# Aula Prática 11 - Estruturas de Dados I (BCC202)

Marco Antonio M. Carvalho
Universidade Federal de Ouro Preto
Departamento de Computação

10 de agosto de 2021

## Instruções

- Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado;
- Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas;
- A avaliação considerará o tempo de execução e o percentual de respostas corretas;
- Eventualmente realizadas entrevistas sobre os estudos dirigidos para complementar a avaliação;
- Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada;
- Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software;
- Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota e frequência;
- Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos;
- Não serão considerados algoritmos parcialmente implementados.

# 1 Tabelas Hash

Crie um programa que, dadas a quantidade $n$ de endereços de uma tabela hash (enumerados de 0 a $n$-1) e uma sequência de chaves, calcula o endereço para inserções das chaves. Considere que para uma chave $x$ a função de dispersão é $h(x) = x \bmod n$ e que o tratamento de colisões é feito por encadeamento.

### Especificação da Entrada

A primeira linha contém um valor $n$ ($1 \leq n \leq 100$) que indica a quantidade de endereços na tabela seguido por um espaço e um valor $c$ arbitrário que indica a quantidade de chaves a serem armazenadas. A segunda linha contém cada uma das chaves, separadas por um espaço em branco.

### Especificação da Saída

A saída deverá refletir a tabela hash preenchida e deve ser impressa conforme o exemplo fornecidos abaixo, em que a quantidade de linhas de cada caso de teste é determinada pelo valor de $n$.

### Exemplo de Entrada

```
13 9
44 45 49 70 27 73 92 97 95
```

### Exemplo de Saída

```
0 -> \
1 -> 27 -> 92 -> \
2 -> \
3 -> \
4 -> 95 -> \
5 -> 44 -> 70 -> \
6 -> 45 -> 97 -> \
7 -> \
8 -> 73 -> \
9 -> \
10 -> 49 -> \
11 -> \
12 -> \
```

### Estrutura do código

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Uma tabela hash deve ser alocada dinamicamente e manipulada para determinação da resposta.

### Diretivas de Compilação

```
$ gcc hash.c -c
$ gcc principal.c -c
$ gcc hash.o principal.o -o programa
```