Lista de Exercícios

Exercícios adaptados dos livros de BAKES (2016) e GRONER (2019)

Exercícios de Lista

Para os exercícios de 1 até 5, inclua os arquivos **lista.h** e **lista.c** em seu projeto. Faça a entrada de dados de **N** valores da lista, utilizando a função **InserirNoFim**.

1) Considerando uma lista previamente preenchida, faça uma função para buscar o produto de menor valor. A função deve retornar se a operação foi possível ou não e, se encontrado, os valores retornam por referência. Faça a alteração na estrutura **Dados**, para armazenar as seguintes informações:

```
typedef struct {
   int codProd;        //código do produto
   char nomeProd[10]; //nome do produto
   float valor;        //valor do produto
   int qtdeEstoque;    //quantidade disponível em estoque
} Dados
```

Para os exercícios de 2 até 5, faça a alteração na estrutura **Dados**, para armazenar um número inteiro, bem como a sub-rotina **ExibirLista**.

2) Considerando uma lista previamente preenchida, faça uma função que retorne quantos números pares existem na lista.

3) Considerando uma lista previamente preenchida, faça uma função que retorne uma nova lista contendo apenas os números pares da lista.

4) Considerando uma lista previamente preenchida, faça uma função que retorne a média dos valores da lista.

5) Considerando uma lista previamente preenchida, faça uma função que retorne o número de elementos da lista que possuem um número primo armazenado.

Exercícios de Pilha

Para os exercícios de 6 e 7, inclua os arquivos **pilha.h** e **pilha.c** em seu projeto. Faça a alteração na estrutura **Dados**, para armazenar um número inteiro.

6) Elabore um programa que, dado um valor inteiro entre 0 e 255, apresente sua representação binária (com 8 algarismos) utilizando uma pilha. Seu programa deve percorrer os seguintes passos:

a) armazenar na pilha o resto das divisões sucessivas enquanto que o numerador seja maior 1;
b) armazenar na pilha o último valor do numerador;
c) armazenar na pilha valores zero, enquanto a pilha tiver menos do que 8 itens;
d) retirar e exibir os valores da pilha, enquanto existirem.

7) Altere o programa anterior para realizar a conversão de um número inteiro positivo para qualquer base (até 36) que o usuário determinar. Para fazer a exibição, crie uma string com os caracteres que representam os algarismos, de 0 até 9 seguido de A até Z.

Exercícios de Fila

Para os exercícios de 8 e 9, inclua os arquivos **fila.h** e **fila.c** em seu projeto. Faça a alteração na estrutura **`Dados`**, para armazenar um número inteiro.

8) Faça a entrada de dados de duas fila, de tamanhos definidos pelo usuário. Para cada uma utilize a função **`InserirNaLista`** para armazenar os valores. Depois disso, faça uma função que recebe as duas filas e retorna uma fila com os elementos das duas intercalados conforme a ordem com que elas se dispõem na fila. Quando os valores da fila menor não existirem mais, incluir somente os valores da outra fila.
Por exemplo se a fila 1 tiver os valores 1, 2, 3, 4 e 5, enquanto que a fila 2 tiver os valores 9, 8 e 7, a fila 3 deverá apresentar os valores: 1, 9, 2, 8, 3, 7, 4 e 5.
Recomenda-se aumentar o tamanho máximo da fila no arquivo **fila.h**.

9) Para as partidas de videogame, o pessoal resolveu formar dois times, cada qual, armazenado em uma lista com o *nickname* dos jogadores. Após cadastrar todos os jogadores, o programa deve exibir quem são os primeiros de cada fila e iniciar um laço de repetição. Durante o laço, o programa pede para que seja digitado o número do time do jogador que venceu a partida (1 ou 2), tirando o próximo jogador da fila, exibindo o *nickname* e incluindo-o no final da própria fila. O programa é executado até que seja digitado o valor 0.

10) Jogos *online* são uma realidade para todas as plataformas atuais (console, mobile e computador). Hoje, um número cada vez maior de jogadores pode jogar de maneira simultânea em diversos jogos. Contudo, quando o jogador entra na plataforma e a quantidade de jogadores ainda não é o suficiente para nova partida ou se todas as salas disponíveis estão cheias, ele é colocado em uma fila de espera. Você ficou encarregado de escrever o código dessa fila de espera.
No início do seu programa, as salas são criadas com a capacidade máxima de usuários definidos previamente no código. Enquanto a sala possuir lugar, novos jogadores devem ser adicionados nela (os jogadores são identificados por seus IDs numéricos e seus *nicknames*). Quando a sala estiver cheia, eles devem ser adicionados em uma fila de espera. Quando a partida terminar, os jogadores devem ser movidos da fila de espera para a partida.
Construir um menu que possibilite ao usuário do seu sistema simular qual ação ele deseja testar (terminar uma partida, adicionar novo jogador, iniciar uma partida com os jogadores atuais ou mostrar *status* do sistema).

Referências

BACKES, A. R. **Estrutura de dados descomplicada**: em linguagem C. Rio de Janeiro: Elsevier. 1 ed. 2016.

GRONER, L. **Estruturas de dados e algoritmos com JavaScript**: Escreva um código JavaScript complexo e eficaz usando a mais recente ECMAScript. São Paulo: Novatec. 2 ed. 2019. 408 p.