TABELE DE DISPERSIE

În lucrare sunt prezentate principalele operații asupra unei tabele de dispersie: construirea tabelei de dispersie, inserarea unei înregistrări, căutarea unei înregistrări, afişarea înregistrărilor. De asemenea se fac câteva considerații asupra alegerii funcției de dispersie.

1. Considerații teoretice

1.1.Tipuri de tabele

Tabelul este o colecție de elemente de acelaşi tip, identificabile prin chei. Elementele sale se mai numesc înregistrări.

Tabelele pot fi :

- **fixe**, cu un număr de înregistrări cunoscut dinainte (in momentul crearii);
- **dinamice,** cu un numar variabil de elemente.

Tabelele dinamice pot fi organizate sub formă de:

- listă dinamică simplu sau dublu înlănțuită;
- arbore de căutare;
- tabele de dispersie.

Din categoria **tabelelor fixe** face parte tabelul de cuvinte rezervate dintr-un limbaj de programare. Acesta este organizat ca un tablou de pointeri spre cuvintele rezervate, introduse în ordine alfabetică. Căutarea utilizată este cea binară.

Tabelele dinamice organizate sub formă de liste au dezavantajul căutării liniare. Arborele de căutare reduce timpul de căutare. În cazul în care cheile sunt alfanumerice, comparațiile sunt mari consumatoare de timp. Pentru astfel de situații, cele mai potrivite sunt tabelele de dispersie.

1.2.Funcția de dispersie (hashing)

Funcția de dispersie este funcția care transformă o cheie într-un număr natural numit cod de dispersie:

f: K -> H

unde K este mulțimea cheilor, iar H este o mulțime de numere naturale.

Funcția **f** nu este injectivă .Două chei pentru care $\mathbf{f(k_1)=f(k_2)}$ se spune că intră în coliziune, iar înregistrările respective se numesc sinonime.

Asupra lui **f** se pun două condiții:

- valoarea ei pentru un $k \varepsilon K$ să rezulte cât mai simplu şi rapid;

- să minimizeze numărul de coliziuni.

Un exemplu de funcție de dispersie este următoarea:

f(k)=γ(k) mod M

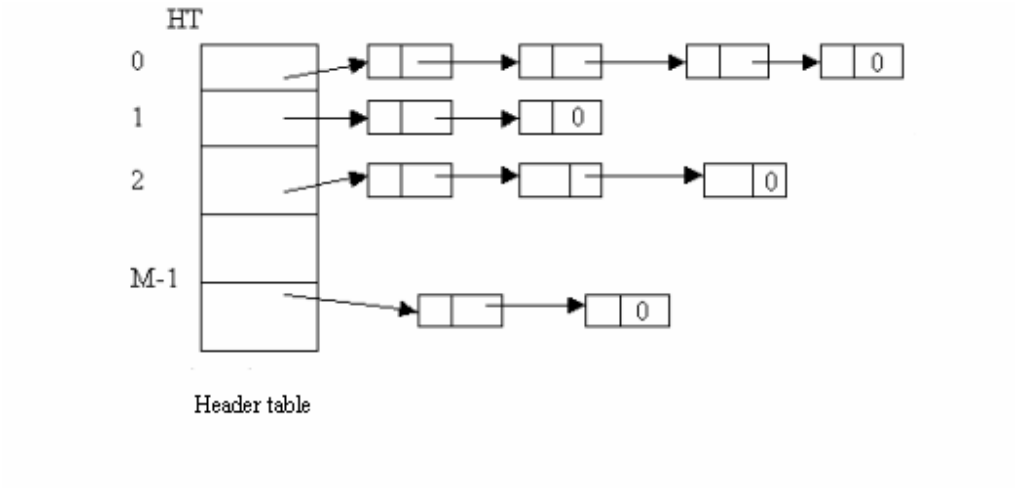
unde *γ(k)* este o funcție care transformă cheia într-un număr natural, iar M este un număr natural recomandat a fi prim.

Funcția **γ(k***)* se alege în funcție de natura cheilor. Dacă ele sunt numerice, atunci **γ(k)=k**. În cazul cheilor alfanumerice, cea mai simplă funcție **γ(k***)* este suma codurilor ASCII ale caracterelor din componența lor; ca urmare funcția **f** de calcul a dispersiei este următoarea:

```
#define M…
int f(char *key)
{
 int i,suma;
 suma=0;
 for(i=0;i<length(key);i++)
        suma=suma+*(key+i);
 return suma%M;
}
```

1.3.Tabela de dispersie

Rezolvarea coliziunilor se face astfel: toate înregistrările pentru care cheile intră în coliziune sunt inserate într-o listă simplu înlănțuită. Vor exista astfel mai multe liste, fiecare conținând înregistrări cu același cod de dispersie. Pointerii spre primul element din fiecare listă se păstrează într-un tablou, la indexul egal cu codul de dispersie .Ca urmare modelul unei tabele de dispersie este următorul:



Un nod al listei are structura următoare:

```
typedef struct tip_nod {
                        char *cheie;
                        // alte informatii utile
                        struct tip_nod *urm;
                       }TIP_NOD;
```

Tabloul HT (header table) este declarat astfel:

TIP_NOD *HT[M];

Inițial el conține pointerii nuli:

```
for(i=0;i<M;i++)
        HT[i]=0;
```

Căutarea într-o tabelă de dispersie a unei înregistrări având pointerul **key** la cheia sa, se face astfel:

- se calculează codul de dispersie:

```
h=f(key);
```

- se caută înregistrarea având pointerul **key** la cheia sa, din lista având pointerul spre primul nod HT[h].

Căutarea este liniară:

```
p=HT(h);
while(p!=0)
 {
        if(strcmp(key,p->cheie)==0) return p;
        p=p->urm;
 }
return 0;
```

Inserarea unei înregistrări într-o tabelă de dispersie se face astfel:

- se construieşte nodul de pointer **p,** care va conține informația utilă şi pointerul la cheia înregistrării:

```
p=new TIP_NOD;
citire_nod(p);
```

- se determină codul de dispersie al înregistrării:

```
h=f(p->cheie);
```

- dacă este prima înregistrare cu codul respectiv, adresa sa este depusă în tabelul HT:

```
if(HT[h]==0)
{
        HT[h]=p;
        p->urm=0;
}
```

în caz contrar se verifică dacă nu cumva mai există o înregistrare cu cheia respectivă. În caz afirmativ se face o prelucrare a înregistrării existente (ştergere, actualizare) sau este o eroare (cheie dublă). Dacă nu există o înregistrare de cheia respectivă, se inserează în listă ca prim element nodul de adresă **p:**

```
q=cautare(p->cheie);
if(q==0){  /* nu exista o înregistrare de cheia respectiva */
        p->urm=HT[h];
        HT[h]=p;
      }
else prelucrare(p,q);/* cheie dubla */
```

Construirea tabelei de dispersie se face prin inserarea repetată a nodurilor.

1.4.Listarea tuturor înregistrărilor

Listarea tuturor înregistrărilor pe coduri se face simplu, conform algoritmului următor:

```
for(i=0;i<M;i++){
        if(HT[i]!=0){
                printf("\nInregistrări avand codul de dispersie=%d\n",i);
                p=HT[i];
                while(p!=0){
                   afisare(p);
                   p=p->urm;
                }
        }
}
```

2. Tema

.

3.1. Să se implementeze algoritmii prezentați aferenți unei tabele de dispersie. Înregistrarea va conține datele aferente unui student. Cheia va fi numele şi prenumele studentului. Scrieți în plus față de cele prezentate o funcție de ștergere a unei înregistrări de cheie dată.

3.2. Scrieți un program care să tipărească inregistrarile dintr-o tabelă de dispersie în ordine alfabetică.

3.2. Să se afişeze frecvența de apariție a literelor dintr-un text utilizând o tabelă de dispersie.

Intrarea va fi textul

Ieşirea va fi următoarea:

Aa 11
Bb 12
Cc 1
….
Zz 4