ESTRUTURA DE DADOS II

13ª ATIVIDADE

Tabela de Dispersão ou Tabela Hash

Prof. Eugênio Júlio Messala Cândido Carvalho
eugeniojuliomessla@gmail.com
(62) 99241-0579

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	5	João Neves
5	6	Igor Carlos
6	7	Diogo Carvalho
7	8	Mario Teles
8	10	Cristiane Maria
9		
10		

Dados armazenados ordenados pela chave. Como executar as operações:

Incluir Dados

Excluir Dados

Consultar Dados

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	5	João Neves
5	6	Igor Carlos
6	7	Diogo Carvalho
7	8	Mario Teles
8	10	Cristiane Maria
9		
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Incluir Dados

Chave: 4

Nome: Godofredo da Silva

Mover os dados da tabela - abrindo o espaço necessário para a inclusão.

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4		
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9	10	Cristiane Maria
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Incluir Dados

Chave: 4

Nome: Godofredo da Silva

Incluir os dados na tabela

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9	10	Cristiane Maria
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Incluir Dados

Chave: 4

Nome: Godofredo da Silva

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9	10	Cristiane Maria
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Excluir Dados:

Chave: 5

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9	10	Cristiane Maria
10		

Dados armazenados ordenados pela chave.

Como executar as operação:

Excluir Dados:

Chave: 5

Encontrar o dado de chave 5 na tabela - busca

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5		
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9	10	Cristiane Maria
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Excluir Dados:

Chave: 5

Apagar o dado da tabela

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5	6	Igor Carlos
6	7	Diogo Carvalho
7	8	Mario Teles
8	10	Cristiane Maria
9		
10		

Dados armazenados ordenados pela chave.

Como executar as operações:

Excluir Dados:

Chave: 5

Mover os elemento da tabela

TABELA HASH

Vetor	Dados	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4	4	Godofredo da Silva
5	6	Igor Carlos
6	7	Diogo Carvalho
7	8	Mario Teles
8	10	Cristiane Maria
9		
10		

Dados armazenados ordenados pela chave.

Como executar a operação:

Consultar Dados

Chave: 7

Buscar pelo elemento na tabela

TABELA HASH

Muitas aplicações exigem um conjunto dinâmico que admita apenas as operações de inserção, consulta e exclusão.

Uma tabela hash é uma generalização da noção mais simples de um arranjo comum (vetor). O endereçamento direto em um arranjo comum faz uso eficiente de nossa habilidade para examinar uma posição arbitrária de forma direta.

O endereçamento direto é aplicável quando temos condições de alocar um arranjo que tenha uma única posição para cada chave possível.

TABELA HASH

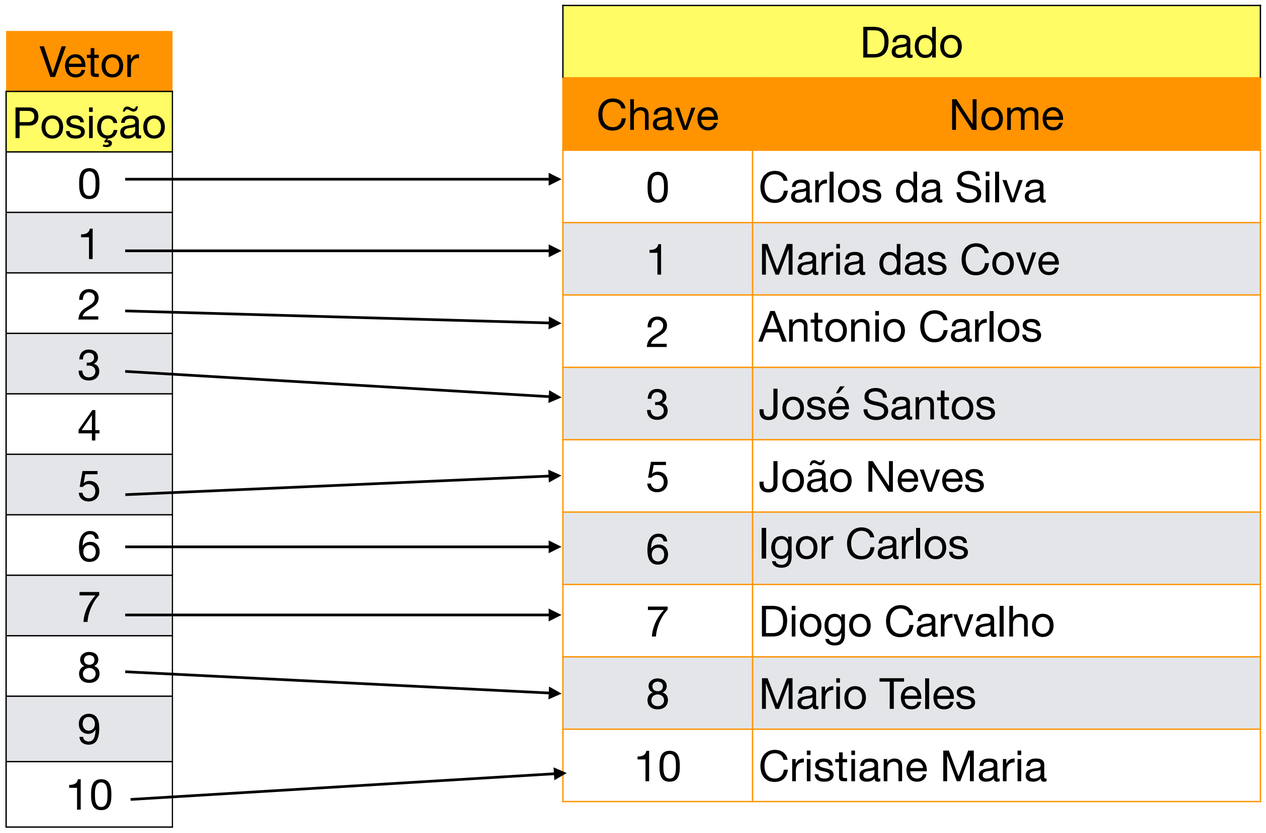
Tabela de Endereço Direto

O endereçamento direto é uma técnica simples que funciona bem quando o universo U de chaves é razoavelmente pequeno. Suponha que uma aplicação necessite de um conjunto dinâmico no qual cada elemento tenha uma chave definida a partir do universo U= {0, 1, 2, …, m-1} onde m não e muito grande. Iremos supor que não existe dois elementos com a mesma chave.

Para representar o conjunto dinâmico, usamos um arranjo ou uma tabela de endereço direto T[0..m-1], na qual cada posição corresponde a uma chave no universo U.

TABELA HASH

Endereçamento Direto - Conjunto de 11 chaves



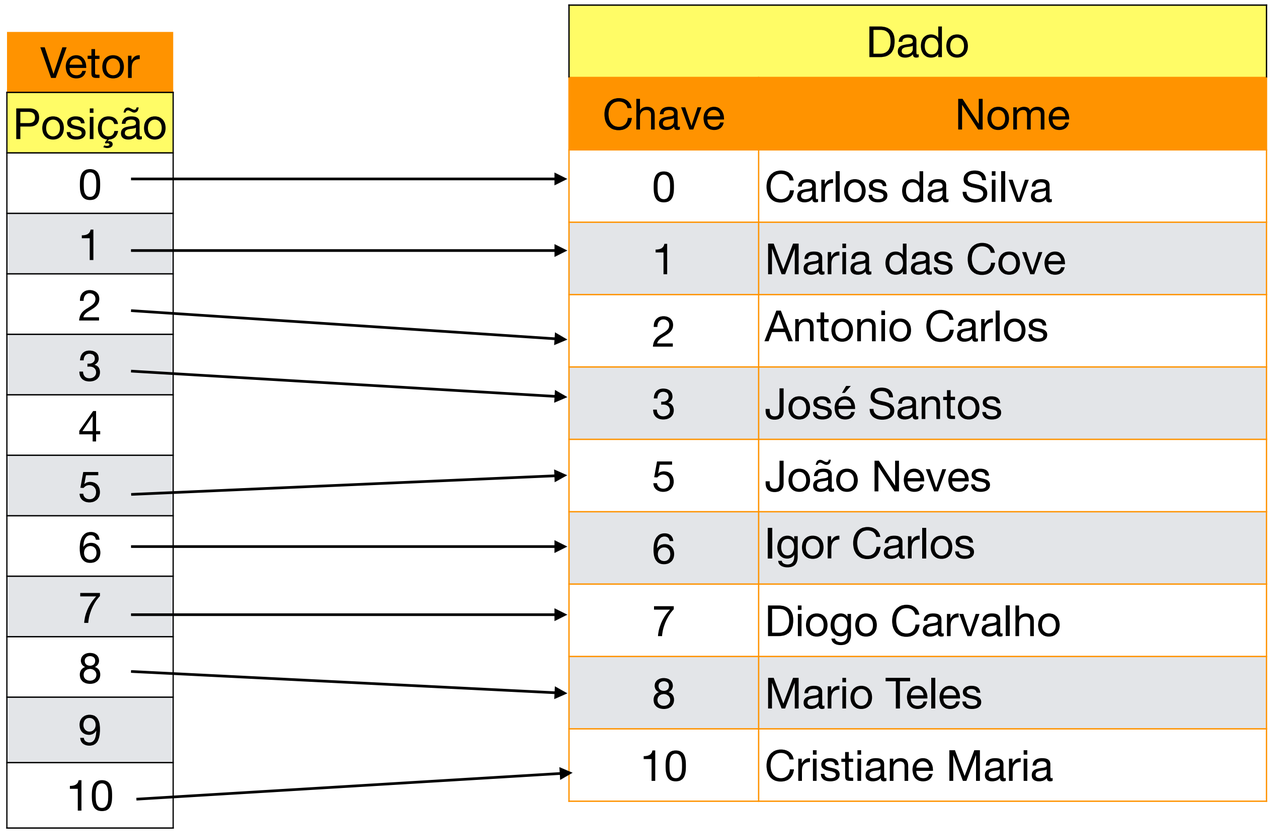
Vetor	Dado	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4		
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9		
10	10	Cristiane Maria

Como executar as operações:

- **Incluir Dados**
- **Excluir Dados**
- **Consultar Dados**

TABELA HASH

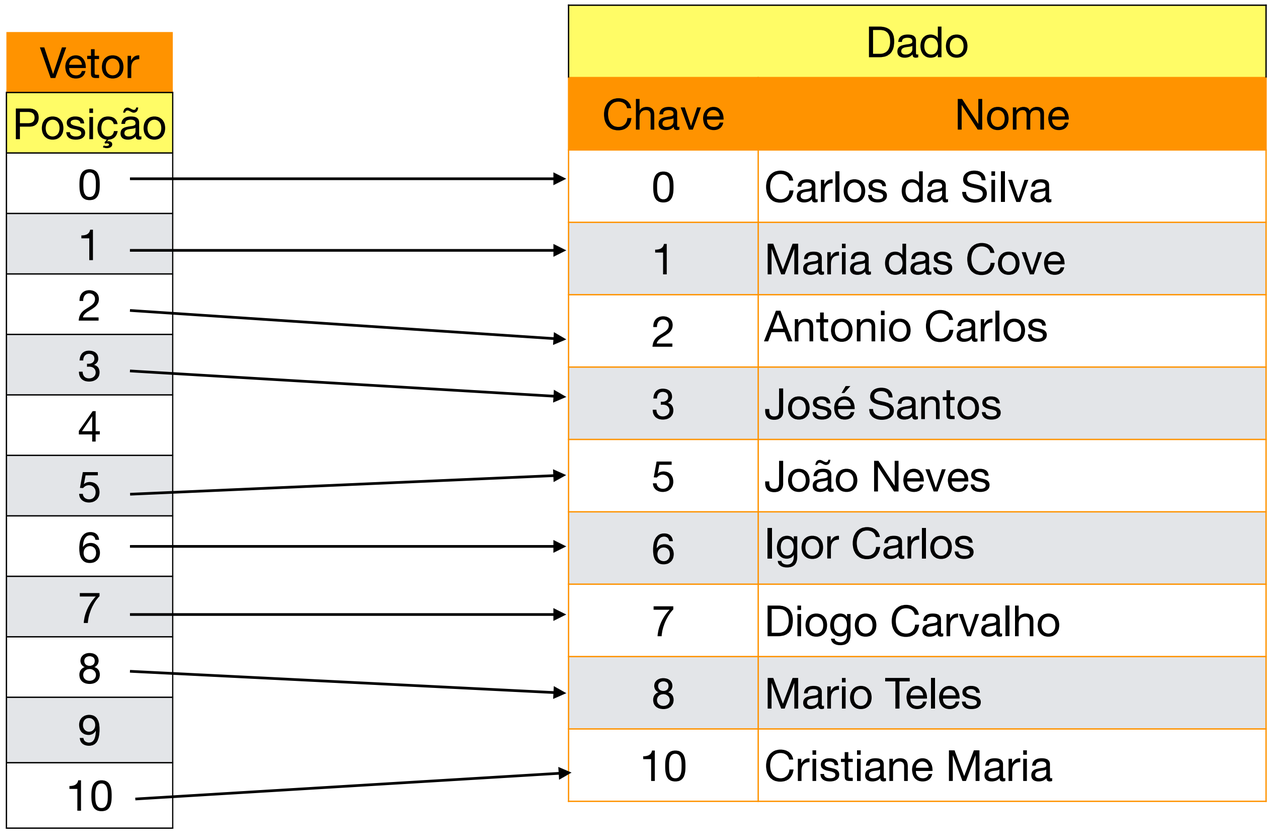
Endereçamento Direto - Conjunto de 11 chaves




```
Incluir(chave, dado){
    arranjo[ chave ] = dado;
}
```

TABELA HASH

Endereçamento Direto - Conjunto de 11 chaves

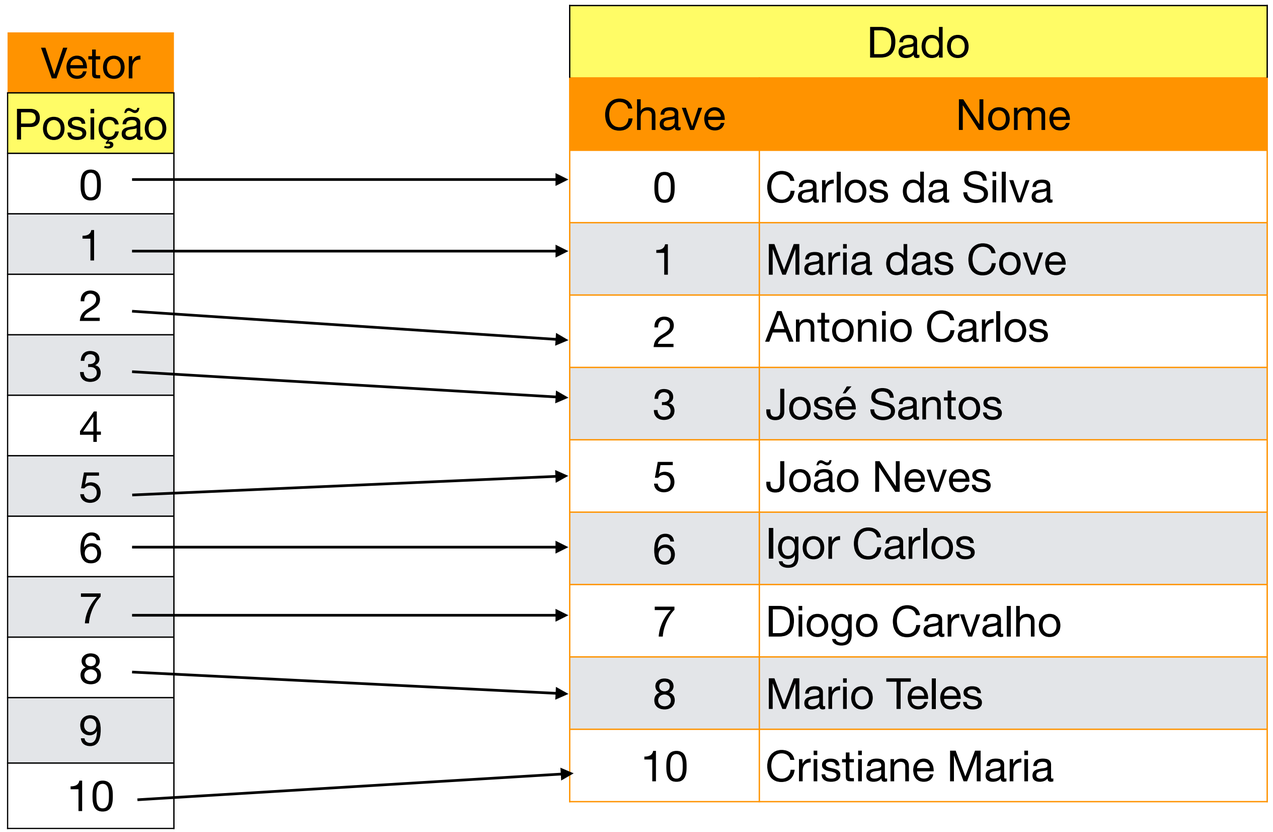


Vetor	Dado	
Posição	Chave	Nome
0	0	Carlos da Silva
1	1	Maria das Cove
2	2	Antonio Carlos
3	3	José Santos
4		
5	5	João Neves
6	6	Igor Carlos
7	7	Diogo Carvalho
8	8	Mario Teles
9		
10	10	Cristiane Maria

```
Excluir( chave ){
   arranjo[ chave ] = nulo;
}
```

TABELA HASH

Endereçamento Direto - Conjunto de 11 chaves




```
Consultar( chave ){
    return arranjo[ chave ];
}
```

TABELA HASH

A dificuldade com o endereçamento direto é obvia: se o universo U é grande, o armazenamento de uma tabela T de tamanho |U| pode ser impraticável, ou mesmo impossível, em virtude da memória disponível em um computador típico.

Além disso, o conjunto K de chaves realmente armazenadas pode ser tão pequeno em relação a U que a maior parte do espaço alocado para T seria desperdiçado.

Quando o conjunto K de chaves armazenadas é muito menor que o universo U de todas as chaves possíveis, uma tabela hash exige muito menos espaço de armazenamento que uma tabela de endereço direto.

TABELA HASH

Com o endereçamento direto, um elemento com chave k é armazenado na posição K.

No caso do hash, esse elemento é armazenado na posição ***h***(k); ou seja, uma ***função hash h*** é utilizada para calcular a posição a partir da chave K. Aqui, ***h*** mapeia o universo U de chaves nas posições de uma tabela hash T[0 … m-1]. O melhor valor para m, tamanho da tabela, seria um número primo próximo de (total de chaves/3).

Exemplo: um conjunto com 2000 possíveis chaves, aceitando até 3 colisões por chave seria uma tabela de tamanho m = 701. O número 701 foi escolhido porque é um primo próximo a 2000/3.

Dizemos que um elemento com a chave K efetua o hash para a posição ***h***(k); dizemos também que ***h***(k) é o valor hash da chave k.

TABELA HASH

Endereçamento Hash - Conjunto de 100 chaves
Tabela hash de tamanho 11

Vetor
Posição
0
1
2
3
4
5
6
7
8
9
10

Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	Igor Carlos	21 mod 11 = 10
11	Diogo Carvalho	11 mod 11 = 0
71	Cristiane Maria	71 mod 11 = 5

Calculo do Endereçamento
Método da Divisão

Função Hash = chave mod m

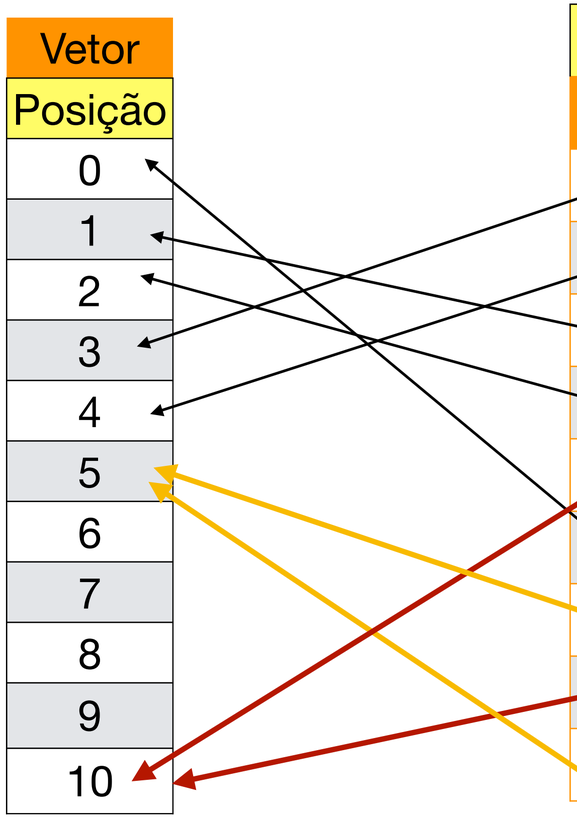
Onde m é o tamanho da tabela Hash

TABELA HASH

O detalhe fundamental dessa boa idéia é que duas ou mais chaves podem ter o hash na mesma posição. Chamamos essa situação de **colisão**. Porém existem técnicas eficientes para resolver o conflito criado por colisões.

TABELA HASH

Endereçamento Hash - Conjunto de 100 chaves
Tabela hash de tamanho 11
Colisão de Chaves



Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	**Igor Carlos**	**21 mod 11 = 10**
11	Diogo Carvalho	11 mod 11 = 0
71	**Cristiane Maria**	**71 mod 11 = 5**
32	**Mario Teles**	**32 mod 11 = 10**
16	**José Santos**	**16 mod 11 = 5**

Calculo do Endereçamento
Método da Divisão

Função Hash = chave mod m

Onde m é o tamanho da tabela Hash

Colisão das Chaves

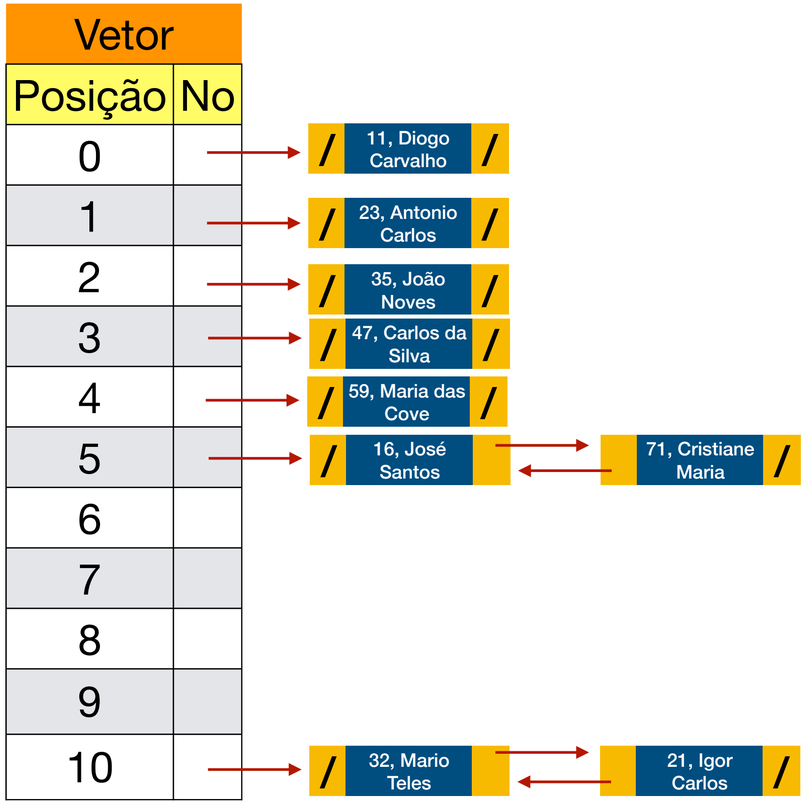
TABELA HASH

Resolução de colisões por encadeamento

No encadeamento, colocamos todos os elementos que efetuem hash para a mesma posição em uma lista ligada.

TABELA HASH

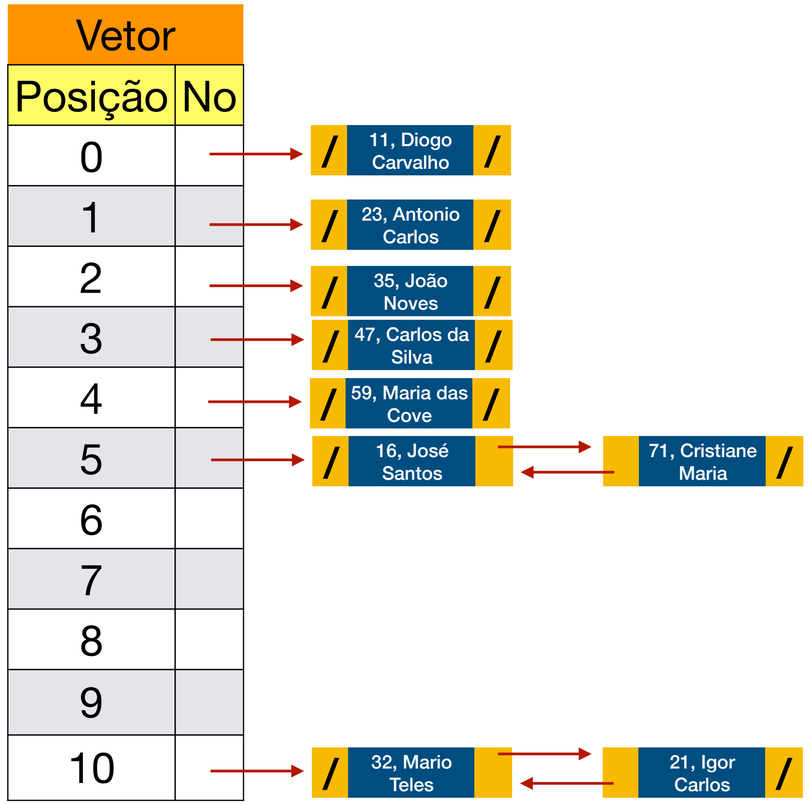
Colisão de Chaves
Resolvendo Colisões por Encadeamento



Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	**Igor Carlos**	**21 mod 11 = 10**
11	Diogo Carvalho	11 mod 11 = 0
71	**Cristiane Maria**	**71 mod 11 = 5**
32	**Mario Teles**	**32 mod 11 = 10**
16	**José Santos**	**16 mod 11 = 5**

TABELA HASH

As operações sobre uma tabela hash T são fáceis de implementar quando as colisões são resolvidas por encadeamento.



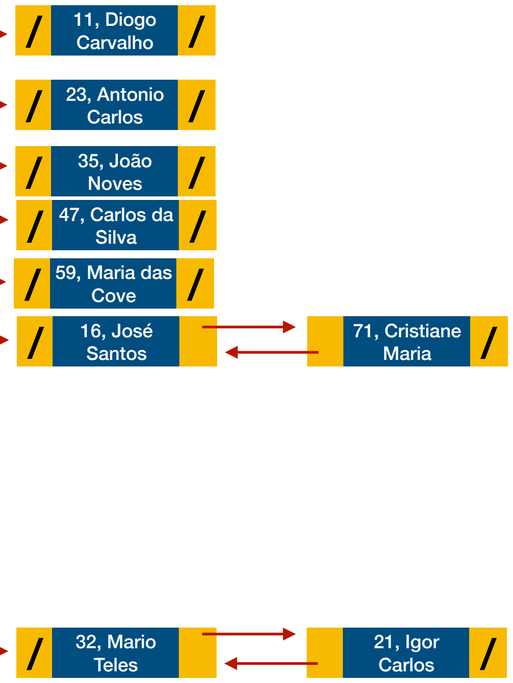
Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	Igor Carlos	21 mod 11 = 10
11	Diogo Carvalho	11 mod 11 = 0
71	Cristiane Maria	71 mod 11 = 5
32	Mario Teles	32 mod 11 = 10
16	José Santos	16 mod 11 = 5

Como executar as operações:

- **Incluir Dados**
- **Excluir Dados**
- **Consultar Dados**

TABELA HASH

Vetor	
Posição	No
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



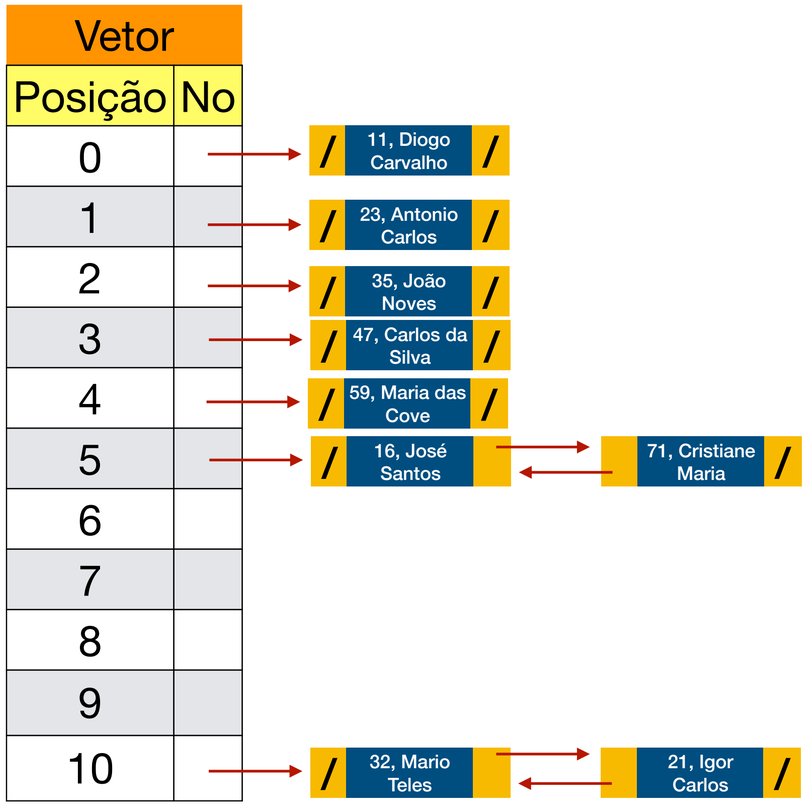
Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	Igor Carlos	21 mod 11 = 10
11	Diogo Carvalho	11 mod 11 = 0
71	Cristiane Maria	71 mod 11 = 5
32	Mario Teles	32 mod 11 = 10
16	José Santos	16 mod 11 = 5

Incluir(chave, dado){

Insere dado no inicio da lista T[*h*(chave)]

}

TABELA HASH



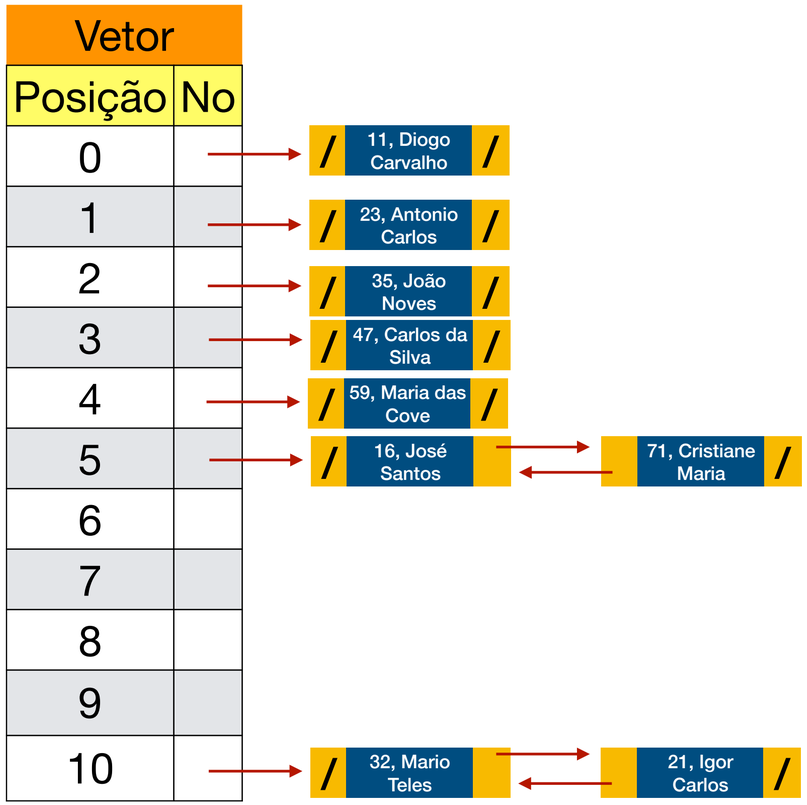
Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	Igor Carlos	21 mod 11 = 10
11	Diogo Carvalho	11 mod 11 = 0
71	Cristiane Maria	71 mod 11 = 5
32	Mario Teles	32 mod 11 = 10
16	José Santos	16 mod 11 = 5

Excluir(chave){

Elimina chave da lista T[*h*(chave)]

}

TABELA HASH



Dado		Função Hash
Chave	Nome	Valor Hash
47	Carlos da Silva	47 mod 11 = 3
59	Maria das Cove	59 mod 11 = 4
23	Antonio Carlos	23 mod 11 = 1
35	João Neves	35 mod 11 = 2
21	Igor Carlos	21 mod 11 = 10
11	Diogo Carvalho	11 mod 11 = 0
71	Cristiane Maria	71 mod 11 = 5
32	Mario Teles	32 mod 11 = 10
16	José Santos	16 mod 11 = 5

Consultar(chave){

procura por um elemento com a chave

na lista T[*h*(chave)]

}

TABELA HASH

A estrutura de dados Tabela Hash ou Tabela de Dispersão é responsável por acelerar muitos algoritmos que envolvem, inserção, consulta e remoção de dados em uma tabela.

Programa usando uma tabela Hash

1. Ler os dados de um arquivo texto com o seguinte layout: matrícula; nome completo;
2. Criar uma tabela hash por encadeamento de objetos com o uso da chave matrícula;
3. Definir as seguintes funcionalidade: consulta, inserção e remoção.
4. Mostrar os dados na grid em conformidade com a posição deste dentro da tabela hash e os elementos com colisão.

TABELA HASH

REFERÊNCIAS

Ascencio, Ana Fernanda Gomes, **Estruturas de dados : algoritmos, análise da complexidade e implementações em JAVA e C/C++** – São Paulo : Pearson Prentice Hall, 2010.

Goodrich, Michael T., **Estruturas de dados e algoritmos em Java [recurso eletrônico],** 5. ed. – Dados eletrônicos. – Porto Alegre : Bookman, 2013.

Szwarcfiter, Jayme Luiz, **Estruturas de dados e seus algoritmos,** 3.ed. [Reimpr.]. - Rio de Janeiro : LTC, 2015.

https://www.ime.usp.br/~pf/algoritmos/aulas/hash.html

https://www.programiz.com/dsa/hash-table