Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação

Avaliação Final – Fila de Prioridade
Disciplina Programação (CK0226) – Semestre 2020.2
ATIVIDADE INDIVIDUAL

Prof. Miguel Franklin

Desenvolver um projeto em linguagem C representando uma estrutura de dados de fila de prioridade para uma aplicação de atendimento de clientes de um banco.

A estrutura deverá ser composta de três filas distintas (Premium, Ouro e Comum) e de um escalonador. As filas serão identificadas por inteiros 0, 1 e 2, respectivamente.

Cada uma das filas terá disciplina de atendimento FIFO (*First In First Out*), isto é, o primeiro elemento a entrar na fila será o primeiro elemento a sair. As filas deverão armazenar um número inteiro (int), que representará o número da conta corrente do cliente do banco.

O escalonador deverá gerenciar as três filas, organizando-as em uma fila única. Assim, o escalonador deverá puxar elementos (clientes) das filas, seguindo a ordem de prioridade.

As funções a serem implementadas para a estrutura de dados são as seguintes:

- `void enfileirar (int num_fila, int numero_conta);`

  Esta função deverá inserir na fila `num_fila` o cliente de número de conta `numero_conta`. Não é necessário checar se o número de conta é único. Considere que o `num_fila` está sempre no intervalo [0; 2].

- `int desenfileirar();`

  Esta função deverá escolher a próxima fila a ser atendida (de acordo com a DISCIPLINA DE ATENDIMENTO, a ser detalhada abaixo) e retirar o próximo número de conta a ser atendido, retornando o número da conta.

A aplicação (função main) deverá ler um arquivo texto de configuração cujo nome será passado como parâmetro na chamada da aplicação. Neste arquivo, a primeira linha conterá apenas um valor inteiro *N*, que representará quantos clientes deverão ser enfileirados. As *N* linhas seguintes terão dois valores, separados por barra '/' (sem espaços): o número da fila do cliente e o número da conta corrente do cliente.

Um exemplo de arquivo de configuração é mostrado abaixo:

```
10
1/50
0/562
2/59
0/433
1/684
0/543
2/444
0/546
1/756
2/313
```

Após ler o arquivo de configuração e colocar todos os clientes nas respectivas filas, o programa deverá efetuar o desenfileiramento de todos os clientes das filas, na ordem definida pela DISCIPLINA DE ATENDIMENTO e, dentro de cada fila, na ordem FIFO. Para cada cliente desenfileirado, o programa deve mostrar na tela, um por linha, o cliente que está sendo atendido. Para o exemplo mostrado acima, a saída do programa deverá ser:

```
562
433
543
50
684
59
546
756
444
313
```

A implementação da aplicação (função main) deverá, então, fazer chamadas de função à implementação do escalonador unicamente (definidas na Parte II deste enunciado). A implementação do escalonador, por sua vez, é quem poderá fazer chamadas as funções da fila FIFO (definidas na Parte I deste enunciado).

DISCIPLINA DE ATENDIMENTO

O escalonador deve seguir o seguinte procedimento para cada rodada de atendimento:

Passo 1. Atender 3 (três) clientes Premium, se houver;
Passo 2. Em seguida, atender 2 (dois) clientes Ouro, se houver;
Passo 3. Em seguida, atender 1 (um) cliente Comum, se houver, e ir para o primeiro passo da próxima rodada;

Parte I: Estrutura de Dados Fila FIFO

Deverá ser desenvolvida uma estrutura de dados utilizando ponteiros e alocação dinâmica de memória (nos mesmos moldes das estruturas de dados desenvolvidas em laboratório) para uma fila com disciplina FIFO (*First In, First Out*), onde um elemento que chega primeiro na fila é também o primeiro elemento a ser retirado. Esta estrutura de dados deve ser desenvolvida nos arquivos fila_fifo.c e fila_fifo.h. Esta fila terá como informação a ser armazenada apenas um valor inteiro, que será a chave de indexação. As funções dessa API não devem realizar nenhuma entrada e saída de informação na tela (scanf ou printf). As seguintes funções devem ser implementadas:

```
void f_inicializar (Fila_FIFO **f);
```
Inicializa a fila.

```
int f_inserir (Fila_FIFO **f, int chave);
```
Insere um determinado valor inteiro indexado por um valor de chave na fila. Retorna 1 se a inserção for bem sucedida e 0 se houver algum problema (duplicação de chave ou falta de memória).

```
int f_obter_proxima_chave (Fila_FIFO **f);
```
Retorna o número de chave do próximo elemento da fila, retirando-o da fila. Retorna -1 se a fila estiver vazia.

Parte II: Escalonador

Desenvolver um escalonador de filas que trata de forma diferenciada 3 filas, com as características de disciplina citadas no enunciado no trabalho. Os arquivos no projeto para o escalonador deverão ser: “escalonador.c” e “escalonador.h”.

O escalonador deve ter as seguintes funções:

```
void e_inicializar (Escalonador *e);
```
Inicializa o escalonador, alocando e inicializando as 3 filas de clientes.

```
int e_inserir_por_fila (Escalonador *e, int classe, int num_conta);
```
Insere na fila “classe” o cliente de número “num_conta.

```
int e_obter_prox_num_conta(Escalonador *e);
```
Retorna o número da conta do próximo cliente a ser atendido de acordo com a DISCIPLINA DE ATENDIMENTO, retirando-o da sua respectiva fila.

Critérios de Avaliação

A avaliação será realizada em duas fases:

1. Análise do código-fonte;
2. Análise do atendimento dos requisitos do enunciado;
3. Análise funcional automatizada da execução do programa (teste).

O código-fonte será avaliado de acordo com os seguintes critérios qualitativos:

i. Eficácia do programa em suprir todos os requisitos;
ii. Eficiência do programa (otimização);
iii. Organização do código (uso racional de subprogramas, estruturas, etc.);
iv. Legibilidade do código (uso de endentação e semântica dos identificadores de variáveis);
v. Documentação (comentários dentro do código fonte).

Os trabalhos serão corrigidos no **Linux**. Portanto, certifique-se que o trabalho feito no Windows também compila e roda no Linux.