

## Representação Tabela Hash

Gerado por Doxygen 1.12.0



<b>1 Pesquisa e Ordenação de Dados - Tabela Hash com Lista Encadeada</b>	<b>1</b>
1.1 Descrição do Projeto	1
1.1.1 Funcionalidades Principais	1
1.2 Estrutura do Projeto	1
1.3 Pré-requisitos	2
1.3.1 Instalação de Dependências	2
1.4 Processo de compilação e execução	2
1.5 Gerando documentação	2
1.5.1 Gerando PDF's	2
<b>2 Índice dos Componentes</b>	<b>3</b>
2.1 Lista de Classes	3
<b>3 Índice dos Arquivos</b>	<b>5</b>
3.1 Lista de Arquivos	5
<b>4 Classes</b>	<b>7</b>
4.1 Referência da Estrutura hash	7
4.1.1 Descrição detalhada	7
4.2 Referência da Estrutura lista	7
4.2.1 Descrição detalhada	8
4.3 Referência da Estrutura tlist	8
4.3.1 Descrição detalhada	8
<b>5 Arquivos</b>	<b>9</b>
5.1 Referência do Arquivo app/main.c	9
5.1.1 Descrição detalhada	9
5.2 Referência do Arquivo include/tabela_hash.h	9
5.2.1 Descrição detalhada	10
5.2.2 Funções	10
5.2.2.1 criar_tabela()	10
5.2.2.2 criar_tlist()	10
5.2.2.3 exibir_hash()	11
5.2.2.4 inserir_hash()	11
5.2.2.5 liberar_tabela()	12
5.3 tabela_hash.h	12
5.4 Referência do Arquivo src/tabela_hash.c	13
5.4.1 Descrição detalhada	14
5.4.2 Funções	14
5.4.2.1 chaveDivisao()	14
5.4.2.2 criar_tabela()	14
5.4.2.3 criar_tlist()	15
5.4.2.4 exibir_hash()	15
5.4.2.5 extrair_valor()	15

5.4.2.6 insere_tlist()	15
5.4.2.7 inserir_hash()	16
5.4.2.8 liberar_tabela()	16
5.4.2.9 limpar_lista()	16
5.4.2.10 pesquisa_hash()	16
5.4.2.11 potencia()	17

<b>Index</b>	<b>19</b>
--------------	-----------

## Chapter 1

# Pesquisa e Ordenação de Dados - Tabela Hash com Lista Encadeada

Implementação de uma tabela hash com listas encadeadas para operações eficientes de inserção e busca de valores inteiros. Este projeto foi desenvolvido em C e tem como objetivo estudar e demonstrar o uso da estruturas de dados hash para otimização de armazenamento e busca de dados.

### 1.1 Descrição do Projeto

Esta aplicação em C implementa uma estrutura de dados de tabela hash utilizando listas encadeadas como método de resolução de colisões. Os dados são lidos de um arquivo, armazenados na tabela hash e organizados em listas encadeadas nas posições correspondentes.

#### 1.1.1 Funcionalidades Principais

- Inserção de valores inteiros em uma tabela hash.
- Conversão de caracteres lidos de arquivo em valores inteiros.
- Exibição de conteúdo da tabela hash.
- Liberação de memória alocada para a estrutura de dados.

### 1.2 Estrutura do Projeto

- `src/`: Contém os arquivos de código-fonte de implementação
- `include/`: Contém os arquivos de cabeçalho do projeto
- `app/`: Arquivo principal para execução do programa
- `docs/`: Diretório para a documentação do projeto gerada pelo Doxygen
- `README.md`: Arquivo de introdução ao projeto

## 1.3 Pré-requisitos

Para compilar e rodar o projeto, é necessário ter:

- **GCC** ou outro compilador C instalado.
- **Doxygen** (para gerar a documentação).

### 1.3.1 Instalação de Dependências

No Ubuntu/Debian:

```
1$ sudo apt update
1$ sudo apt install build-essential doxygen valgrind texlive-minimal
```

## 1.4 Processo de compilação e execução

O processo de compilação pode ser realizado seguindo os passos:

```
1$ cd Trabalho_2
1$ make
1$ make run
```

Existe uma target que pode ser utilizada para analisar o uso de memória:

```
1$ make valgrind
```

## 1.5 Gerando documentação

A ferramenta utilizada para gerar esta documentação foi o **Doxygen**. O passo-a-passo a seguir detalha o procedimento:

Primeiro é necessário gerar o arquivo de configuração:

```
1$ doxygen -g
```

Após gerado, alguns parâmetros básicos para edição:

- **INPUT:** Incluir os diretórios que contém os códigos que deseja gerar a documentação
- **EXCLUDE:** Excluir os diretórios aqui listados para não incluir na documentação

Após ajustados os parâmetros desejados, basta gerar a documentação de fato:

```
1$ doxygen Doxyfile
```

### 1.5.1 Gerando PDF's

Primeiramente precisa gerar a documentação acima, porém ajustando o parâmetro `GENERATE_LATEX = YES`. Após, acessar o diretório e rodar os seguintes comandos algumas vezes:

```
lpdflatex refman.tex
lmakeindex refman.idx
lpdflatex refman.tex
lpdflatex refman.tex
```

## Chapter 2

# Índice dos Componentes

### 2.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

<a href="#">hash</a>	Estrutura para representar a tabela hash . . . . .	<a href="#">7</a>
<a href="#">lista</a>	Estrutura temporária que será utilizada na extração dos números dos arquivos . . . . .	<a href="#">7</a>
<a href="#">tlist</a>	Estrutura para representar o nó dos valores lidos do arquivo . . . . .	<a href="#">8</a>





## Chapter 3

# Índice dos Arquivos

### 3.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

app/ <a href="#">main.c</a>	Aplicação básica com a finalidade de apresentar o funcionamento da tabela hash . . . . .	9
include/ <a href="#">tabela_hash.h</a>	Declaração das funções e estruturas utilizadas para manipular a tabela hash . . . . .	9
src/ <a href="#">tabela_hash.c</a>	Implementação de uma tabela de hash onde as colisões são tratadas através de encadeamento separado . . . . .	13



## Chapter 4

# Classes

### 4.1 Referência da Estrutura hash

Estrutura para representar a tabela hash.

```
#include <tabela_hash.h>
```

#### Atributos Públicos

- int **quantidade**  
*Quantidade de elementos que estarão na tabela hash.*
- int **tamanho\_tabela**  
*Quantidade buckets que a tabela hash terá*
- [Tlist](#) \*\* **itens**

#### 4.1.1 Descrição detalhada

Estrutura para representar a tabela hash.

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- [src/tabela\\_hash.c](#)

### 4.2 Referência da Estrutura lista

Estrutura temporária que será utilizada na extração dos números dos arquivos.

```
#include <tabela_hash.h>
```

### Atributos Públicos

- char **casas** [[MAX\\_CASAS](#)]  
*Array que armazena os dígitos dos elementos extraídos do arquivo.*
- int **quantidade**  
*Quantidade de casas decimais que cada valor terá*

#### 4.2.1 Descrição detalhada

Estrutura temporária que será utilizada na extração dos números dos arquivos.

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- [src/tabela\\_hash.c](#)

### 4.3 Referência da Estrutura tlist

Estrutura para representar o nó dos valores lidos do arquivo.

```
#include <tabela_hash.h>
```

### Atributos Públicos

- int **valor**  
*Representa o nó que será armazenado no tabela hash.*
- struct [tlist](#) \* **next**  
*Ponteiro para o próximo nó*

#### 4.3.1 Descrição detalhada

Estrutura para representar o nó dos valores lidos do arquivo.

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- [src/tabela\\_hash.c](#)

## Chapter 5

# Arquivos

### 5.1 Referência do Arquivo app/main.c

Aplicação básica com a finalidade de apresentar o funcionamento da tabela hash.

```
#include <stdio.h>
#include "tabela_hash.h"
```

#### Definições e Macros

- `#define TAM_TABELA 23`  
*Quantidade de índices que a tabela hash terá disponível.*

#### Funções

- `int main (void)`

#### 5.1.1 Descrição detalhada

Aplicação básica com a finalidade de apresentar o funcionamento da tabela hash.

Autor

[emerson.comar@estudante.uff.br](mailto:emerson.comar@estudante.uff.br)

### 5.2 Referência do Arquivo include/tabela\_hash.h

Declaração das funções e estruturas utilizadas para manipular a tabela hash.

## Definições de Tipos

- typedef struct [tlist](#) **Tlist**
- typedef struct [hash](#) **Hash**
- typedef struct [lista](#) **Lista**

## Funções

- [Hash](#) \* [criar\\_tabela](#) (int tamanho\_tabela)  
*Aloca uma tabela hash com o tamanho especificado.*
- void [liberar\\_tabela](#) ([Hash](#) \*tabela\_hash)  
*Libera a memória alocada para a tabela hash.*
- void [inserir\\_hash](#) ([Hash](#) \*tabela\_hash, [Tlist](#) \*lista\_tlist)  
*Transfere os valores da lista encadeada para a tabela hash.*
- void [exibir\\_hash](#) ([Hash](#) \*tabela\_hash)  
*Exibe os índices que não estão apontados para nulo.*
- [Tlist](#) \* [criar\\_tlist](#) (FILE \*arquivo)  
*Irá extrair os valores numéricos presentes no arquivo e montar uma lista encadeada do tipo Tlist contendo estes valores extraídos.*

### 5.2.1 Descrição detalhada

Declaração das funções e estruturas utilizadas para manipular a tabela hash.

Autor

[emerson.comar@estudante.uff.br](mailto:emerson.comar@estudante.uff.br)

### 5.2.2 Funções

#### 5.2.2.1 criar\_tabela()

```
Hash * criar_tabela (
    int tamanho_tabela)
```

Aloca uma tabela hash com o tamanho especificado.

Parâmetros

<i>tamanho_tabela</i>	Quantidade de index que a tabela hash terá
-----------------------	--

Retorna

A tabela alocada, ou NULL caso falhar

#### 5.2.2.2 criar\_tlist()

```
Tlist * criar_tlist (
    FILE * arquivo)
```

Irá extrair os valores numéricos presentes no arquivo e montar uma lista encadeada do tipo Tlist contendo estes valores extraídos.

**Parâmetros**

<code>arquivo</code>	o arquivo na qual os valores numéricos serão extraídos
----------------------	--

**Retorna**

Lista encadeada com os valores extraídos dos arquivos

- Lê cada caractere do arquivo até o final do arquivo, representado pelo EOF
- Caso o valor seja diferente de `;` o valor será armazenado em uma posição no array e o contador `quantidade` será incrementado
- Ao encontrar o valor `;` a função `extrair_valor` será chamada e, após, o valor será inserido na lista encadeada.

**Observação**

A função assume que:

- O arquivo estará corretamente formatado com apenas `;` e valores numéricos.
- Os números não tenham mais do que 10 casas decimais.

**Aviso**

A memória alocada precisa ser liberada antes da finalização.

**5.2.2.3 `exibir_hash()`**

```
void exibir_hash (  
    Hash * tabela_hash)
```

Exibe os índices que não estão apontados para nulo.

**Parâmetros**

<code>tabela_hash</code>	Estutura na qual os índices serão listados
--------------------------	--

Percorre os itens da tabela hash que não estejam apontando para NULL e percorre a lista encadeada, exibindo os valores contidos

**5.2.2.4 `inserir_hash()`**

```
void inserir_hash (  
    Hash * tabela_hash,  
    Tlist * lista_tlist)
```

Transfere os valores da lista encadeada para a tabela hash.

**Parâmetros**

<i>tabela_hash</i>	A tabela hash que irá anexar os dados
<i>lista_tlist</i>	A lista que contém os dados que serão transpostos

Irá percorrer a lista encadeada, criando um node novo e calculando a posição na tabela hash referente. Após, será inserido na tabela. Para evitar duplicata e aumentar a complexidade de busca e inserção, não será incluídos valores duplicados na tabela hash.

**5.2.2.5 liberar\_tabela()**

```
void liberar_tabela (  
    Hash * tabela_hash)
```

Libera a memória alocada para a tabela hash.

**Parâmetros**

<i>tabela_hash</i>	A tabela hash que será desalocada
--------------------	-----------------------------------

Libera a memória alocada para a tabela hash.

**Parâmetros**

<i>tabela_hash</i>	Tabela hash que será liberada
--------------------	-------------------------------

**5.3 tabela\_hash.h**

[Ir para a documentação desse arquivo.](#)

```
100001  
100004 #ifndef TABELA_HASH_H  
100005 #define TABELA_HASH_H  
100006  
100009 typedef struct tlist Tlist;  
100010  
100013 typedef struct hash Hash;  
100014  
100017 typedef struct lista Lista;  
100018  
100022 Hash *criar_tabela(int tamanho_tabela);  
100023  
100024  
100027 void liberar_tabela(Hash *tabela_hash);  
100028  
100032 void inserir_hash(Hash *tabela_hash, Tlist *lista_tlist);  
100033  
100034  
100037 void exibir_hash(Hash *tabela_hash);  
100038  
100039  
100043 Tlist *criar_tlist(FILE *arquivo);  
100044  
100045 #endif
```



## 5.4 Referência do Arquivo src/tabela\_hash.c

Implementação de uma tabela de hash onde as colisões são tratadas através de encadeamento separado.

```
#include <stdio.h>
#include <stdlib.h>
#include "tabela_hash.h"
```

### Componentes

- struct [tlist](#)  
*Estrutura para representar o nó dos valores lidos do arquivo.*
- struct [hash](#)  
*Estrutura para representar a tabela hash.*
- struct [lista](#)  
*Estrutura temporária que será utilizada na extração dos números dos arquivos.*

### Definições e Macros

- #define [MAX\\_CASAS](#) 10  
*Número máximo de casas decimais na qual os números extraídos do arquivo poderão ter.*

### Funções

- [Hash](#) \* [criar\\_tabela](#) (int tamanho\_tabela)  
*Aloca uma tabela hash com o tamanho especificado.*
- void [limpar\\_lista](#) ([Tlist](#) \*lista)  
*Liberação da memória alocada referente à lista encadeada.*
- void [liberar\\_tabela](#) ([Hash](#) \*tabela\_hash)  
*Liberação da memória alocada da tabela hash.*
- int [chaveDivisao](#) (int chave, int tamanho\_tabela)  
*Função hash responsável em determinar qual o índice o valor irá ocupar na tabela hash.*
- int [pesquisa\\_hash](#) ([Hash](#) \*tabela\_hash, int chave, int posicao)  
*Pesquisa se um determinado valor está presente na tabela hash.*
- void [inserir\\_hash](#) ([Hash](#) \*tabela\_hash, [Tlist](#) \*lista\_tlist)  
*Transfere os valores da lista encadeada para a tabela hash.*
- void [exibir\\_hash](#) ([Hash](#) \*tabela\_hash)  
*Exibe os índices que não estão apontados para nulo.*
- int [potencia](#) (int base, int expoente)  
*Realiza a operação de potenciação.*
- int [extrair\\_valor](#) ([Lista](#) \*lista\_valores)  
*Irá "converter" as casas decimais no valor numérico correto.*
- void [insere\\_tlist](#) ([Tlist](#) \*\*lista\_tlist, int valor)  
*Cria um novo nó tlist com o valor extraído do arquivo e insere na lista ao final da lista encadeada.*
- [Tlist](#) \* [criar\\_tlist](#) (FILE \*arquivo)  
*Irá extrair os valores numéricos presentes no arquivo e montar uma lista encadeada do tipo Tlist contendo estes valores extraídos.*

### 5.4.1 Descrição detalhada

Implementação de uma tabela de hash onde as colisões são tratadas através de encadeamento separado.

Autor

[emerson.comar@estudante.uff.br](mailto:emerson.comar@estudante.uff.br)

### 5.4.2 Funções

#### 5.4.2.1 chaveDivisao()

```
int chaveDivisao (  
    int chave,  
    int tamanho_tabela)
```

Função hash responsável em determinar qual o índice o valor irá ocupar na tabela hash.

Parâmetros

<i>chave</i>	O valor que deseja inserir na tabela hash
<i>tamanho_tabela</i>	A quantidade de índices que a tabela hash terá

Retorna

O valor do índice na qual a chave será armazenada

#### 5.4.2.2 criar\_tabela()

```
Hash * criar_tabela (  
    int tamanho_tabela)
```

Aloca uma tabela hash com o tamanho especificado.

Parâmetros

<i>tamanho_tabela</i>	Quantidade de index que a tabela hash terá
-----------------------	--

Retorna

A tabela alocada, ou NULL caso falhar

### 5.4.2.3 criar\_tlist()

```
Tlist * criar_tlist (
    FILE * arquivo)
```

Irá extrair os valores numéricos presentes no arquivo e montar uma lista encadeada do tipo Tlist contendo estes valores extraídos.

- Lê cada caractere do arquivo até o final do arquivo, representado pelo EOF
- Caso o valor seja diferente de ; o valor será armazenado em uma posição no array e o contador quantidade será incrementado
- Ao encontrar o valor ; a função extrair\_valor será chamada e, após, o valor será inserido na lista encadeada.

#### Observação

A função assume que:

- O arquivo estará corretamente formatado com apenas ; e valores numéricos.
- Os números não tenham mais do que 10 casas decimais.

#### Aviso

A memória alocada precisa ser liberada antes da finalização.

### 5.4.2.4 exibir\_hash()

```
void exibir_hash (
    Hash * tabela_hash)
```

Exibe os índices que não estão apontados para nulo.

Percorre os itens da tabela hash que não estejam apontando para NULL e percorre a lista encadeada, exibindo os valores contidos

### 5.4.2.5 extrair\_valor()

```
int extrair_valor (
    Lista * lista_valores)
```

Irá "converter" as casas decimais no valor numérico correto.

A função utiliza o Princípio Fundamental da Contagem para realizar a conversão

#### Parâmetros

<i>lista_valores</i>	Estrutura de dados que conterà as casas decimais que será transformada no valor numérico
----------------------	--

#### Retorna

O valor numérico extraído

### 5.4.2.6 insere\_tlist()

```
void insere_tlist (
    Tlist ** lista_tlist,
    int valor)
```

Cria um novo nó tlist com o valor extraído do arquivo e insere na lista ao final da lista encadeada.

**Parâmetros**

<i>lista_tlist</i>	Endereço de memória da primeira posição da lista encadeada
<i>valor</i>	Valor extraído do arquivo na qual será o valor do nó criado

**5.4.2.7 inserir\_hash()**

```
void inserir_hash (  
    Hash * tabela_hash,  
    Tlist * lista_tlist)
```

Transfere os valores da lista encadeada para a tabela hash.

Irá percorrer a lista encadeada, criando um node novo e calculando a posição na tabela hash referente. Após, será inserido na tabela. Para evitar duplicata e aumentar a complexidade de busca e inserção, não será incluídos valores duplicados na tabela hash.

**5.4.2.8 liberar\_tabela()**

```
void liberar_tabela (  
    Hash * tabela_hash)
```

Liberação da memória alocada da tabela hash.

Libera a memória alocada para a tabela hash.

**Parâmetros**

<i>tabela_hash</i>	Tabela hash que será liberada
--------------------	-------------------------------

**5.4.2.9 limpar\_lista()**

```
void limpar_lista (  
    Tlist * lista)
```

Liberação da memória alocada referente à lista encadeada.

**Parâmetros**

<i>lista</i>	Lista encadeada que será liberada
--------------	-----------------------------------

**5.4.2.10 pesquisa\_hash()**

```
int pesquisa_hash (  
    Hash * tabela_hash,  
    int chave,  
    int posicao)
```

Pesquisa se um determinado valor está presente na tabela hash.

**Parâmetros**

<i>tabela_hash</i>	Tabela hash que será pesquisada
<i>chave</i>	Valor que deseja procurar na tabela hash
<i>posicao</i>	Posição em que o valor deverá estar

**Retorna**

Retorna 0 caso encontrado a chave e 1 caso não encontrado

**5.4.2.11 potencia()**

```
int potencia (  
    int base,  
    int expoente)
```

Realiza a operação de potenciação.

**Parâmetros**

<i>base</i>	A base da potência
<i>expoente</i>	Expotente da potência

**Retorna**

O valor calculado da base elevado à potência



# Index

app/main.c, [9](#)

chaveDivisao  
    [tabela\\_hash.c, 14](#)

criar\_tabela  
    [tabela\\_hash.c, 14](#)  
    [tabela\\_hash.h, 10](#)

criar\_tlist  
    [tabela\\_hash.c, 14](#)  
    [tabela\\_hash.h, 10](#)

exibir\_hash  
    [tabela\\_hash.c, 15](#)  
    [tabela\\_hash.h, 11](#)

extrair\_valor  
    [tabela\\_hash.c, 15](#)

hash, [7](#)

include/tabela\_hash.h, [9](#), [12](#)

insere\_tlist  
    [tabela\\_hash.c, 15](#)

inserir\_hash  
    [tabela\\_hash.c, 16](#)  
    [tabela\\_hash.h, 11](#)

liberar\_tabela  
    [tabela\\_hash.c, 16](#)  
    [tabela\\_hash.h, 12](#)

limpar\_lista  
    [tabela\\_hash.c, 16](#)

lista, [7](#)

Pesquisa e Ordenação de Dados - Tabela Hash com  
    Lista Encadeada, [1](#)

pesquisa\_hash  
    [tabela\\_hash.c, 16](#)

potencia  
    [tabela\\_hash.c, 17](#)

src/tabela\_hash.c, [13](#)

tabela\_hash.c  
    [chaveDivisao, 14](#)  
    [criar\\_tabela, 14](#)  
    [criar\\_tlist, 14](#)  
    [exibir\\_hash, 15](#)  
    [extrair\\_valor, 15](#)  
    [insere\\_tlist, 15](#)  
    [inserir\\_hash, 16](#)  
    [liberar\\_tabela, 16](#)  
    [limpar\\_lista, 16](#)  
    [pesquisa\\_hash, 16](#)  
    [potencia, 17](#)

tabela\_hash.h  
    [criar\\_tabela, 10](#)  
    [criar\\_tlist, 10](#)  
    [exibir\\_hash, 11](#)  
    [inserir\\_hash, 11](#)  
    [liberar\\_tabela, 12](#)

tlist, [8](#)