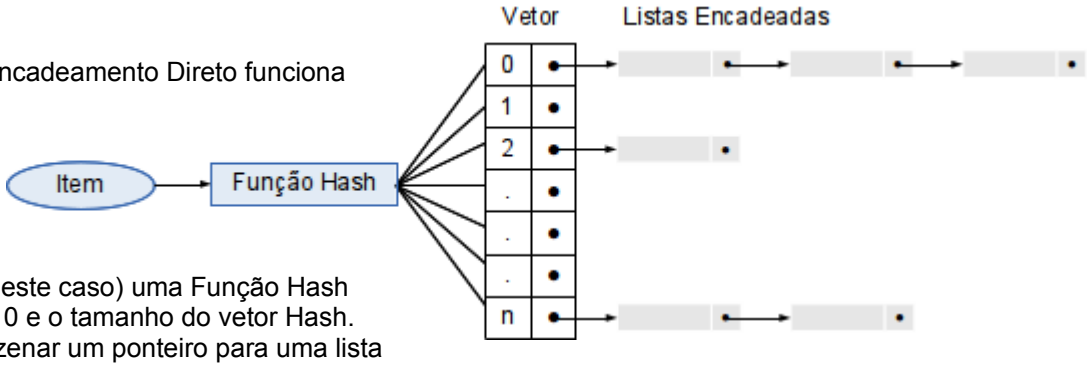
OBJETIVO

Relembrar os conceitos básicos de estruturas de dados e ponteiros obtidos nas disciplinas de AED1 e AED2 através da implementação de uma Tabela Hash com Encadeamento Direto que armazene as informações de uma pessoa (`char nome[51]; long long int cpf; int idade;`).

QUESTÃO ÚNICA

Uma Tabela Hash com Encadeamento Direto funciona como na figura abaixo.



Dado um item (pessoa, neste caso) uma Função Hash calcula um número entre 0 e o tamanho do vetor Hash. Este vetor hash irá armazenar um ponteiro para uma lista encadeada (similar à feita no 1o TP). Esta lista encadeada é necessária para lidar com as "colisões", que acontecem quando dois ou mais itens possuem o mesmo retorno da função Hash.

Sua nova missão, caso deseje aceitar, é implementar esta tabela hash onde os itens são "pessoas" e a **Função Hash** é dada pelo *resto da divisão entre o CPF da pessoa e o tamanho do vetor* da tabela hash. O tamanho do vetor será passado pela linha de comando (`argv[1]`). Nos casos de colisões, o item deverá ser inserido no início da lista encadeada.

Leia um arquivo de entrada cujo nome será passado também pela linha de comando (`argv[2]`). Cada linha deste arquivo terá a seguinte sintaxe:

```
<Nome>\t<CPF>\t<Idade>\n
```

onde o `<Nome>` é uma string (que pode conter espaços) de, no máximo, 50 caracteres, `<CPF>` é um inteiro longo duplo (`long long int`) – sem dígito e sem necessidade de validação – e `<Idade>` é um inteiro. Tudo separado por **tabs** (`\t`) e seguido de uma linha nova (`\n`). Para cada linha, adicione a pessoa na tabela hash. Após ler todo o arquivo de entrada, para cada elemento do Vetor da Tabela Hash, imprima a lista encadeada apontada por ele.

Exemplo de Entrada (arquivo_entrada.txt):

```
Fulando de Tal 1o       2183810669      26
Fulando de Tal 2o       308013959       68
Fulando de Tal 3o       1266626547      45
Fulando de Tal 4o       3152212616      12
Fulando de Tal 5o       706121745       38
Fulando de Tal 6o       133102833       94
Fulando de Tal 7o       1103026326      45
Fulando de Tal 8o       2061522446      42
Fulando de Tal 9o       311927579       22
Fulando de Tal 10o      162095725       82
```

Nota Importante: Se você copiar e colar o exemplo acima, substitua os **espaços** entre os valores por **tabs**. Ou, melhor, baixe o arquivo usando o link:
https://beans.icomp.ufam.edu.br/lpa/lab2_arquivo_entrada.txt

Saída Esperada para uma tabela Hash de tamanho 5 (./tabela_hash 5 arquivo_entrada.txt):

```
POSIÇÃO 0 DA TABELA HASH:
- Fulando de Tal 10o    162095725       82
- Fulando de Tal 5o     706121745       38
POSIÇÃO 1 DA TABELA HASH:
- Fulando de Tal 8o     2061522446      42
- Fulando de Tal 7o     1103026326      45
- Fulando de Tal 4o     3152212616      12
POSIÇÃO 2 DA TABELA HASH:
- Fulando de Tal 3o     1266626547      45
POSIÇÃO 3 DA TABELA HASH:
- Fulando de Tal 6o     133102833       94
POSIÇÃO 4 DA TABELA HASH:
- Fulando de Tal 9o     311927579       22
- Fulando de Tal 2o     308013959       68
- Fulando de Tal 1o     2183810669      26
```

Dicas:

Sugestão de funções a serem implementadas:

`bool lista_pessoas_adicionar(pessoa_t *pessoa, lista_pessoas_t **lista);`
→ Adiciona uma pessoa a uma lista encadeada (similar ao TP1)

`void lista_pessoas_listar(lista_pessoas_t *lista);`
→ Imprime as pessoas em uma lista encadeada (similar ao TP1, mas no formato especificado)

`tabela_hash_t tabela_hash_pessoas_criar();`
→ Aloca memória para um vetor de ponteiros para listas encadeadas e faz cada ponteiro de lista (elemento do vetor) apontar para nulo.
→ `tabela_hash_t` pode ser definido como: `typedef lista_pessoas_t** tabela_hash_t;` O motivo de ser um ponteiro para ponteiros é que alocaremos memória para um vetor de ponteiros para listas (topos das listas).

`int tabela_hash_pessoas_funcao(pessoa_t *pessoa);`
→ Calcula a Função Hash de uma pessoa (`pessoa->cpf % tabela_hash_tam`).

`bool tabela_hash_pessoas_adicionar(pessoa_t *pessoa, tabela_hash_t tabela_hash);`
→ Adiciona a pessoa na lista encadeada localizada na posição do vetor especificado pela função acima.

`void tabela_hash_pessoas_listar(tabela_hash_t tabela_hash);`
→ Para cada posição do vetor, executa a função `lista_pessoas_listar`.

`int main(int argc, char **argv);`
→ Lê o tamanho da Tabela Hash especificado na linha de comando (`argv[1]`) e salva em uma variável global. Use o `sscanf` para converter o `argv[1]` para inteiro.
→ Cria a Tabela Hash;
→ Lê o arquivo de entrada (argv[2]) e, para cada linha: aloca memória para uma nova pessoa, seta os valores e a adiciona na Tabela Hash. Use o fscanf para ler a linha. Exemplo:
`fscanf(arq_in, "%50[^\t]\t%lld\t%d\n", &p->nome, &p->cpf, &p->idade);`
→ Imprime a tabela

ENTREGA DO LABORATÓRIO

O laboratório é presencial e deve ser entregue durante o horário da aula. Para entregar, envie o código-fonte para horacio@icomp.ufam.edu.br com o assunto “Entrega do 2o Laboratório de LPA”.

Depois de entregue, você pode sair ou, se desejar, pode ajudar algum(a) colega a terminar o trabalho dele(a), desde que não haja cópia do seu código (plágio).