

# Tabelas Hash

As tabelas Hash, também conhecidas como tabelas de dispersão, armazenam elementos com base no valor absoluto de suas chaves e em técnicas de tratamento de colisões. Para o cálculo do endereço onde deve ser armazenada uma determinada chave, utiliza-se uma função denominada função de dispersão, que transforma a chave em um dos endereços disponíveis na tabela.

Suponha que uma aplicação utilize uma tabela de dispersão com 13 endereços-base (índices de 0 a 12) e empregue a função de dispersão  $h(x) = x * mod13$ , em que  $x$  representa a chave do elemento cujo endereço-base deve ser calculado.

Se a chave  $x$  for igual a 49, a função de dispersão retornará o valor 10, indicando o local onde esta chave deverá ser armazenada. Se a mesma aplicação considerar a inserção da chave 88, o cálculo retornará o mesmo valor 10, ocorrendo neste caso uma colisão. O Tratamento de colisões serve para resolver os conflitos nos casos onde mais de uma chave é mapeada para um mesmo endereço-base da tabela. Este tratamento pode considerar, ou o recálculo do endereço da chave ou o encadeamento externo ou exterior.

O professor gostaria então que você o auxiliasse com um programa que calcula o endereço para inserções de diversas chaves em algumas tabelas, com funções de dispersão e tratamento de colisão por encadeamento exterior.

## Entrada

A entrada contém vários casos de teste. A primeira linha de entrada contém um inteiro  $N$  indicando a quantidade de casos de teste. Cada caso de teste é composto por duas linhas. A primeira linha contém um valor  $M$  ( $1 \leq M \leq 100$ ) que indica a quantidade de endereços-base na tabela (normalmente um número primo) seguido por um espaço e um valor  $C$  ( $1 \leq C \leq 200$ ) que indica a quantidade de chaves a serem armazenadas. A segunda linha contém cada uma das chaves (com valor entre 1 e 200), separadas por um espaço em branco.

## Saída

A saída deverá ser impressa conforme os exemplos fornecidos abaixo, onde a quantidade de linhas de cada caso de teste é determinada pelo valor de  $M$ . Uma linha em branco deverá separar dois conjuntos de saída.

## Exemplo 1

### Entrada

```
2
13 9
```

```
44 45 49 70 27 73 92 97 95
```

```
7 8
```

```
35 12 2 17 19 51 88 86
```

## Saída

```
0 -> \
```

```
1 -> 27 -> 92 -> \
```

```
2 -> \
```

```
3 -> \
```

```
4 -> 95 -> \
```

```
5 -> 44 -> 70 -> \
```

```
6 -> 45 -> 97 -> \
```

```
7 -> \
```

```
8 -> 73 -> \
```

```
9 -> \
```

```
10 -> 49 -> \
```

```
11 -> \
```

```
12 -> \
```

```
0 -> 35 -> \
```

```
1 -> \
```

```
2 -> 2 -> 51 -> 86 -> \
```

```
3 -> 17 -> \
```

```
4 -> 88 -> \
```

```
5 -> 12 -> 19 -> \
```

```
6 -> \
```

Author: Neilor Tonin, Brasil