

Nome: Marcos Antonio Carlos Garcia

RA: 840121

## TRABALHO DE ESTRUTURA DE DADOS

---

```
int main() {

    Aluno* alunos[MAX_ALUNO];

    FILE* arq = ler_arquivo("dados.txt");

    if (arq == NULL) printf("Arquivo nao existe.\n");

    ler_conteudo_arquivo(arq, alunos);

    Hash* hash = criar_tabela_hash();

    for (int i = 0; i < MAX_ALUNO; i++) {
        add(hash, alunos[i]);
    }

    int acumulador = 0;
    int ociosa = 0;
    for (int i = 0; i < MAX_LISTA; i++) {
        Aluno* aluno = hash->lista[i];
        if (aluno == NULL) {
            ociosa++;
        }
        int count = 0;
        while(aluno) {
            count++;
            aluno = aluno->prox;
        }
        printf("Quantidade de elementos na posicao %d da hash-table: %d\n", i, count);
        acumulador += count;
    }
    printf("-----\n");
    printf("Media de elementos nas listas: %d\n", acumulador/MAX_LISTA);
    printf("-----\n");
    printf("Colisoes: %d\n", get_colisoes());
    printf("-----\n");
    printf("Posicoes ociosas na tabela hash: %d\n", ociosa);
    printf("-----\n");

    limpesa_de_memoria(hash, arq);

}
```

Começamos com a leitura do arquivo e a construção das estruturas com os dados e a alocação das estruturas em um vetor de mesmo tipo. Logo após isso fazemos a criação da estrutura "Hash" e a alocação dos dados nessa estrutura. Por fim temos uma ação para recuperar as informações sobre o processo.

---

### 1 Criação da estrutura Hash

```
// main.c
Hash* hash = criar_tabela_hash();
// =====

// hash_table.h
#define MAX_LISTA 1000

typedef struct {
    Aluno* lista[MAX_LISTA];
} Hash;

Hash* criar_tabela_hash();
int hash_code(int matricula);
void add(Hash* hash_table, Aluno* aluno);
int get_colisoes();
// =====

// hash_table.c
Hash* criar_tabela_hash() {
    Hash* hash = (Hash*) malloc(sizeof(Hash));
    if (hash == NULL) {
        printf("Erro ao criar a tabela Hash.\n\n");
        exit(0);
    }
    for (int i = 0; i < MAX_LISTA; i++) {
        hash->lista[i] = NULL;
    }
    return hash;
}
// =====
```

Temos a definição da estrutura e seus comportamentos no arquivo de cabeçalho. A criação se dá no método acima chamado "criar\_tabela\_hash", alocando espaço em memória, faz a

verificação se houve erro e alimenta os espaços da tabela com nulo. O tamanho é definido pela variável de pré processamento MAX\_LISTA, que está definida no documento de cabeçalho.

```
int hash_code(int matricula) {  
    return (matricula & 0x7FFFFFFF) % MAX_LISTA;  
}
```

Aqui temos a definição de como os identificadores da tabela hash serão gerados para cada elemento a ser inserido na estrutura.

```
void add(Hash* hash_table, Aluno* aluno) {
    int hash_aluno = hash_code(aluno->matricula);
    Aluno* aluno_alocado = hash_table->lista
[hash_aluno];
    if (aluno_alocado == NULL) {
        hash_table->lista[hash_aluno] = aluno;
    } else {
        colisoas++;
        while(aluno_alocado) {
            if (aluno_alocado->prox == NULL) {
                aluno_alocado->prox = aluno;
                break;
            }
            aluno_alocado = aluno_alocado->prox;
        }
    }
}
```

Esse comportamento aloca nossos dados em uma lista encadeada dinâmica. Primeiramente fazemos o calculo para criar o hash de identificação da estrutura, buscamos na tabela se há algum dado com o hash gerado. Não havendo fazemos a atribuição dos dados no índice correspondente. Mas se há dados para o mesmo índice, percorremos até o local em que deve ser feita a alocação dos novos dados.

```
int acumulador = 0;
int ociosa = 0;
for (int i = 0; i < MAX_LISTA; i++) {
    Aluno* aluno = hash->lista[i];
    if (aluno == NULL) {
        ociosa++;
    }
    int count = 0;
    while(aluno) {
        count++;
        aluno = aluno->prox;
    }
    printf("Quatidade de elementos na posicao %d da hash-table: %d\n", i, count);
    acumulador += count;
}
printf("-----\n");
printf("Media de elementos nas listas: %d\n", acumulador/MAX_LISTA);
printf("-----\n");
printf("Colisoes: %d\n", get_colisoes());
printf("-----\n");
printf("Posicoes ociosas na tabela hash: %d\n", ociosa);
printf("-----\n");
```

Aqui fazemos a leitura dos dados para fazer a aferição de algumas questões.

## Resultados

```
Quatidade de elementos na posicao 0 da hash-table: 259
Quatidade de elementos na posicao 1 da hash-table: 222
Quatidade de elementos na posicao 2 da hash-table: 238
Quatidade de elementos na posicao 3 da hash-table: 216
Quatidade de elementos na posicao 4 da hash-table: 230
Quatidade de elementos na posicao 5 da hash-table: 231
Quatidade de elementos na posicao 6 da hash-table: 250
Quatidade de elementos na posicao 7 da hash-table: 203
Quatidade de elementos na posicao 8 da hash-table: 232
Quatidade de elementos na posicao 9 da hash-table: 206
```

```
-----
Media de elementos nas listas: 228
```

```
-----
Colisoes: 2277
```

```
-----
Posicoes ociosas na tabela hash: 0
-----
```

Resultado para uma estrutura com 10 posições na tabela Hash. Teremos 10 listas, a média de elementos de cada lista será 228 com 2277 colisões e sem posições ociosas.

```
Quatidade de elementos na posicao 81 da hash-table: 29
Quatidade de elementos na posicao 82 da hash-table: 25
Quatidade de elementos na posicao 83 da hash-table: 29
Quatidade de elementos na posicao 84 da hash-table: 22
Quatidade de elementos na posicao 85 da hash-table: 27
Quatidade de elementos na posicao 86 da hash-table: 22
Quatidade de elementos na posicao 87 da hash-table: 22
Quatidade de elementos na posicao 88 da hash-table: 25
Quatidade de elementos na posicao 89 da hash-table: 16
Quatidade de elementos na posicao 90 da hash-table: 21
Quatidade de elementos na posicao 91 da hash-table: 22
Quatidade de elementos na posicao 92 da hash-table: 26
Quatidade de elementos na posicao 93 da hash-table: 22
Quatidade de elementos na posicao 94 da hash-table: 23
Quatidade de elementos na posicao 95 da hash-table: 13
Quatidade de elementos na posicao 96 da hash-table: 26
Quatidade de elementos na posicao 97 da hash-table: 23
Quatidade de elementos na posicao 98 da hash-table: 22
Quatidade de elementos na posicao 99 da hash-table: 19
```

```
-----
Media de elementos nas listas: 22
```

```
-----
Colisoes: 2187
```

```
-----
Posicoes ociosas na tabela hash: 0
-----
```

Resultado para uma estrutura com 100 posições na tabela Hash. Teremos 100 listas, a média de elementos de cada lista será 22 com 2187 colisões e sem posições ociosas.

```
Quatidade de elementos na posicao 981 da hash-table: 3
Quatidade de elementos na posicao 982 da hash-table: 2
Quatidade de elementos na posicao 983 da hash-table: 2
Quatidade de elementos na posicao 984 da hash-table: 1
Quatidade de elementos na posicao 985 da hash-table: 3
Quatidade de elementos na posicao 986 da hash-table: 2
Quatidade de elementos na posicao 987 da hash-table: 4
Quatidade de elementos na posicao 988 da hash-table: 6
Quatidade de elementos na posicao 989 da hash-table: 3
Quatidade de elementos na posicao 990 da hash-table: 3
Quatidade de elementos na posicao 991 da hash-table: 0
Quatidade de elementos na posicao 992 da hash-table: 1
Quatidade de elementos na posicao 993 da hash-table: 0
Quatidade de elementos na posicao 994 da hash-table: 1
Quatidade de elementos na posicao 995 da hash-table: 1
Quatidade de elementos na posicao 996 da hash-table: 2
Quatidade de elementos na posicao 997 da hash-table: 5
Quatidade de elementos na posicao 998 da hash-table: 4
Quatidade de elementos na posicao 999 da hash-table: 3
```

```
-----
Media de elementos nas listas: 2
```

```
-----
Colisoes: 1390
```

```
-----
Posicoes ociosas na tabela hash: 103
-----
```

Resultado para uma estrutura com 1.000 posições na tabela Hash. Teremos 1.000 listas, a média de elementos de cada lista será 2 com 1390 colisões com 103 posições ociosas.

```
Quatidade de elementos na posicao 9981 da hash-table: 0
Quatidade de elementos na posicao 9982 da hash-table: 0
Quatidade de elementos na posicao 9983 da hash-table: 1
Quatidade de elementos na posicao 9984 da hash-table: 0
Quatidade de elementos na posicao 9985 da hash-table: 0
Quatidade de elementos na posicao 9986 da hash-table: 0
Quatidade de elementos na posicao 9987 da hash-table: 1
Quatidade de elementos na posicao 9988 da hash-table: 0
Quatidade de elementos na posicao 9989 da hash-table: 0
Quatidade de elementos na posicao 9990 da hash-table: 0
Quatidade de elementos na posicao 9991 da hash-table: 0
Quatidade de elementos na posicao 9992 da hash-table: 0
Quatidade de elementos na posicao 9993 da hash-table: 0
Quatidade de elementos na posicao 9994 da hash-table: 1
Quatidade de elementos na posicao 9995 da hash-table: 0
Quatidade de elementos na posicao 9996 da hash-table: 0
Quatidade de elementos na posicao 9997 da hash-table: 1
Quatidade de elementos na posicao 9998 da hash-table: 0
Quatidade de elementos na posicao 9999 da hash-table: 0
```

```
-----
Media de elementos nas listas: 0
-----
```

```
Colisoes: 174
-----
```

```
Posicoes ociosas na tabela hash: 7887
-----
```

Resultado para uma estrutura com 10.000 posições na tabela Hash. Teremos 10.000 listas, a média de elementos de cada lista será 0 com 174 colisões com 7887 posições ociosas.

---