

Listas ligadas com ligações simples

A. Considere as seguintes definições dos tipos de dados **INFO**, **NodoL** e **PNodoL** (apontamentos das aulas teóricas):

```
typedef struct {  
    int numAluno;           // número de aluno (chave): { 70000, ..., 75000 }  
    float notasMTP[2];      // notas dos mini-testes práticos: [0.0, 2.0]  
    float notasTE[2];       // notas dos testes escritos: [0.0, 8.0]  
    int notaFinal;          // nota final (valor arredondado da soma das 4 notas): { 0, ..., 20 }  
} INFO;  
  
struct NodoL {  
    INFO Elemento;  
    struct NodoL *Prox;  
};  
  
typedef struct NodoL *PNodoL;
```

Cada elemento (registo) do tipo de dados **INFO** corresponde aos dados sobre a avaliação de um aluno à disciplina de AED, durante a época de Aprendizagem. O campo **numAluno** é a chave (valor único para cada elemento do tipo **INFO**).

Descarregar as seguintes bibliotecas (**Folhas práticas ---> Bibliotecas e Exercícios das folhas práticas ---> Listas ligadas simples**):

Aleatorio.h ---> operações para gerar números aleatoriamente
EADLista.h ---> operações básicas sobre a EAD **Lista**
OperacoesBasicasExA.h ---> operações básicas sobre o tipo de dados **INFO**
OperacoesExA.h ---> operações a implementar relativas aos exercícios do **grupo A**
MainExA.c ---> programa principal relativo aos exercícios do **grupo A**

Resolva os exercícios propostos utilizando as operações contidas nas bibliotecas referidas e acrescentado as novas operações à biblioteca "OperacoesExA.h".

1. Implementar uma função que,
 - receba um valor inteiro positivo **N**
 - crie e devolva uma lista com **N** elementos do tipo **INFO** gerados aleatoriamente (usar o gerador de números da biblioteca **Aleatorio**).
2. Elaborar um programa que realize as seguintes ações (pela ordem indicada):
 - defina o tamanho de uma lista (**TAM**), com um valor gerado aleatoriamente entre 0 e 15
 - use a função anterior para criar uma lista com **TAM** elementos do tipo **INFO**
 - mostre a lista criada, em que os elementos são apresentados **(a)** da cabeça/início para a cauda/fim e **(b)** da cauda/fim para a cabeça/início.

As funções que forem implementadas a partir daqui, devem ser testadas no programa (acrescentar).

3. Implementar uma função iterativa que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o **tamanho** (número de elementos) da lista.
4. Implementar uma função recursiva que
 - receba uma lista com elementos do tipo **INFO** e um valor inteiro **N**,
 - devolva a quantidade de elementos da lista com valor no campo **notaFinal** igual a **N**.
5. Implementar uma função iterativa que
 - receba uma lista com elementos do tipo **INFO** e um valor inteiro **N**,
 - devolva o número de elementos da lista com valor no campo **notaFinal** maior ou igual a **N**.
6. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o **maior** valor no campo **notaFinal** da lista.
7. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o aluno com melhor nota nas frequências (o elemento da lista com o maior valor na soma dos valores dos campos **notasTE**).
8. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o aluno com pior nota nos mini-testes práticos (o elemento da lista com o menor valor num dos campos **notasMTP**).
9. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **K**,
 - devolva o menor elemento da lista com valor no campo **notaFinal** maior ou igual a **K**.
10. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e dois valores inteiros, **K1** e **K2**,
 - devolva o número de elementos da lista com valores do campo **notaFinal** entre **K1** e **K2**.
11. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **num**,
 - remova o elemento da lista com valor no campo **numAluno** igual a **num**.
12. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **K**,
 - remova todos os elementos da lista cujo valor no campo **notaFinal** seja **menor** do que **K**.
13. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **nota**,
 - remova todos os elementos da lista com valor no campo **notaFinal** igual a **nota**.

- 14.** Implementar uma função que,
- receba uma lista com elementos do tipo **INFO** e um número inteiro positivo **N**,
 - remova da lista os seus **N** primeiros elementos (caso **N** seja maior do que o tamanho da lista, todos os seus elementos devem ser removidos).
- 15.** Implementar um função que
- receba uma lista com elementos do tipo **INFO**, e
 - ordene-a por ordem crescente do campo **notaFinal**.
- 16.** Implementar uma função que,
- receba uma lista com elementos do tipo **INFO** ordenados crescentemente pelo campo **notaFinal** e um inteiro **num**,
 - remova todos os elementos da lista com valor no campo **notaFinal** menor ou igual a **num**.
- 17.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - devolva um elemento com o segundo maior valor no campo **notaFinal** (percorrer a lista apenas uma vez).
- 18.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - remova o primeiro elemento e o último elemento da lista.
- 19.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - remova todos os elementos da lista com exceção do primeiro elemento e do último elemento.
- 20.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - coloque o último elemento na segunda posição da lista, mantendo a ordem dos restantes.
- 21.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - inverta a lista (o último elemento passa a primeiro, o penúltimo a segundo, ...).
- 22.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO** e um número inteiro **N**,
 - remova o **N-ésimo** elemento da lista.
- 23.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO** e um número inteiro **num**, e
 - construa duas novas listas, em que uma delas seja formada por elementos da lista cujo valor no campo **numAluno** é menor que **num** e a outra seja formada pelos restantes elementos da lista.

B. Considere a seguinte definição do tipo de dados **INFO**:

```
typedef struct{
    int numCC;           // número de cartão de cidadão (chave): valores entre 100000 e 200000
    int dataNasc[3];     // data de nascimento (dia/mes/ano), com ano entre 1930 e 2010
    float altura;        // em metros (ex: 1.86): valores entre 1.20 e 2.20
    int genero;          // 1 = feminino e 0 = masculino
} INFO;
```

Resolva os exercícios propostos utilizando as operações contidas nas bibliotecas referidas e acrescentado as novas operações à biblioteca "OperacoesExB.h".

1. Implementar uma função que,
 - receba um valor inteiro positivo **N**
 - crie e devolva uma lista com **N** elementos do tipo **INFO** gerados aleatoriamente (usar o gerador de números da biblioteca **Aleatorio**).
2. Elaborar um programa que realize as seguintes ações (pela ordem indicada):
 - defina o tamanho de uma lista (TAM), com um valor gerado aleatoriamente entre 0 e 15
 - use a função anterior para criar uma lista com TAM elementos do tipo **INFO**
 - mostre a lista criada, em que os elementos são apresentados **(a)** da cabeça/início para a cauda/fim e **(b)** da cauda/fim para a cabeça/início.

As funções que forem implementadas a partir daqui, devem ser testadas no programa (acrescentar).

3. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um número real **alt** (entre 1.0 e 2.2)
 - devolva a quantidade de elementos da lista, cujo valor do campo **altura** é maior que **alt**
4. Implementar uma função que
 - receba uma lista de elementos do tipo **INFO** e um número inteiro **A** (entre 1920 e 2025)
 - devolva a quantidade de elementos da lista nascidos no ano **A**.
5. Implementar uma função que,
 - receba uma lista com elementos do tipo **INFO** e um inteiro positivo **N** (entre 1 e 10),
 - remova da lista os seus **N** primeiros elementos; se **N** é maior que o tamanho da lista, então todos os seus elementos devem ser removidos, ficando a lista vazia.
6. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO**,
 - divida a lista em 2 listas do mesmo tipo, uma com todos os elementos do género feminino (**genero = 1**) e a outra com todos os elementos do género masculino (**genero = 2**).

Listas ligadas com ligações duplas

C. Considere as seguintes definições dos tipos de dados **INFO**, **NodoL** e **PNodeL** (apontamentos das aulas teóricas):

```
typedef struct {
    int numAluno;           // número de aluno (chave): { 70000, ..., 75000 }
    float notasMTP[2];      // notas dos mini-testes práticos: [0.0, 2.0]
    float notasTE[2];       // notas dos testes escritos: [0.0, 8.0]
    int notaFinal;          // nota final (valor arredondado da soma das 4 notas): { 0, ..., 20 }
} INFO;

struct NodoL {
    INFO Elemento;
    struct NodoL *Prox;
    struct NodoL *Ant;
};

typedef struct NodoL *PNodeL;
```

Cada elemento (registo) do tipo de dados **INFO** corresponde aos dados sobre a avaliação de um aluno à disciplina de AED, durante a época de Aprendizagem. O campo **numAluno** é a chave (valor único para cada elemento do tipo **INFO**).

Descarregar as seguintes bibliotecas (**Folhas práticas ---> Bibliotecas e Exercícios das folhas práticas ---> Listas ligadas duplas**):

Aleatorio.h ---> operações para gerar números aleatoriamente

EADLista.h ---> operações básicas sobre a EAD **Lista**

OperacoesBasicasExC.h ---> operações básicas sobre o tipo de dados **INFO**

OperacoesExC.h ---> operações a implementar relativas aos exercícios do **grupo C**

MainExC.c ---> programa principal relativo aos exercícios do **grupo C**

Resolva os exercícios propostos utilizando as operações contidas nas bibliotecas referidas e acrescentado as novas operações à biblioteca "OperacoesExC.h".

1. Implementar uma função que,

- receba um valor inteiro positivo **N**
- crie e devolva uma lista com **N** elementos do tipo **INFO** gerados aleatoriamente (usar o gerador de números da biblioteca **Aleatorio**).

2. Elaborar um programa que realize as seguintes ações (pela ordem indicada):

- defina o tamanho de uma lista (**TAM**), com um valor gerado aleatoriamente entre 0 e 15
- use a função anterior para criar uma lista com **TAM** elementos do tipo **INFO**
- mostre a lista criada, em que os elementos são apresentados **(a)** da cabeça/início para a cauda/fim e **(b)** da cauda/fim para a cabeça/início.

As funções que forem implementadas a partir daqui, devem ser testadas no programa (acrescentar).

3. Implementar uma função iterativa que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o **tamanho** (número de elementos) da lista.
4. Implementar uma função recursiva que
 - receba uma lista com elementos do tipo **INFO** e um valor inteiro **N**,
 - devolva a quantidade de elementos da lista com valor no campo **notaFinal** igual a **N**.
5. Implementar uma função iterativa que
 - receba uma lista com elementos do tipo **INFO** e um valor inteiro **N**,
 - devolva o número de elementos da lista com valor no campo **notaFinal** maior ou igual a **N**.
6. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o **maior** valor no campo **notaFinal** da lista.
7. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o aluno com melhor nota nas frequências (o elemento da lista com o maior valor na soma dos valores dos campos **notasTE**).
8. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO**,
 - devolva o aluno com pior nota nos mini-testes práticos (o elemento da lista com o menor valor num dos campos **notasMTP**).
9. Implementar uma função (iterativa ou recursiva) que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **K**,
 - devolva o menor elemento da lista com valor no campo **notaFinal** maior ou igual a **K**.
10. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **num**,
 - remova o elemento da lista com valor no campo **numAluno** igual a **num**.
11. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **K**,
 - remova todos os elementos da lista cujo valor no campo **notaFinal** seja **menor** do que **K**.
12. Implementar uma função que
 - receba uma lista com elementos do tipo **INFO** e um inteiro **nota**,
 - remova todos os elementos da lista com valor no campo **notaFinal** igual a **nota**.
13. Implementar um função que
 - receba uma lista com elementos do tipo **INFO**, e
 - ordene-a por ordem crescente do campo **notaFinal**.

- 14.** Implementar uma função que,
- receba uma lista com elementos do tipo **INFO** e um número inteiro positivo **N**,
 - remova da lista os seus **N** primeiros elementos (caso **N** seja maior do que o tamanho da lista, todos os seus elementos devem ser removidos).
- 15.** Implementar uma função que,
- receba uma lista com elementos do tipo **INFO** ordenados crescentemente pelo campo **notaFinal** e um inteiro **num**,
 - remova todos os elementos da lista com valor no campo **notaFinal** menor ou igual a **num**.
- 16.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - devolva um elemento com o segundo maior valor no campo **notaFinal** (percorrer a lista apenas uma vez).
- 17.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - remova o primeiro elemento e o último elemento da lista.
- 18.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - remova todos os elementos da lista com exceção do primeiro e do último elementos.
- 19.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - coloque o último elemento na segunda posição da lista, mantendo a ordem dos restantes.
- 20.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO**,
 - inverta a lista (o último elemento passa a primeiro, o penúltimo a segundo, ...).
- 21.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO** e um número inteiro **N**,
 - remova o **N-ésimo** elemento da lista.
- 22.** Implementar uma função que
- receba uma lista com elementos do tipo **INFO** e um número inteiro **num**, e
 - construa duas novas listas, em que uma delas seja formada por elementos da lista cujo valor no campo **numAluno** é menor que **num** e a outra seja formada pelos restantes elementos da lista.