se zapisao drugi broj, npr. 50. Tada je dovoljan samo jedan korak. Broj koraka može biti i veći, npr. broj 45 se pronalazi u 7 koraka.

Linearno pretraživanje bi značilo postavljati redom pitanja je li zapisani broj 1, 2, 3, i tako redom do 100. Prosječan broj koraka za linearno pretraživanje bi bio 50.

Dakle, preduvjet za rad algoritma za binarno pretraživanje je da su podaci koji se pretražuju (polje cijelih brojeva) POREDANO (SORTIRANO). Algoritam radi tako da se odrede donja i gornja granica pretraživanja. U prvom koraku je donja granica ***dg = 0*** i gornja granica

gg = N – 1, gdje je N broj podataka u polju. Zatim se računa indeks sredine intervala koji se pretražuje na slijedeći način: ***s = (dg + gg) / 2***. Promatramo element polja koji se nalazi na ***s***-tom mjestu, odnosno ***V[s]***. Sada mogu nastupiti tri slučaja:

1. ***traženi broj > V[s]*** – u ovom slučaju vrijednost donje granice intervala koji se pretražuje se pomiče na slijedeći način: ***dg = s + 1***, i pretraživanje se nastavlja,
2. ***traženi broj < V[s]*** – u ovom slučaju vrijednost gornje granice intervala koji se pretražuje se pomiče na slijedeći način: ***gg = s - 1***, i pretraživanje se nastavlja,
3. ***traženi broj = V[s]*** – pronađen je traženi broj i pretraživanje se prekida.

Pretraživanje se prekida i ako je zadovoljen uvjet ***dg = gg***.

Pseudokod:

učitaj broj **N** i polje **V[]**
poredaj polje **V** uzlazno
učitavaj brojeve **X** i za svaki učitani **X** radi:
    dg = 0, **gg** = N – 1
    dok (**dg** < **gg**) radi:
        s = (**dg** + **gg**) / 2
        ako (**V**[**s**] > **X**) onda: **gg** = **s -** 1
        ako (**V**[**s**] < **X**) onda: **dg** = **s** + 1
        inače: ispiši: Broj je pronadjen!
            Prekid petlje Dok

PODSJETNIK!

Sortiranje polja se vrši uzlazno i silazno. Uzlazno sortiranje preuređuje polje tako da elementi budu poredani u rastućem redoslijedu od najmanjeg prema najvećem dok je silazno sortiranje suprotno od navedenog. Postoji mnogo metoda sortiranja (dubble, quick, shell sort i dr.) i svaka od tih metoda vrši zamjenu elemenata u skladu s traženim poretkom, čime se mijenja položaj pojedinih elemenata u polju.

Najjednostavnija metoda je **Bubble sort** (uzlazno):

```
int sort;
do
{
	sort = 0;
	for (i=0; i<n-1;i++){ //za svaki element x[i] provjerava se da li je veći
		if (x[i] > x[i+1]) // od slijedećeg elementa
		{ //ukoliko je veći njegova vrijednost se sprema u
			float temp; // temp varijablu te mu se pridružuje vrijednost
			temp = x[i]; // slijedeće varijable, dok se slijedećoj varijabli
			x[i]=x[i+1]; // prodružuje vrijednost varijable temp odnosno
			x[i+1]=temp; // prethodne varijable
			sort = 1; //sort =1 sve dok se rade izmjene. Ukoliko nije
		//napravljena niti jedna izmjena sort =0 i do-while petlja se završava
		}
      }
} while (sort ==1);
```

Sortiranje niza pomoćnom varijablom (uzlazno):

Učitani niz podataka ako želimo ispisati sortiran uzlazno (od najmanjeg ka najvećem) koristimo sljedeće:

- pronađemo najmanji element niza
- smjestimo ga na prvo mjesto x[0]
- zatim tražimo najmanji od preostalih elemenata i smještamo ga na mjesto x[1]
- …

```
for (i=0; i<n-1; i++){    //n je broj elemenata u nizu vrijednosti
	for (j=i+1; j<n; j++){
		if ( x[i]>x[j] ) {  //zamijeni vrijednosti
			p=x[i];
			x[i]=x[j];
			x[j]=p;
		}
	}
}
```

Zadatak za vježbu:

1. Napisati C program koji generira ***500*** slučajnih brojeva u rasponu 0 – 1000. Učitati boj ***N*** s tipkovnice koji također mora biti veći od 0 i manji od 1000 te provjeriti sekvencijalnim pretraživanjem nalazi li se taj broj u ranije generiranoj listi. Ispisati da li je pronađen broj ***N*** te ispisati broj pretraživanja.

Pojašnjenje generiranja slučajnih brojeva pomoću funkcije **rand**:

int rand (void);

Generira slučajni broj – vraća vrijednost nasumično odabranog cijelog broja u rasponu od e 0 to RAND_MAX. Ovaj broj se generira algoritmom koji vraća niz nepovezanih brojeva svaki puta kada pozovemo ovu funkciju. Ovaj algoritam koristi izvor kako bi generirao ovaj niz, te se na početku treba inicijalizirati pomoću **srand**.

RAND_MAX je konstanta definirana u <stdlib>. Njena vrijednost može varirati između različitih implementacija, ali je najmanja garantirana vrijednost 32767.

Princip generiranja slučajnih brojeva u određenom rasponu vrijednosti je definiranje tog raspona korištenjem **rand** i modul %:

(vrijednost % 100) je raspon vrijednosti od 0 do 99

(vrijednost % 100 + 1) je raspon vrijednosti od 1 do 100

(vrijednost % 30 + 1985) je raspon vrijednosti od 1985 do 2014

Primjer:

```c
/* rand primjer: POGODI BROJ */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main ()
{
  int tajna, pogodi;
  /* inicijalizacija slučajnog izvora: */
  srand ( time(NULL) );
  /* generiraj slučajni broj: */
  tajna = rand() % 10 + 1;
  do {
    printf ("\nPogodite broj (1 d 10): ");
    scanf ("%d",&pogodi);
    if (tajna < pogodi) printf ("\nTajni broj je manji");
    else if (tajna > pogodi) printf ("\nTajni broj je veci");
  } while (pogodi!= tajna);

  printf ("\nBravo!");
  return 0;
}
```

Zadatak za zadaću

2. Napisati C program koji s tipkovnice učitava broj ponavljanja ***P***. Zatim se ***P*** puta generira slučajni broj u rasponu 0 – 20. Generirati ***P*** puta slučajni broj ***br*** koji se treba pronaći u nizu koristeći obje metode pretraživanja: sekvencijalno i binarno. Na kraju je potrebno izračunati i ispisati srednji broj pretraživanja za svaku metodu pojedinačno (pri izračunu srednjeg broja pretraživanja u račun uzeti samo ona pretraživanja koja su bila pozitivna).

   Ispis na ekranu:

   Unesite broj ponavljanja:

   Generirani niz: * * * *….

   Generirani brojevi: * * * * *…

   Srednja vrijednost pretrazivanja:

   Sekvencijalno - **

   Binarno - **