Projeto de
SI200 Algoritmos e Programação de Computadores II

Prof. Luis A. A. Meira

4 de outubro de 2019

Suponha que você está fazendo um programa para um restaurante.

1 Restaurante.h

Primeiramente crie um arquivo *Restaurante.h*. Neste arquivo deve haver haver uma estrutura *Pedido* e uma estrutura *Pedidos*. A estrutura *Pedido* deve conter um ID incremental, que inicia em 1 e vai aumentando. Não pode haver dois pedidos com mesmo ID. O ID deve aumentar sempre de uma unidade. Além do ID a estrutura deve conter um texto livre onde será anotado o pedido. Também deve possuir um int mesa, que representa a mesa onde foi feito o pedido e um marcador de tempo (timestamp). Você pode usar o $time(NULL)$ para marcar o tempo de criação do pedido.

No arquivo Restaurante.h ficam as assinaturas das funções.

Você deve criar uma função $Pedido * newPedido(char * texto, int\ mesa)$. O ID será preenchido automaticamente com o menor valor disponível.

Você deve criar uma função $Pedido * lePedido()$. Nesta função, os dados do pedido serão lidos do teclado. Esta função deve, obrigatoriamente chamar *newPedido*.

Você deve criar uma função $addPedido(Pedido * p)$. Esta função adiciona um novo pedido. Note que não há uma variável do tipo *Pedidos*. Isso porque o novo pedido não ficará em memória e sim em arquivo.

Você deve criar uma função $Pedidos * listaPedidos()$ que devolve todos os pedidos que ainda não foram processados.

Você deve criar uma função $Pedido * removeMaisAntigo()$. Esta função remove da lista de pedidos o pedido com menor ID. O pedido deve ser impresso na tela. Deve ser impresso o tempo transcorrido entre a criação e a remoção do pedido. O pedido removido é devolvido por meio de um *return*.

2 Restaurante.c

Nesta função você deve implementar as funções. ATENÇÃO, nenhuma informação poderá ser salva em memória. Toda informação deve ser salva em arquivos.

Imagine que existem 5 garçons trabalhando. Se os dados ficarem na memória, os garçon podem criar pedidos com IDs repetidos. Por esta razão, toda informação deve estar salva em arquivos. Os arquivos são compartilhados por todos os garçons, bem como pela cozinha.

Você deve criar uma função $newPedidos()$ que cria um conjunto de pedidos vazio.

Você deve implementar todas as funções do arquivo Restaurante.h

Por exemplo, você pode criar um arquivo $MAX\_ID.txt$. Toda vez que um novo ID for criado, o sistema pega o $MAX\_ID.txt$ e soma 1.

Já os pedidos podem ser salvos em um arquivo texto, um abaixo do outro. Lembrando que existem operações de inserir pedido e de remover o pedido mais antigo.

3 Main.c

No *main.c* você deve criar um menu com opções inserir pedido, listar pedidos, remover pedido mais antigo. Ao remover o pedido mais antigo, você deve imprimir o tempo gasto entre a abertura e a remoção do pedido, bem como as informações do pedido removido.

O main não deve guardar nenhuma informação. Toda informação criada deve ser passada para as funções e depois devem ser descartadas. O único local onde ficam as informações são os arquivos.

Isso significa que, se fechar o main e abrir novamente, o programa deve continuar funcionando normalmente. Isso também significa que dois programas podem executar ao mesmo tempo.

4 Conflitos

Suponha que existem dois ou mais programas executando ao mesmo tempo. Abrindo pedidos por exemplo. Isso pode gerar um conflito entre eles. Por exemplo, o primeiro programa lê $MAX\_ID = 30$ e cria um pedido com $ID = 31$. Ao mesmo tempo, outro programa lê $MAX\_ID = 30$ e cria um pedido com $ID = 31$. Você tem alguma sugestão de como tratar este problema?