ACH2023 – turmas 04 e 94 – EP2 2º Semestre de 2017

Fila e atendimento preferencial – versão 2

Todos estamos habituados a enfrentarmos filas (estruturas nas quais aqueles que chegam vão para o final da fila e a pessoa a ser atendida é aquela que está no início da fila). Adicionalmente, também encontramos "filas preferenciais" em diversos locais.

Neste EP você deverá gerenciar uma estrutura de fila que tenta atender aos dois propósitos de uma vez: ter o comportamento de uma fila tradicional (na qual o primeiro a ser atendido é aquele que está no início da fila e quem entra na fila deverá entrar no final dela), mas também possuir uma segunda forma de atendimento: atender primeiro a pessoa mais velha da fila.

Dentre as operações previstas para esta fila estão:

- inserção de um elemento;
- exclusão do elemento mais velho da fila;
- exclusão do primeiro elemento da fila;
- consulta à idade de um elemento da fila;
- **abandono da fila (um elemento qualquer resolve sair da fila);**
- consulta à quantidade de elementos (tamanho) da fila.

Para este EP, você deverá implementar um conjunto de funções de gerenciamento de filas utilizando principalmente dois conceitos: filas e listas duplamente ligadas ordenadas **(circulares e com nó cabeça)**.

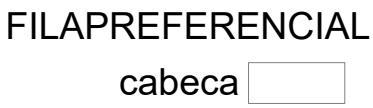
A seguir são apresentadas as estruturas de dados envolvidas nesta implementação e como elas serão gerenciadas.

A estrutura básica será o *REGISTRO*, que contém seis campos: *id* (identificador inteiro do elemento), *idade* (número inteiro com a idade do elemento), *antFila* (ponteiro para o elemento anterior, isto é, aquele que chegou antes na fila), *proxFila* (ponteiro para o elemento posterior, isto é, aquele que chegou depois na fila), *antIdade* (ponteiro para o elemento mais prioritário, isto é, o que possui idade imediatamente maior do que a do elemento atual), e *proxIdade* (ponteiro para o elemento menos prioritário, isto é, o que possui idade imediatamente menor do que a do elemento atual).

```
typedef struct aux {
  int id;
  int idade;
  struct aux* antFila;
  struct aux* proxFila;
  struct aux* antIdade;
  struct aux* proxIdade;
} REGISTRO, * PONT;
```

A estrutura *FILAPREFERENCIAL* possui um campo: *cabeca* que é um ponteiro para o nó cabeça que é um elemento do tipo *REGISTRO* e será criado pela função criarFila e nunca deverá ser excluído. Não é considerado um elemento válido (do ponto de vista do usuário) e sempre estará no início da estrutura. Tanto a fila propriamente dita (de elementos inseridos pela ordem de chegada) quanto a lista ligada ordenada de forma decrescente pela idade serão estruturas duplamente ligadas e circulares e este nó cabeça será o nó cabeça de ambas. O ponteiro *antFila* do nó cabeça apontará para o último elemento da fila; o ponteiro *proxFila* do nó cabeça apontará para o primeiro elemento da fila; o ponteiro *antIdade* do nó cabeça apontará para o último elemento da lista ordenada por

idade (o elemento com menor idade); e o ponteiro *proxIdade* apontará para o primeiro elemento da lista ordenada por idade (o elemento mais velho).

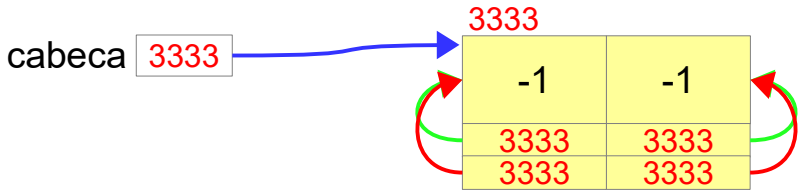



```
typedef struct {
                PONT cabeca;
              } FILAPREFERENCIAL, * PFILA;
```

A função criarFila é responsável por criar uma nova fila, já criando o nó cabeça e acertando seus ponteiros.

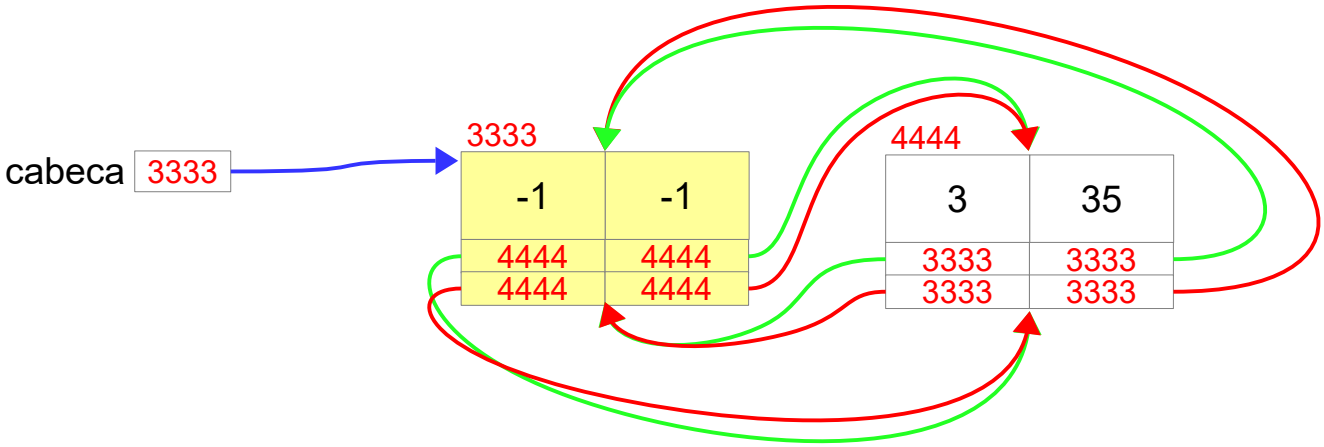
```
PFILA criarFila(){
    PFILA res = (PFILA) malloc(sizeof(FILAPREFERENCIAL));
    PONT cab =  (PONT) malloc(sizeof(REGISTRO));
    cab->id = -1;
    cab->idade = -1;
    cab->antFila = cab;
    cab->proxFila = cab;
    cab->antIdade = cab;
    cab->proxIdade = cab;
    res->cabeca = cab;
    return res;
}
```

Exemplo de fila preferencial recém criada:

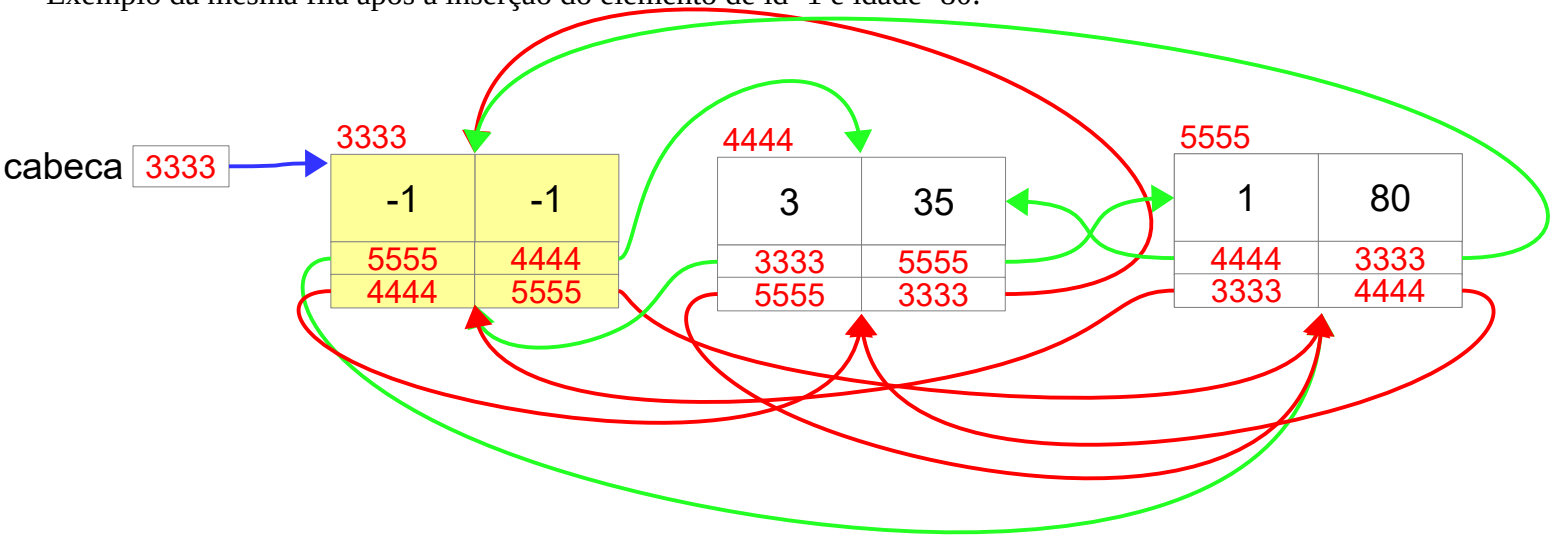


Ao se inserir um novo elemento na estrutura, este deverá ser incluindo nas duas listas duplamente ligadas, circulares e com nó cabeça, lembrando-se que uma se comporta como uma fila (inserção no final) e outra como uma lista duplamente ligada ordenada de forma decrescente de acordo com a idade.

Exemplo de fila após a inserção do elemento de id=3 e idade=35



Exemplo da mesma fila após a inserção do elemento de id=1 e idade=80:



Funções que deverão ser implementadas no EP

int tamanho(PFILA f): função que recebe o endereço de uma fila preferencial e retorna o número de elementos na fila (ou mais precisamente, da lista duplamente ligada, o nó cabeça não deverá ser contado).

bool inserirElemento(PFILA f, int id, int idade): função que recebe o endereço de uma fila preferencial, o identificador do novo elemento e o valor de sua idade.
Esta função deverá retornar *false* caso já exista um registro com esse identificador na fila.
Caso contrário, a função deverá alocar memória para esse novo registro e inseri-lo nas posições corretas: da fila (no final da fila) e de acordo com sua idade (na lista preferencial), acertando todos os ponteiros necessários e retornar *true*. Se já houver outros elementos na lista ordenada pela idade com a mesma idade do elemento que será inserido, então este deverá ser inserido após todos aqueles que possuem sua idade.

PONT removerElementoFila(PFILA f): esta função recebe como parâmetro o endereço de uma fila preferencial e deverá retornar NULL caso a fila esteja vazia (ou melhor, caso só possua o nó cabeça). Caso contrário, deverá retirar o primeiro elemento da fila (elemento apontado por *f->cabeca->proxFila*), acertar todos os ponteiros necessários e retornar o endereço do respectivo registro. A memória desse registro não deverá ser apagada, pois o usuário pode querer usar esse registro para alguma coisa.

PONT removerElementoIdade(PFILA f): esta função recebe como parâmetro o endereço de uma fila preferencial e deverá retornar NULL caso a fila esteja vazia. Caso contrário, deverá retirar o elemento com maior idade (elemento apontado por *f->cabeca->proxIdade*), acertar todos os ponteiros necessários e retornar o endereço do respectivo registro. A memória desse registro não deverá ser apagada, pois o usuário pode querer usar esse registro para alguma coisa.

bool abandonarFila(PFILA f, int id): esta função recebe como parâmetro o endereço de uma fila preferencial e o identificador de um elemento deverá retornar *false* caso não exista ninguém na fila com o respectivo identificador. Caso contrário, deverá retirar o elemento da fila que possui esse identificador, acertar todos os ponteiros necessários, liberar a memória desse elemento e retornar *true*.

bool consultarIdade(PFILA f, int id, int resposta)*: função que recebe o endereço de uma fila preferencial, o identificador do elemento e um endereço para uma memória do tipo *int*.
Esta função deverá retornar *false* caso não haja um registro com esse identificador na fila.
Caso contrário, a função deverá colocar na memória apontada pela variável *resposta* o valor da idade do respectivo elemento e retornar *true*.

Informações gerais:

Os EPs desta disciplina são trabalhos individuais que devem ser submetidos pelos alunos via sistema TIDIA até às 23:55h (com margem de tolerância de 60 minutos).

Você receberá três arquivos para este EP:

- filaPreferencial.h que contém a definição das estruturas, os *includes* necessários e o cabeçalho/assinatura das funções. Vocês não deverão alterar esse arquivo.
- filaPreferencial.c que conterá a implementação das funções solicitadas (e funções adicionais, caso julguem necessário). Este arquivo já contém o esqueleto geral das funções e alguns códigos implementados.
- usaFilaPreferencial.c que contém alguns testes executados sobre as funções implementadas.

Você deverá submeter **apenas** o arquivo filaPreferencial.c, porém renomeie este arquivo para seu_número_USP.c (por exemplo, 3140792.c) antes de submeter.

Não altere a assinatura de nenhuma das funções e não altere as funções originalmente implementadas (*exibirLog* e *criarFila*).

Nenhuma das funções que você implementar deverá imprimir algo. Para *debugar* o programa você pode imprimir coisas, porém, na versão a ser entregue ao professor, suas funções não deverão imprimir nada (exceto pela função *exibirLog* que já imprime algumas informações).

Você poderá criar novas funções (auxiliares), mas não deve alterar o arquivo filaPreferencial.h e seu código será testado com uma versão diferente do arquivo usaFilaPreferencial.c. Suas funções serão testadas individualmente e em grupo.

Todos os trabalhos passarão por um processo de verificação de plágios. Em caso de plágio, todos os alunos envolvidos receberão nota zero.