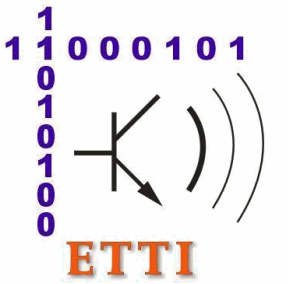




UNIVERSITATEA  
POLITEHNICA  
DIN BUCUREȘTI



# Hash tables

---

## Liste dublu înlanțuite

# Cuprins

## Partea I – Noțiuni teoretice

1. Hash tables
2. Liste dublu înlanțuite



## Partea II – Aplicații Laborator

Rezolvare aplicații Moodle



## 1. Hash table

Hash table (sau hash map) este o structură ce implementează o funcție asociativă, adică perechi de tipul key-value, prin intermediul unui tablou de tip array. Accesul la elemente se face, în general, mult mai rapid decât în cazul listelor înlănțuite și nu ține cont de dimensiunea datelor.

Exemplu: căutarea unei persoane într-o bază de date după nume.

key = nume

value = identitate

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.  
key = nume  
value = identitate

get ("Andy")

Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.

key = nume

value = identitate

get ("Andy")

Caz 1: liste simplu înlănțuite

nume == "Andy"? ❌



Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.  
key = nume  
value = identitate

get ("Andy")

## Caz 1: liste simplu înlănțuite

nume == "Andy"? ❌



Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.  
key = nume  
value = identitate

get ("Andy")

## Caz 1: liste simplu înlănțuite

nume == "Andy"? ❌



Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.  
key = nume  
value = identitate

get ("Andy")

Caz 1: liste simplu înlănțuite

nume == "Andy"? ❌



Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6



# 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.  
key = nume  
value = identitate

get ("Andy")

Caz 1: liste simplu înlănțuite

nume == "Andy"? ✓



Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

## 1. Hash table

```
struct id{
    int varsta;
    float greutate;
    int inaltime;
};

struct persoana{
    char * nume; // key
    struct id identitate; // value
};
```

Exemplu: căutarea unei persoane într-o bază de date după nume.

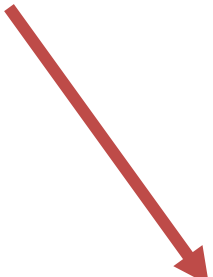
key = nume

value = identitate

$h()$  = funcție de hashing; transformă conținutul unui câmp de dimensiune arbitrară ( $key$ ) într-o valoare întreagă (index)

$get("Andy") = array[h("Andy")] = array[4]$

### Caz 2: hash tables



array	Sue	Carl	Bob	Charlie	Andy	Heather	Sena
	12	22	45	16	31	72	66
	45,5	81,9	73	67,8	91,4	61,5	66
	1,64	1,77	1,88	1,74	1,93	1,56	1,66
	0	1	2	3	4	5	6

## 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

array

0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4

array

				Andy 31 91,4 1,93		
0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4

Bob => B (66) + o (111) + b (98) => 275 % 7 = 2

array

		Bob 45 73 1,88		Andy 31 91,4 1,93		
0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4

Bob => B (66) + o (111) + b (98) => 275 % 7 = 2

Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1

array

	Carl 22 81,9 1,77	Bob 45 73 1,88		Andy 31 91,4 1,93		
0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4  
Bob => B (66) + o (111) + b (98) => 275 % 7 = 2  
Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1  
Charlie => C (67) + h (104) + a (97) + r (114) + l (108) + i (105) + e (101) => 696 % 7 = 3

array

	Carl 22 81,9 1,77	Bob 45 73 1,88	Charlie 16 67,8 1,74	Andy 31 91,4 1,93		
0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4  
Bob => B (66) + o (111) + b (98) => 275 % 7 = 2  
Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1  
Charlie => C (67) + h (104) + a (97) + r (114) + l (108) + i (105) + e (101) => 696 % 7 = 3  
Heather => H (72) + e (101) + a (97) + t (116) + h (104) + e (101) + r (114) => 705 % 7 = 5

array

	Carl	Bob	Charlie	Andy	Heather	
	22	45	16	31	72	
	81,9	73	67,8	91,4	61,5	
	1,77	1,88	1,74	1,93	1,56	
0	1	2	3	4	5	6



# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4  
Bob => B (66) + o (111) + b (98) => 275 % 7 = 2  
Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1  
Charlie => C (67) + h (104) + a (97) + r (114) + l (108) + i (105) + e (101) => 696 % 7 = 3  
Heather => H (72) + e (101) + a (97) + t (116) + h (104) + e (101) + r (114) => 705 % 7 = 5  
Sena => S (83) + e (101) + n (110) + a (97) => 391 % 7 = 6

array

	Carl	Bob	Charlie	Andy	Heather	Sena
	22	45	16	31	72	66
	81,9	73	67,8	91,4	61,5	66
	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4  
Bob => B (66) + o (111) + b (98) => 275 % 7 = 2  
Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1  
Charlie => C (67) + h (104) + a (97) + r (114) + l (108) + i (105) + e (101) => 696 % 7 = 3  
Heather => H (72) + e (101) + a (97) + t (116) + h (104) + e (101) + r (114) => 705 % 7 = 5  
Sena => S (83) + e (101) + n (110) + a (97) => 391 % 7 = 6  
Sue => S (83)+ u (117) + e (101) => 301 % 7 = 0

array

Sue	Carl	Bob	Charlie	Andy	Heather	Sena
12	22	45	16	31	72	66
45,5	81,9	73	67,8	91,4	61,5	66
1,64	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

# 1. Hash table - coliziuni

Funcția de hash preia conținutul de dimensiune aleatoare (e.g. șir de oricâte caractere) și îl transformă într-o valoare de dimensiune fixă (un întreg). Exemplu: suma codurilor ASCII modulo 7.

Andy => A (65) + n (110) + d (100) + y (121) => 396 % 7 = 4

Bob => B (66) + o (111) + b (98) => 275 % 7 = 2

Carl => C (67) + a (97) + r (114) + l (108) => 386 % 7 = 1

Charlie => C (67) + h (104) + a (97) + r (114) + l (108) + i (105) + e (101) => 696 % 7 = 3

Heather => H (72) + e (101) + a (97) + t (116) + h (104) + e (101) + r (114) => 705 % 7 = 5

Sena => S (83) + e (101) + n (110) + a (97) => 391 % 7 = 6

~~Sue => S (83) + u (117) + e (101) => 301 % 7 = 0~~

Bert => B (66) + e (101) + r (114) + t (116) => 397 % 7 = 5

coliziune

array

	Carl	Bob	Charlie	Andy	Heather	Sena
	22	45	16	31	72	66
	81,9	73	67,8	91,4	61,5	66
	1,77	1,88	1,74	1,93	1,56	1,66
0	1	2	3	4	5	6

## 1. Hash table - coliziuni

Dacă un element trebuie așezat în tabel pe o poziție ocupată apare o coliziune. O soluție pentru a rezolva coliziunile este să se creeze o listă simplu înlănțuită pe fiecare poziție din tabel. Elementele care sunt plasate pe poziții deja ocupate vor fi adăugate la finalul acelei liste. În acest caz, tabelul hash devine un vector de liste.

array

0		
1	Carl 22 81,9 1,77	
2	Bob 45 73 1,88	
3	Charlie 16 67,8 1,74	
4	Andy 31 91,4 1,93	
5	Heather 72 61,5 1,56	Bert 61 73,2 1,89
6	Sena 66 66 1,66	

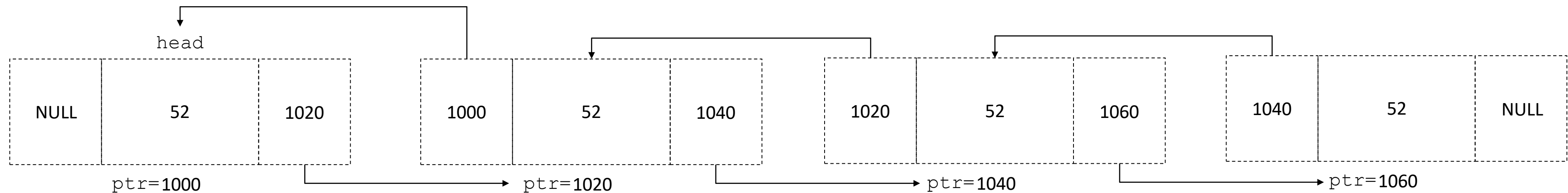
`get("Bert")`

1. Se calculează valoarea hash-ului pentru "Bert":  $h("Bert") = 5$ ;
2. Se parcurge element cu element lista de pe poziția `array[5]`;
3. Dacă elementul curent conține cheia "Bert" se întoarce pointer către nodul curent;
4. Altfel, se avansează la următorul element din listă și se reia pasul 3, până la întâlnirea cheii "Bert" sau epuizarea listei.

## 2. Liste dublu înlanțuite

Listele dublu înlanțuite sunt, asemenea listelor simplu înlanțuite, structuri dinamice de date cu elemente conectate prin legături. Spre deosebire de ele, însă, nodurile au legături și către nodurile precedente, nu numai către următorul nod.

```
struct nod{
    int data;
    struct nod * next;
    struct nod * prev;
};
```



## 2. Liste dublu înlănțuite

```
struct nod * creare_nod(){
    struct nod * nod_nou = malloc (sizeof (struct nod));

    if (!nod_nou){
        printf("Eroare de alocare\n");
    }

    scanf("%d", &nod_nou->data);
    nod_nou->next = NULL;
    nod_nou->prev = NULL;

    return nod_nou;
}
```

```
struct nod * adaugare_nod_inceput_lista(struct nod * head) {
    struct nod * nod_nou = creare_nod();

    if (head == NULL) {
        return nod_nou;
    }

    nod_nou->next = head;
    head->prev = nod_nou;

    return nod_nou;
}
```

```
struct nod{
    int data;
    struct nod * next;
    struct nod * prev;
};
```

```
struct nod * adaugare_nod_sfarsit_lista(struct nod * head) {
    struct nod * nod_nou = creare_nod();
    struct nod * temp = head;

    if (head == NULL) {
        return nod_nou;
    }

    while(temp->next != NULL){
        temp = temp->next;
    }

    temp->next = nod_nou;
    nod_nou->prev = temp;

    return head;
}
```