
Instituto Federal de Educação, Ciência e Tecnologia de Brasília – Câmpus Taguatinga
Ciência da Computação – Estruturas de Dados e Algoritmos
Lista de Exercícios – Listas
Prof. Daniel Saad Nogueira Nunes

Aluno: _____
Matrícula: _____

## Exercício 1

Implemente uma lista encadeada que trabalhe com qualquer tipo (seja genérica). Sua lista deve suportar as operações:

- ACCESS_HEAD: acessa o elemento da cabeça da lista.
- ACCESS_TAIL: acessa o elemento da cauda da lista.
- ACCESS: acessa um elemento qualquer da lista.
- REMOVE_HEAD: remove a cabeça da lista.
- REMOVE_TAIL: remove a cauda da lista.
- REMOVE: remove um elemento qualquer da lista.
- PREPEND: insere um elemento na cabeça da lista.
- APPEND: insere um elemento na cauda da lista.
- INSERT: insere um elemento em uma posição qualquer da lista.
- SIZE: retorna o tamanho da lista.
- LIST_EMPTY: retorna verdadeiro se a lista está vazia e falso caso contrário.

## Exercício 2

Faça um programa que recursivamente remova todos os elementos nulos de uma lista de inteiros.

## Exercício 3

Implemente a concatenação de listas encadeadas *in-place* (sem copiar os dados). Sua função deve possuir a assinatura:

```
list_t* concat(list_t* list_1,list_t* list_2);.
```

No caso da chamada acima, a lista `list_1` seria concatenada com a lista `list_2` formando uma lista maior ainda que seria retornada pela função.

## Exercício 4

Implemente a reversão de listas encadeadas. Sua função deve possuir a assinatura

```
list_t* reverse(list_t* list);
```

## Exercício 5

Implemente a função `drop` que recebe um inteiro $k$ e retira os $k$ primeiros elementos desta lista. Caso $k \geq n$, onde $n$ é o tamanho da lista, a função deverá retornar uma lista vazia. Esta função deverá possuir a seguinte assinatura:

```
list_t* drop(list_t* list,int k)
```

## Exercício 6

Implemente a função `take` que recebe um inteiro $k$ e retorna uma lista com os $k$ primeiros elementos Caso $k = 0$, a lista vazia deverá ser retornada. Esta função deverá possuir a seguinte assinatura:

```
list_t* take(list_t* list,int k)
```

## Exercício 7

Desenvolva o Quicksort para listas encadeadas. Note que o quicksort pode ser implementado através de duas aplicações da função `filter` e a utilização da função `cat`.

## Exercício 8

Implemente a operação FORALL para listas encadeadas. Esta operação deve receber uma função como parâmetro, através de um ponteiro por função, varrer a lista encadeada e aplicar a função em cada elemento da lista. Por exemplo, é possível implementar a função FORALL para que ela incremente todos os elementos da lista ao passar a seguinte função como parâmetro:

```
void list_increment(void* data){
    int* aux = data;
    (*aux)++;
}
```

Já a seguinte função ao ser passada como parâmetro para a operação FORALL divide todos os números pares da lista por 3:

```
void divide_even_by_3(void* data){
    int* aux = data
    if((*aux)%2==0){
       (*aux) = (*aux)/3;
    }
}
```

A função deverá possuir a seguinte assinatura:

```
void forall(list_t* list, void (*fn) (void* data));
```

## Exercício 9

Implemente a operação FILTER, que recebe uma lista e um predicado e retorna uma lista só com os elementos que possuem a propriedade dada pelo predicado. Um predicado nada mais é que uma função que retorna **verdadeiro** se o elemento tem uma determinada propriedade e **falso** caso contrário. Por exemplo, se a operaçãoFILTER atuasse sobre o seguinte predicado:

```
int is_zero(void* data){
    int* v = data;
    if(*v == 0){
        return 1;
    }
    return 0;
}
```

Ela eliminaria todos os elementos não nulos da lista. A sua função deverá possuir a seguinte assinatura:

```
void filter(list_t* list, int (*fn)(void* data));
```

## Exercício 10

Refaça os exercícios anteriores para listas duplamente encadeadas.

## Exercício 11

Uma lista circular é aquela em que a referência do último elemento aponta para a cabeça. Implemente uma lista circular dupla e simples com as operações básicas de lista.

## Exercício 12

**(Par ou ímpar americano)**

Faça um programa que leia um inteiro $n$ e em seguida $n$ nomes com até 50 caracteres. Estes nomes representarão uma roda de amigos no sentido horário. A seguir o seu programa deverá ler $n-1$ números inteiros positivos $a_0, a_1, \ldots, a_{n-2}$, onde cada número $a_i$ indica que na $i$-ésima iteração, deveremos escolher a $a_i$-ésima pessoa à direita da pessoa inicial e eliminá-la da roda. Na primeira iteração a pessoa inicial é aquela que foi lida primeiramente, nas demais iterações a pessoa inicial é aquela à esquerda da que foi eliminada. Seu programa deverá informar a cada iteração qual pessoa foi eliminada e a pessoa restante do jogo.

Exemplo:

- Entrada:

  ```
  5
  Hericlapton Epaminondas Godofreda Astrogildo Holofontina
  10 2 1 3
  ```

- Saída:

```
Hericlapton foi eliminado(a).
Astrogildo foi eliminado(a).
Epaminondas foi eliminado(a).
Holofontina foi eliminado(a).
Godofreda venceu.
```

**Dica: simule com uma lista circular**

## Exercício 13

Compare as estruturas de dados Lista Encadeada e Lista Duplamente Encadeada. Quais as vantagens e desvantagens de cada uma?

## Exercício 14

É possível implementar uma busca binária em uma lista encadeada ou duplamente encadeada? Justifique a sua resposta.