# Structuri de Date

## Tabele de dispersie

Tudor Berariu (2016), Dragoș Corlătescu (2017)

## 1. Introducere

Scopul acestui laborator îl reprezintă implementarea unei tabele de dispersie ce va fi folosită ca un dicționar pentru contorizarea aparițiilor cuvintelor într-un text.

## 2. Tabela de dispersie

O tabelă de dispersie este o structură de date ce permite implementarea eficientă a unei asocieri între chei și valori. Pentru o tabelă de dispersie, operațiile de căutare, inserare și ștergere ale unei chei se execută, în general, în timp constant.

Pentru a defini o tabelă de dispersie trebuie definite tipul cheilor și tipul valorilor. Pentru laboratorul de astăzi cheile vor fi șiruri de caractere, iar valorile vor fi numere întregi (**int**).

Tipul unei funcții de *hashing* este următorul: aceasta primește o cheie și dimensiunea tabelei de dispersie și întoarce poziția din tabelă la care trebuie căutată sau introdusă acea cheie.

```
1  typedef char* Key;
2  typedef int Value;
3  typedef long(*HashFunction)(char*, long);
```

O intrare din tabela de dispersie va fi deci o listă de *sinonime (*chei aflate în *coliziune*), adică o listă de perechi cheie-valoare.

```
1  typedef struct Element {
2    Key key;
3    Value value;
4    struct Element *next;
5  } Element;
```

Definiția unei tabele de dispersie va fi, deci:

```
typedef struct HashTable {
  Element** elements;
  long size;
  HashFunction hashFunction;
} HashTable;
```

## 3. Cerințe

### Cerința 1.

Să se implementeze funcțiile:

**initHashTable -** care crează o tabelă de dispersie de dimensiune **size** și care folosește funcția **f** pentru *hashing.*

**exists** – care verifică dacă o cheie se află în tabela de dispersie.

**getValue** – care întoarce valoarea asociată unei chei din tabela de dispersie.

**put** – care asociază o valoare unei chei în tabela de dispersie. În cazul în care cheia există deja în tabelă, valoarea asociată este suprascrisă, altfel se adaugă o nouă intrare în tabelă.

**deleteKey** – care șterge o intrare din tabela de dispersie.

**print** – care afișează tabela de dispersie. Pentru fiecare cheie trebuie afișată lista de coliziuni.

**freeHashTable** – care eliberează complet memoria alocată pentru tabela de dispersie.

### Cerința 2.

Să se implementeze o funcție de *hashing* pentru șiruri de caractere pentru a fi folosită în tabela de dispersie. O astfel de funcție va întoarce un număr ce reprezintă o adresă în cadrul tabelei.

Un exemplu simplu de astfel de funcție este următorul:

**Date de intrare:** un șir de caractere ***s***, dimensiunea tabelei ***l***

**Date de ieșire:** adresa din tabelă corespunzătoare lui ***s***

```
h = 0;
for i = 0 to strlen(s) – 1 do
        h = h * 17 + s[i];
end
return h % l;
```

## Cerința 3.

Programul primește 3 argumente în linia de comandă:

1. dimensiunea tabelei de dispersie;

2. fișierul A;

3. fișierul B;

Să se construiască o tabelă de dispersie de dimensiunea indicată în care să se introducă toate cuvintele din fișierul A cu numărul de apariții asociat.

Creați o a doua tabelă în care repetați pentru fișierul B ce ați făcut pentru fișierul A.

Identificați numărul de cuvinte comune din cele două fișiere (dacă un cuvânt apare de mai multe ori în cele două fișiere, numărul comun de apariții este date de minimul dintre numărul de apariții din fișierul A și numărul de apariții din fișierul B).