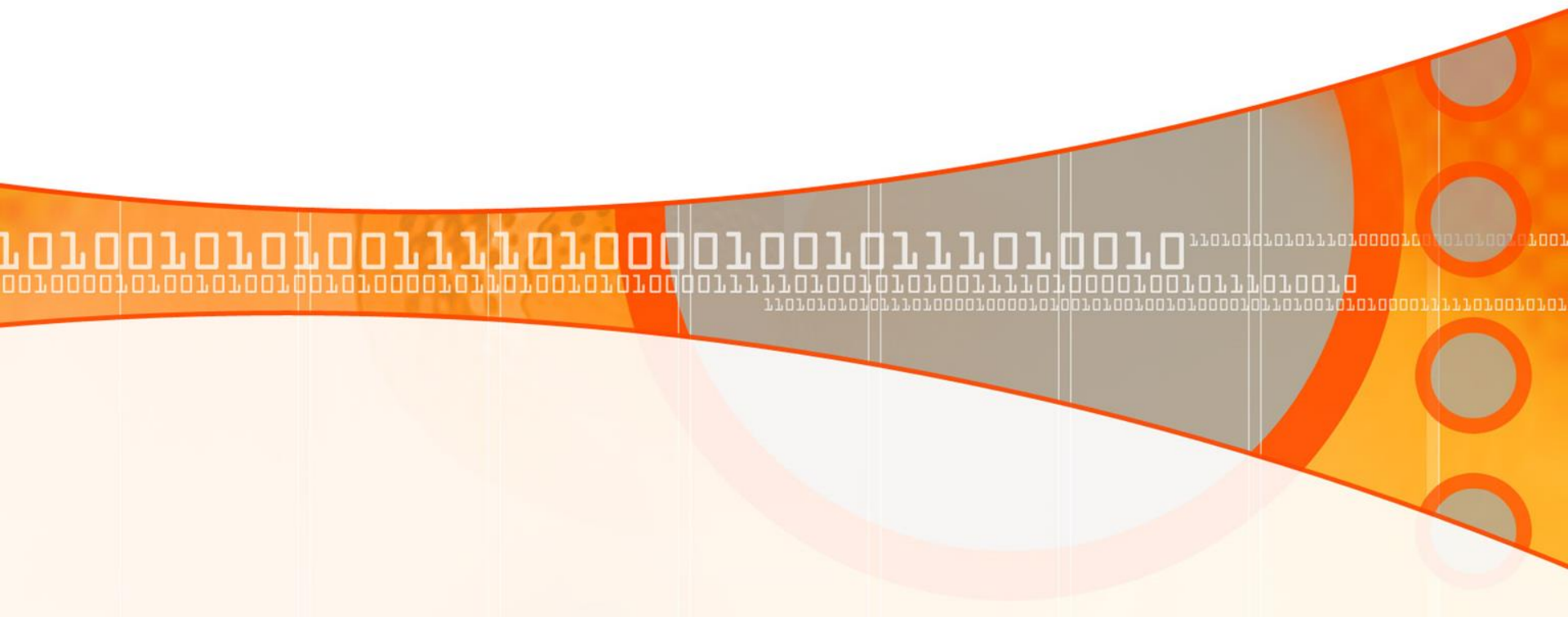
STRUCTURI DE DATE

Tabele de dispersie



TABELE DE DISPERSIE

Operatia de cautare a datelor:

- **Fiecare valoare din colectia de date are asociata o pozitie unica in colectie;**
- **Se determina pozitia valorii in colectia de date.**

Asocierea valoare cheie numerica-pozitie:

- **Structura VECTOR;**
- **Numarul de elemente=valoare maxima a cheii de cautare;**

TABELE DE DISPERSIE

Asocierea valoare cheie numerica-pozitie (continuare):

- **Elementele: exista sau sunt sterse logic (valoare element din afara valorilor de cheie);**
- **Cautare date in acces direct => MINIMIZARE timp de regasire.**

TABELE DE DISPERSIE

Dezavantaje:

- **Dimensiunea memoriei ocupate:**

  *MEMORIE = maxim(valoare cheie căutare) * dimensiune(element)*

Valoare maximă foarte mare => spaţiu de memorie considerabil

- **Nu se ţine cont de numărul real de elemente utilizate; cazul cel mai nefavorabil: numar foarte mic de elemente si valoare mare a cheii maxime;**

TABELE DE DISPERSIE

Dezavantaje (continuare):

- **Tipul cheii de cautare: tip numeric – trebuie sa fie index in accesarea elementelor din vector;**

TABELE DE DISPERSIE

Eliminarea dezavantajelor: tabele de dispersie (hash tables).

Tabela de dispersie:

- **Structura de stocare si cautare;**
- **Cheia de cautare asociata cu pozitia elementului in colectia de date prin functie hash.**

TABELE DE DISPERSIE

Avantaje ale utilizarii tabelului de dispersie:

- **Utilizare mai eficienta a resursei memorie: nu se stocheaza elemente care nu sunt utilizate;**

Funcţie hash cu un nivel de complexitate scăzut:

hash(X) = X modulo 1500, $X \in [54130,55630]$

TABELE DE DISPERSIE

Avantaje ale utilizarii tabelului de dispersie (continuare):

- **Implementarea de chei alfanumerice: se poate utiliza tip alfanumeric pentru cheia de cautare;**

Funcţia hash translateaza valoarea alfanumerică într-o valoare întreagă pozitivă.

TABELE DE DISPERSIE

Funcţia hash pentru un string (in limbajul Java):

$$hash(S) = s[0]*31^{n-1}+s[1]*31^{n-2}+\ldots+s[n-2]*31+s[n-1]$$

s[i] - codul ASCII ;

n - dimensiunea şirului de caractere

$$hash(\text{"salut"})=115*31^4+97*31^3+108*31^2+117*31+116=10920217$$

TABELE DE DISPERSIE

Valoarea obţinută: introdusă din nou într-o funcţie hash pentru a identifica poziţia corespondentă din colectia de date.

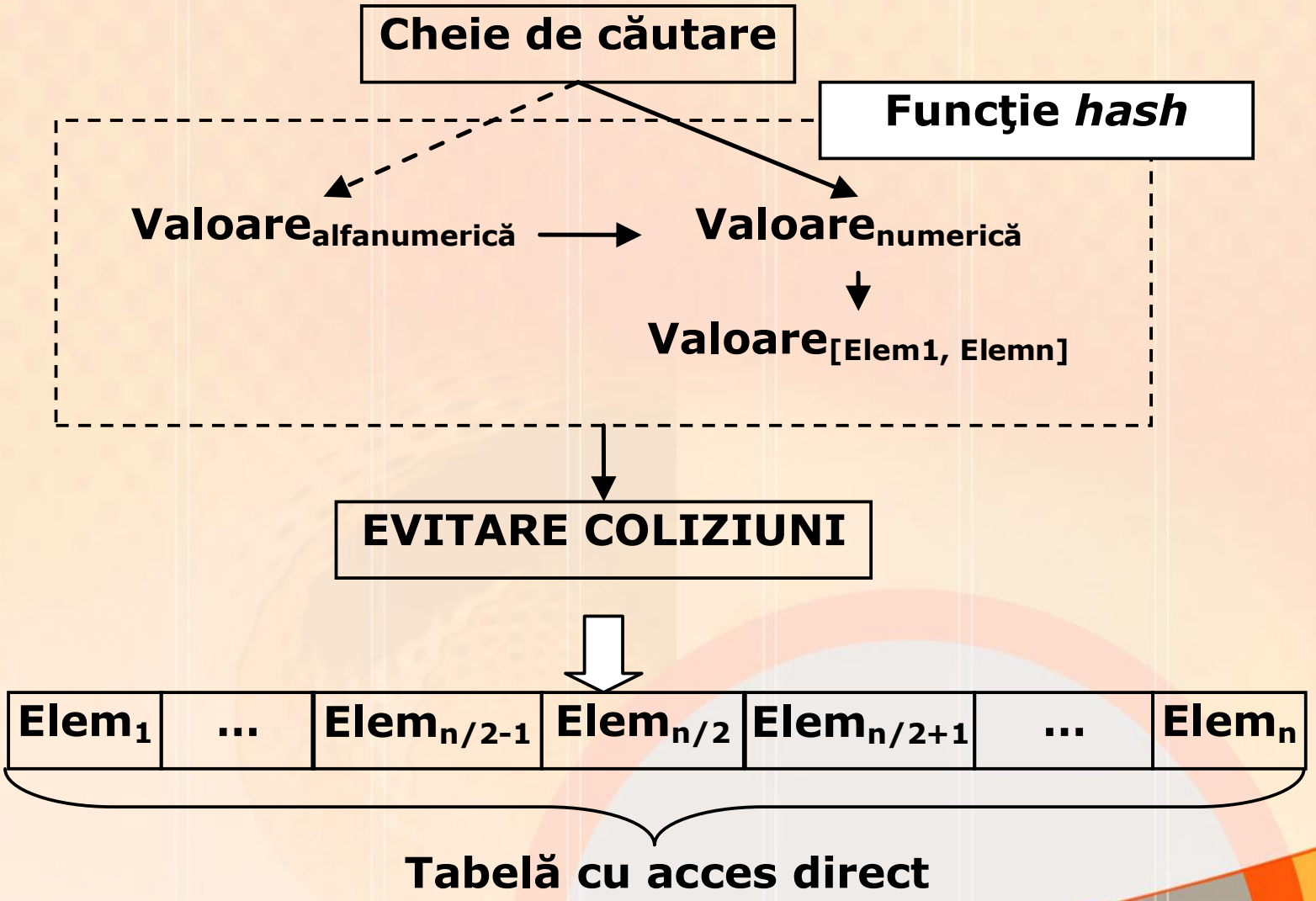
TABELE DE DISPERSIE

Dezavantajul tabelelor de dispersie:

- **Prelucrare suplimentara data de funcţia hash care poate avea în unele situaţii un nivel de complexitate ridicat;**

- **Apariţia în cadrul tabelei a coliziunilor: două valori X_h şi Y_h conduc la $hash(X_h) = hash(Y_h)$; evitarea coliziunilor: operaţii suplimentare precum *chaining, re-hashing, linear probing, quadratic probing* şi *overflow area***

TABELE DE DISPERSIE



TABELE DE DISPERSIE

În funcţie de tipul valorii cu rol de cheie:

- Chei numerice: tipuri fundamentale definite de limbajul de programare utilizat;
- Chei alfanumerice: şiruri de caractere;
- Chei compuse: mai multe atribute.

TABELE DE DISPERSIE

Funcţia hash:

- **Prelucreaza cheia asociată fiecărei înregistrări;**
- **Determina poziţia în cadrul tabelei de dispersie a elementului;**
- **Nu există o funcţie hash generală;**
- **Alegerea functiei hash: în funcţie de caracteristicile mulţimii de valori chei.**

TABELE DE DISPERSIE

Modele matematice ale functiei hash:

- **Impărţire în modul: complexitate scazuta, usurinta de implementare; cheia de căutare este transformată într-o valoare numerică şi apoi transpusă în mulţimea [0; n-1], n dimensiunea tabelei de dispersie:**

pozitie_tabela = val_cheie % val_baza

pozitie_tabela: valoarea hash obţinută

val_cheie: valoarea cheie numerică

*val_baza: dimensiunea tabelei de dispersie; numere prime apropiate de numărul total de înregistrări; caz general: (4*i+3) cu i = 0, 1, 2, 3, …*

TABELE DE DISPERSIE

Modele matematice ale functiei hash (continuare):

- **Inmulţirea cu un număr real aleatoriu din [0;1) şi prelucrarea ulterioară a părţii zecimale cuprinsa in [0 ;1); inmultirea rezultatului cu dimensiunea tabelei de dispersie n duce la obtinerea pozitiei elementului in [0; n-1];**

$$val\_hash=((val\_cheie*random_{[0;1)})-[(val\_cheie*random_{[0;1)})]]*n$$

val_hash: valoare hash

val_cheie: valoare cheie de căutare

$random_{[0;1)}$: număr aleatoriu din [0;1)

n: dimensiune tabela

TABELE DE DISPERSIE

Modele matematice ale functiei hash (continuare):

- **Prelucrarea codurilor ASCII ale caracterelor alfanumerice: pe baza primului caracter din cheie se defineşte relaţia:**

val_hashs_1 = string_cheie[0] % 255

val_hashs_1: valoarea hash

string_cheie: valoarea cheie de căutare

TABELE DE DISPERSIE

val_hashs_1 : model cu un nivel de complexitate scăzut pentru gestiunea unei colectivităţi mici de elemente.

Modelul este ineficient deoarece generează multe coliziuni pentru siruri diferite care încep cu acelaşi caracter.

TABELE DE DISPERSIE

Rafinarea modelului: preluarea mai multor caractere din şirul pentru care se determina valoarea hash (primul si ultimul caracter):

$val\_hashs_2 = (string\_cheie[0] + string\_cheie[lungime_{string\_cheie}]) \% n$

val_hashs_2: valoare hash

string_cheie: cheie de căutare

$lungime_{string\_cheie}$: dimensiune şir de caractere

n: dimensiune tabela de dispersie

TABELE DE DISPERSIE

Pentru a nu reduce dimensiunea tabelei la maxim 255 elemente, se utilizează un număr prim, n suficient de mare

Alte funcţii hash de prelucrare a cheilor alfanumerice analizează toate caracterele din şir:

$$val\_hashs_3 = \sum_{i=1}^{lungime\_cheie} ASCII(string\_cheie[i]) \% n$$

TABELE DE DISPERSIE

Evitarea coliziunilor:

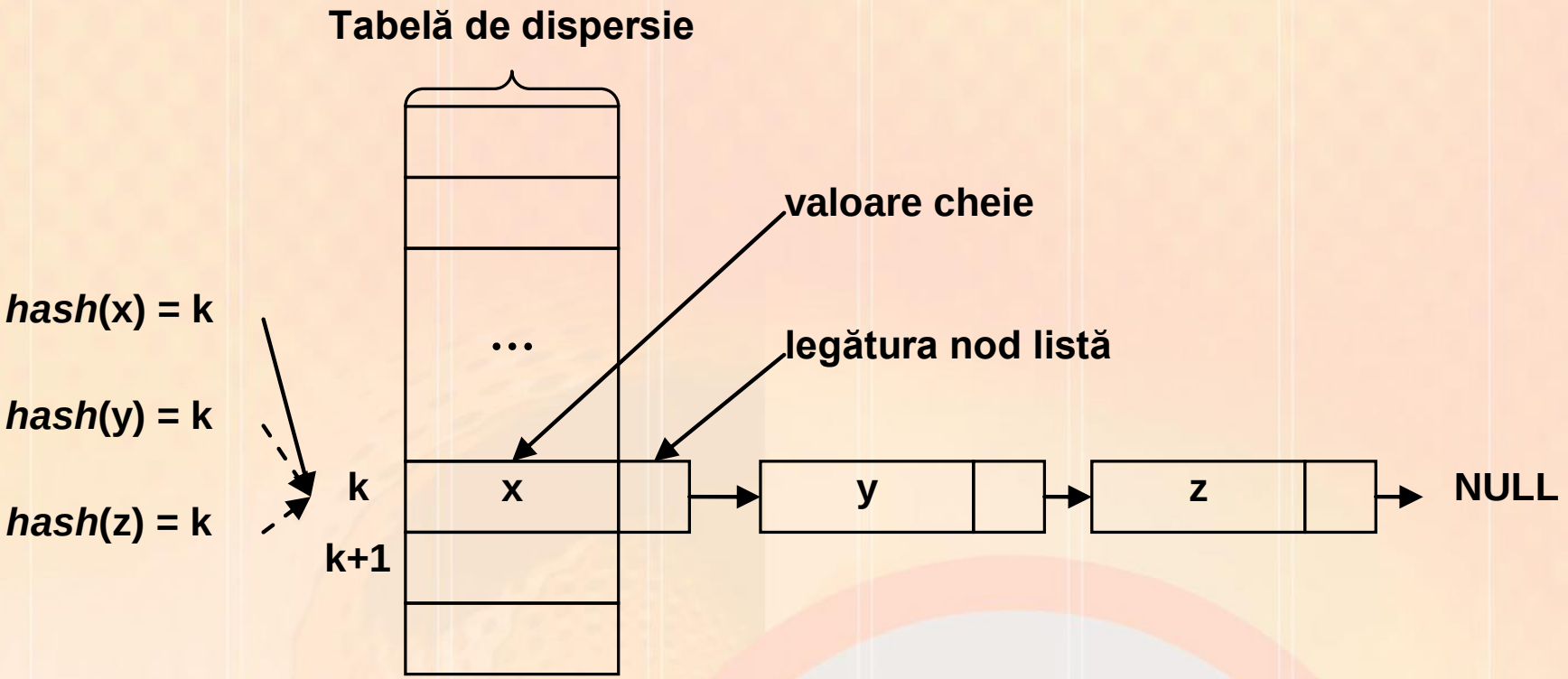
- **Metode de regăsire a elementelor descrise de chei cu valori diferite, dar care conduc la valori hash identice;**
- **Chaining, re-hashing , linear probing, quadratic probing, overflow area.**

TABELE DE DISPERSIE

Chaining:

- **Implementrează lucrul cu liste;**
- **Fiecare poziţie tabela de dispersie conţine adresa unei liste de elemente cu valori hash egale;**
- **Regăsirea unui element: determinarea poziţiei în cadrul tabelei prin calcularea valorii hash şi parcurgerea secvenţială a listei ataşate poziţiei.**

TABELE DE DISPERSIE

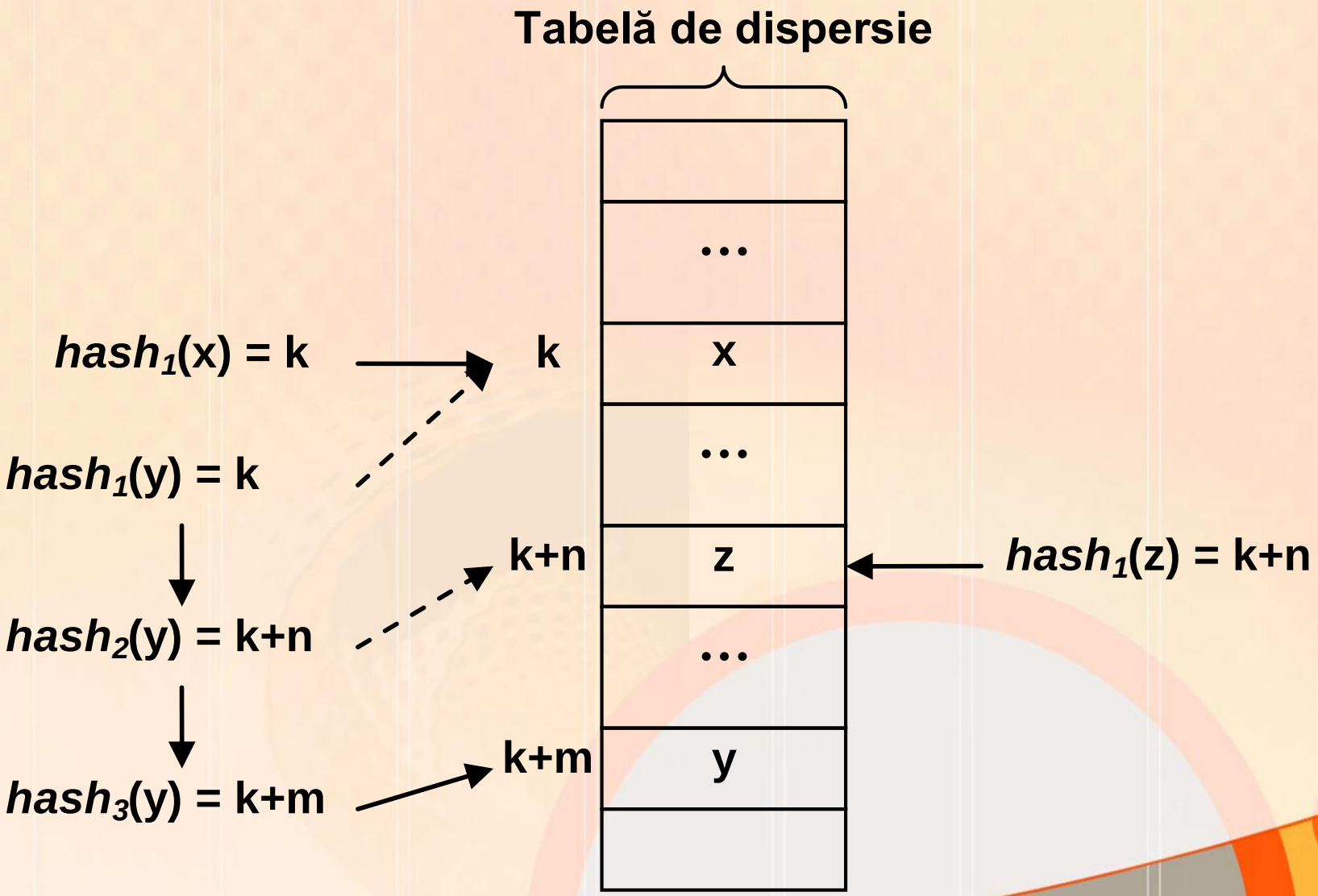


TABELE DE DISPERSIE

Re-hashing:

- **Aplicarea în cascadă a aceleiaşi funcţii hash sau a altui model dintr-o mulţime de funcţii până când valoarea obţinută reprezintă o poziţie liberă din tabela de dispersie;**
- **La fiecare pas al procesului de cautare: valoarea cheii de căutare este introdusă într-o listă de funcţii hash până când se identifică elementul cu valoarea căutată sau nu mai există alte posibilităţi de a recalcula valoarea hash.**

TABELE DE DISPERSIE

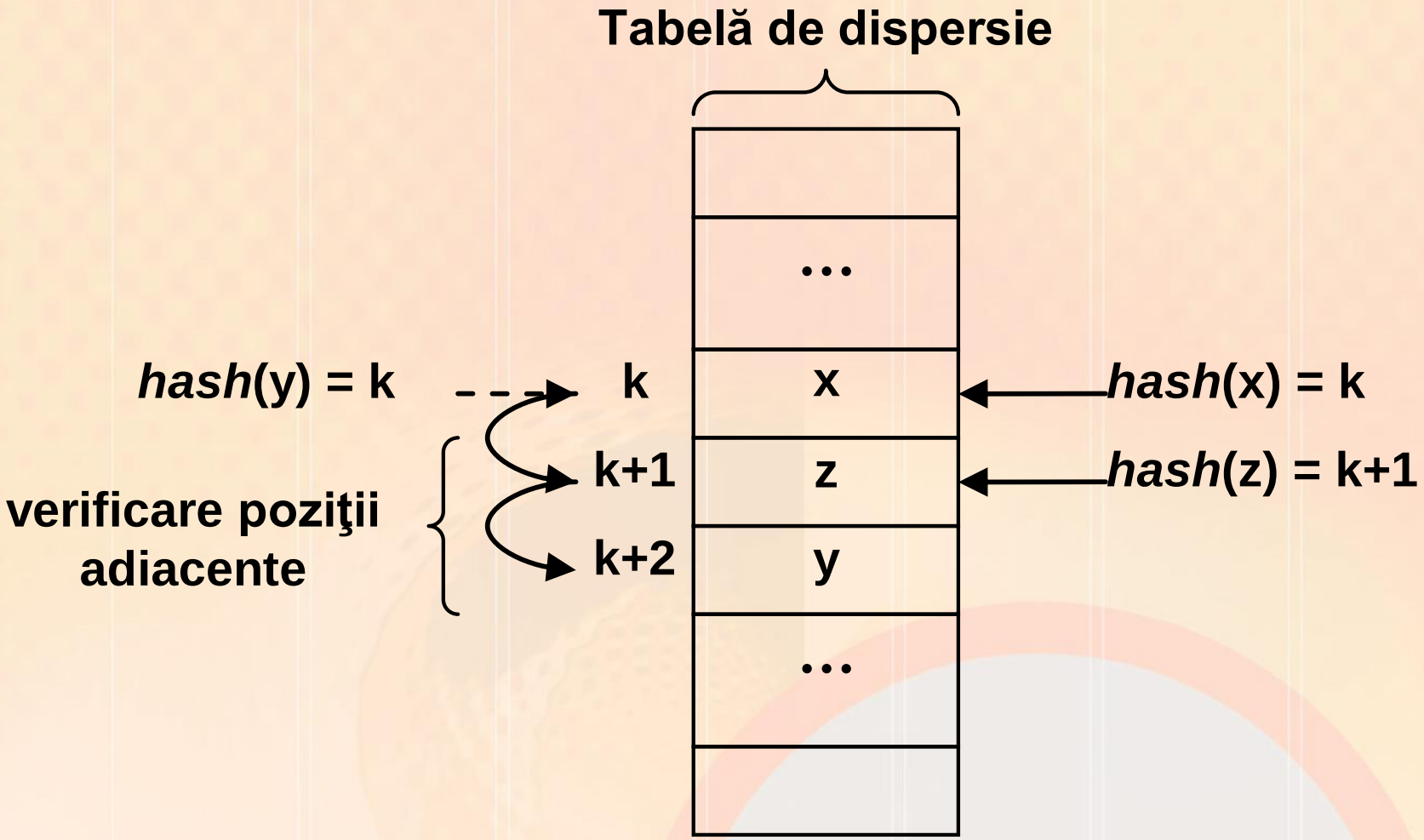


TABELE DE DISPERSIE

Linear probing:

- **Căutarea secvenţială a primei poziţii libere unde este inserat elementul nou (la stânga sau la dreapta coliziunii);**
- **La căutare: verificarea elementelor adiacente poziţiei indicate de valoarea hash;**
- **Gruparea coliziunilor de acelaşi tip în aceeaşi zonă (cluster); rezulta creşterea probabilităţii de apariţie a coliziunilor pentru valorile hash adiacente.**

TABELE DE DISPERSIE



TABELE DE DISPERSIE

Quadratic probing:

- **Evită crearea grupurilor de coliziuni prin utilizarea unui pas de regăsire a următoarei poziţii libere diferit de 1; salturi în tabela de dispersie din două în două poziţii sau din patru în patru;**

TABELE DE DISPERSIE

Quadratic probing (continuare):

- **determinarea următoarei poziţii de inserat:**

$$poziţie = hash(X)+c*i^2$$

poziţie: noua poziţie din tabela pentru inserare sau cautare element

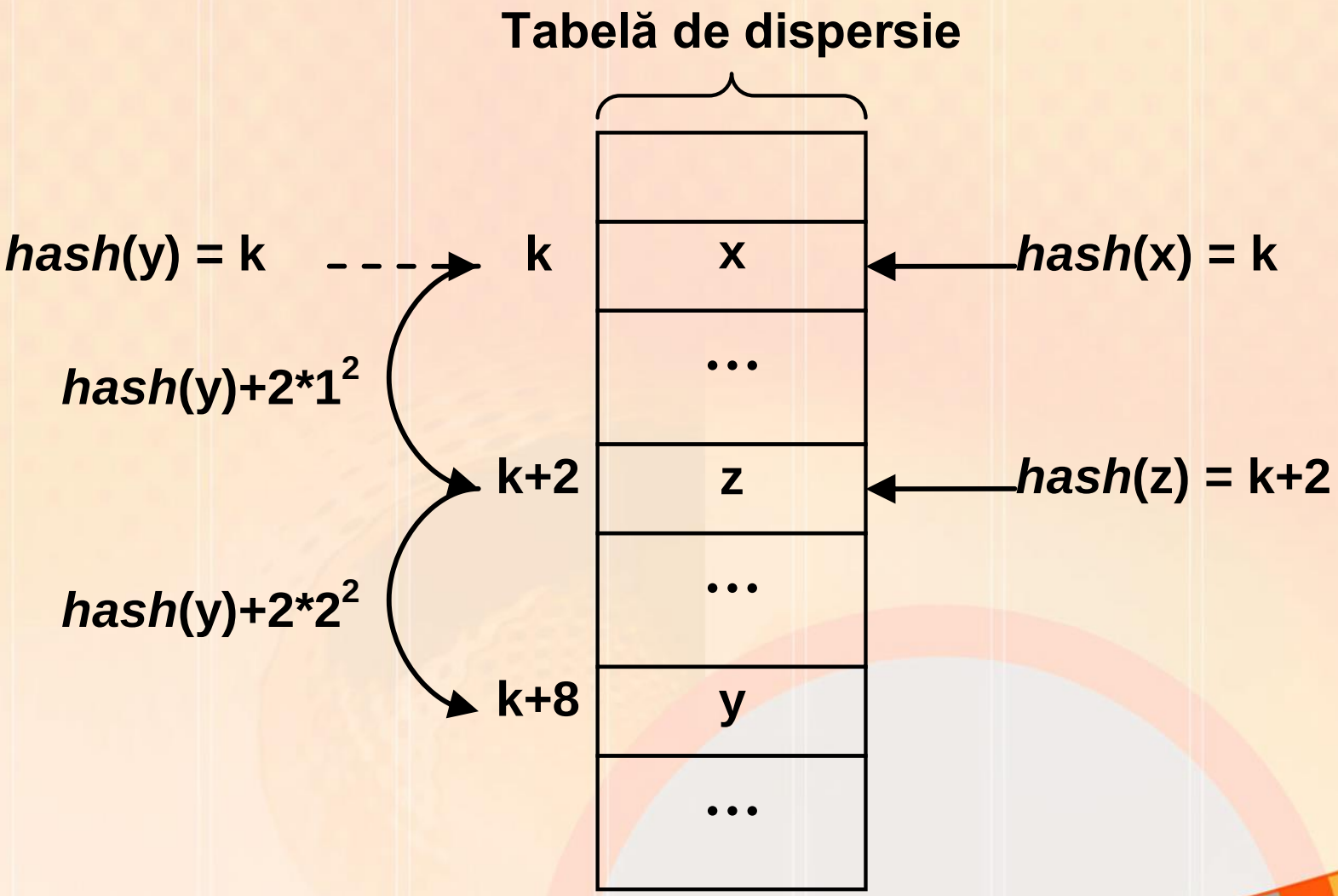
X: cheia asociate elementului

hash(X): poziţia indicată de valoarea hash a elementului

c: valoare constantă {1, 2, 4}

i: număr operaţie re-hash sau număr de poziţii verificate

TABELE DE DISPERSIE



TABELE DE DISPERSIE

Overflow area:

- **împarte tabela de dispersie în:**

  - Zona primară: reţinerea elementelor iniţiale;

  - Zona secundară alocată elementelor ce generează coliziuni;

- **se utilizează un element al zonei secundare pentru a reţine noua valoare sau pentru a continua căutarea;**

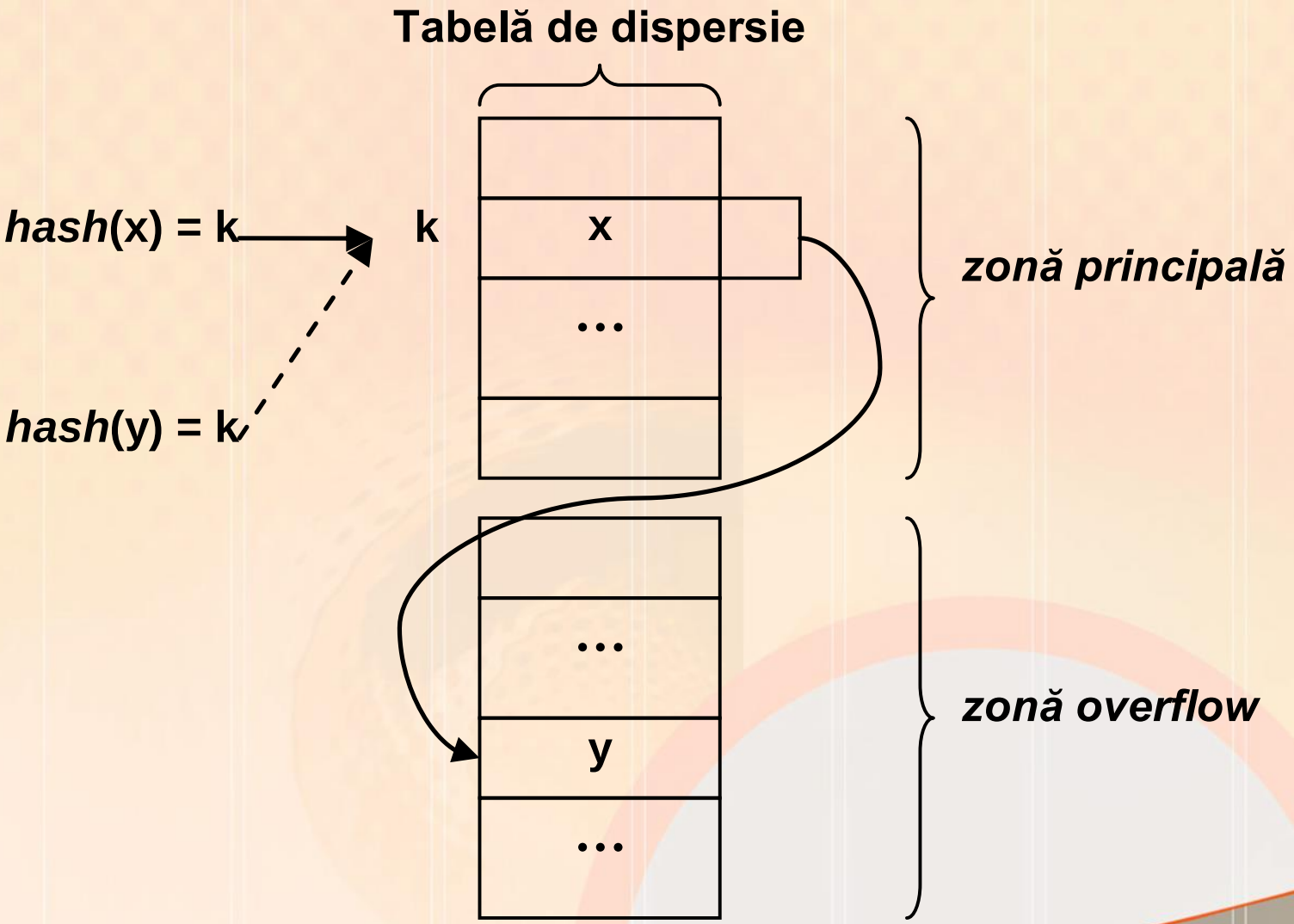
TABELE DE DISPERSIE

Overflow area (continuare):

- **Accesul la zona secundară: prin pointer din zona primară;**

- **Regăsire mai rapidă a informaţiilor decât metoda chaining**

TABELE DE DISPERSIE



TABELE DE DISPERSIE

Probabilitatea de apariţie a coliziunilor la inserare sau la căutare creşte proporţional cu gradul de utilizare a tabelei.

Funcţiile hash cu un grad redus de complexitate nu conduc la rezultate unice pentru valori de intrare distincte.

TABELE DE DISPERSIE

Cu cât tabela are un număr din ce în ce mai mic de poziţii disponibile, cu atat creşte riscul de a avea elemente cu chei de căutare diferite dar care se regăsesc pe poziţii identice.

Eficienţa operaţiei de căutare la un nivel acceptabil: grad de ocupare a tabelei de dispersie < 50% (ineficienta a spatiului, viteza de cautare mare)