Exercícios Listas

Prof. Andrei Braga
Prof. Geomar Schreiner

Exercícios Aulas Passadas

1. Considerando as definições a seguir, faça o que é pedido nos itens abaixo:

```c
typedef struct {
  int dia;
  int mes;
  int ano;
} Data;
```

```c
struct funcionario{
  int id;
  char nome[41];
  double salario;
  Data nascimento;
  struct funcionario *proximo;
};
typedef struct funcionario Funcionario;
```

   a. Crie as estruturas indicadas, e crie o primeiro funcionário da lista;
   b. Adicione um segundo funcionário ao início da lista;
   c. Crie uma função capaz de imprimir todos os funcionários;

Exercícios Aulas Passadas

2. Considerando a estrutura proposta no exercício anterior, faça as seguintes adaptações em seu programa:
    a. O programa deve ler (do teclado) um inteiro N que representará o número de registros que o usuário irá inserir. Após a leitura seu programa deve ler os dados dos N registros e os inserir na lista encadeada.
    b. Imprima a lista para ver se todos os elementos estão presentes
    c. Faça uma função que deleta um funcionário. A função deve receber como parâmetro a lista, e o id do funcionário a ser deletado, e deve retornar o primeiro elemento da lista

Exercícios Aulas Passadas

3. Implemente uma função que receba um vetor de valores inteiros com N elementos e construa uma lista encadeada armazenando os elementos do vetor (elemento a elemento). Assim, se for recebido por parâmetro o vetor v[4] = {1,21,4,6} a função deve retornar uma lista encadeada onde o primeiro elemento é ‘1’, o segundo o ‘21’, o terceiro o ‘4’ e assim por diante. A função deve ter a seguinte assinatura: *ListaInt* *constroiLista (int n, int *v);

Exercícios Aulas Passadas

4. Transforme a estrutura da lista implementada nas questões 1 e 2 em uma lista duplamente encadeada. E implemente as seguintes funcionalidades:
    a. Imprimir a lista do primeiro para o último elemento, e depois do último para o primeiro.
    b. Crie uma função de busca que apresenta as informações de um funcionário. A busca deve ser feita utilizando o id.
    c. Atualize a função de delete.

Exercícios Novos

5. Considerando as estruturas criadas no primeiro exercício:
    a. Implemente uma função que recebe como parâmetro a cabeça da lista, e retorna quantos elementos a lista possui.
    b. Implemente uma função que encontre e retorne o menor salário entre os funcionários.
    c. Implemente uma função que retorne o funcionário mais velho (retorne o Funcionario *).
    d. Implemente uma função que recebe duas listas, e verifica se as listas são iguais (os elementos devem aparecer na mesma ordem). Caso as listas sejam iguais retorne 1, se forem diferentes retorne 0.

Exercícios Novos

6. Considerando a estrutura abaixo, que contém apenas um valor inteiro:

```
typedef struct val {
  int valor;
  struct val *proximo;
  struct val *anterior;
} ListaInt;
```

   a. Implemente uma função que imprima o maior e o menor valor contido na lista.
   b. Implemente uma função que retorne a média dos valores da lista.
   c. Implemente uma função que retorna uma cópia da lista passada como parâmetro. Isso é, a função copia as informações de uma lista para a outra refazendo o aponteiramento.

Exercícios Novos

7. Considerando a estrutura abaixo, que contém apenas um valor inteiro:

```
typedef struct val {
  int valor;
  struct val *proximo;
  struct val *anterior;
} ListaInt;
```

   a. Implemente uma função que busque o menor elemento e o troque de lugar com o primeiro elemento da lista.
   b. Implemente uma função que busque o maior elemento e o coloque no fim da lista.

Exercícios Novos

8. Criar um programa para manter uma lista (duplamente encadeada) de trabalhos para a faculdade. Cada trabalho deve ter as seguintes informações:

   nome - char[80]
   descricao - char[400]
   entrega - int (armazena 0 se Não foi entregue, e 1 se foi entregue}
   prazo - Data (estrutura que usamos anteriormente)

   a. Criar uma estrutura trabalho para representar os trabalhos com os campos acima.
   b. Criar um ponteiro da estrutura de trabalho para a cabeça da lista e outro para a cauda (primeiro e último).
   c. O programa deve exibir ao usuário um menu com as opções para:
      i. Incluir um Trabalho;
      ii. Remover um Trabalho;
      iii. Navegar pelos Trabalhos com opção para ver o próximo Trabalho ou o anterior;
      iv. Pesquisar um Trabalho pelo prazo;
      v. Listar os trabalhos não entregues;
      vi. Sair do Programa.