

Fila Preferencial – versão 2

O objetivo deste EP é gerenciar uma fila de uma loja que atende aos clientes seguindo uma regra simples: primeiro atende a todos os clientes preferenciais e em seguida atende aos demais. Dentro de cada categoria (preferenciais ou não) cada cliente é atendido de acordo com sua ordem de chegada na fila. Para este EP, apenas terão direito ao atendimento preferencial as pessoas cujas idades forem maiores ou iguais a um valor representado em uma constante no código.

Você será responsável por implementar a solução computacional para gerenciar a entrada dos clientes na fila e a ordem de atendimento. Para isto, você deverá gerenciar uma estrutura que contém uma lista ligada de pessoas (**esta estrutura será duplamente ligada, circular e possuirá nó cabeça**). Essa lista conterá informações de todas as pessoas que entrarem na fila (independentemente de terem ou não direito ao atendimento preferencial). Na representação em memória, cada elemento da lista possuirá dados de uma pessoa (identificador e idade), um ponteiro para o elemento anterior da fila e um ponteiro para o próximo elemento da fila

Dentre as operações previstas para esta estrutura estão (em negrito estão destacadas as funções que deverão ser implementadas por você neste EP):

- criação de uma fila preferencial;
- busca por uma pessoa;
- consulta à quantidade de pessoas na fila (tamanho);
- **inserção/entrada de uma pessoa na fila;**
- **atendimento de uma pessoa (da primeira pessoa da fila);**
- **desistência/saída de uma pessoa da fila;**

Para este EP você deverá implementar um conjunto de funções de gerenciamento da “Fila Preferencial” utilizando principalmente os conceitos de **filas e listas duplamente ligadas, circulares e com nó cabeça**.

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

Os elementos básicos dentro de uma fila serão representados pela estrutura *ELEMENTO*, que contém quatro campos: *id* (identificador inteiro da pessoa), *idade* (número inteiro com a idade em anos da pessoa), *ant* (ponteiro para o endereço do elemento anterior da fila) e *prox* (ponteiro para o endereço do próximo elemento da fila).

```
typedef struct aux {  
    int id;  
    int idade;  
    struct aux* ant;  
    struct aux* prox;  
} ELEMENTO, * PONT;
```

ELEMENTO

id	idade
ant	prox

A estrutura *FILAPREFERENCIAL* possui dois campos do tipo ponteiro para *ELEMENTO* (endereço de memória): *cabeca* e *inicioNaoPref*. A variável *cabeca* deve apontar para o nó cabeça (o nó cabeça é criado na inicialização e nunca deverá ser apagado); a variável *inicioNaoPref* deve apontar para a primeira pessoa da fila que não tem direito ao atendimento preferencial (caso não haja ninguém na fila nessa condição, esta variável deve apontar para o nó cabeça). Esta fila de elementos será duplamente ligada, circular e possuirá um nó cabeça. A representação da estrutura pode ser vista a seguir.

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

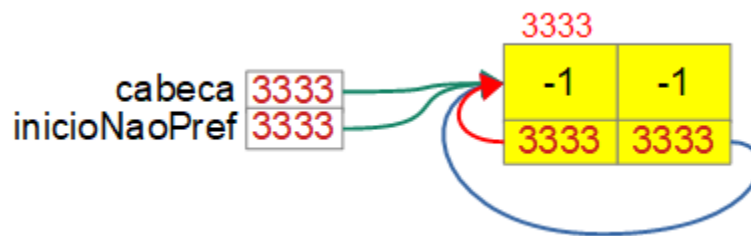
FILA PREFERENCIA

<i>cabeca</i>	
<i>inicioNaoPref</i>	

A função `criarFila` é responsável por criar uma nova fila, preencher os valores iniciais dos campos da estrutura `FILAPREFERENCIAL` (incluindo a alocação da memória do nó `cabeca`) e retornar o endereço da estrutura criada:

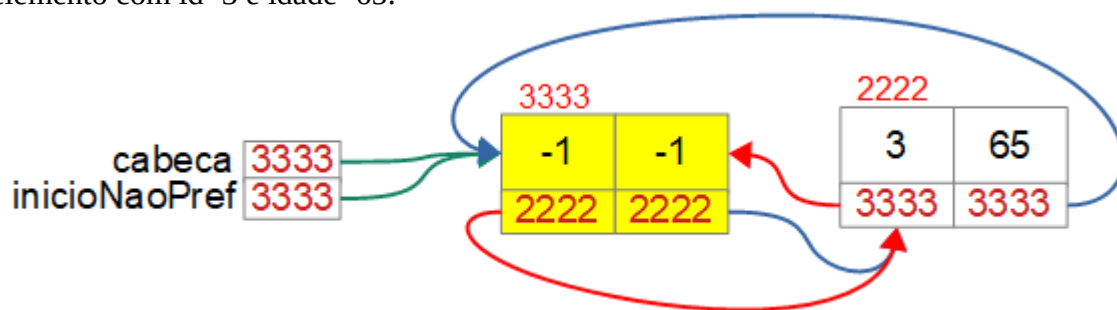
```
PFILA criarFila(){
    PFILA res = (PFILA) malloc(sizeof(FILAPREFERENCIAL));
    res->cabeca = (PONT) malloc(sizeof(ELEMENTO));
    res->inicioNaoPref = res->cabeca;
    res->cabeca->id = -1;
    res->cabeca->idade = -1;
    res->cabeca->ant = res->cabeca;
    res->cabeca->prox = res->cabeca;
    return res;
}
```

Exemplo de fila recém-criada:



Ao se inserir uma primeira pessoa na estrutura, esta deverá ser inserida após o nó `cabeca`. Lembre-se, a estrutura é circular. Adicionalmente, há dois casos diferentes:

1º) Se a pessoa tiver direito ao atendimento preferencial (isto é, idade maior ou igual à constante `IDADEPREFERENCIAL` [vamos assumir neste enunciado que esta constante vale 60]). Exemplo de fila após a inserção do elemento com `id=3` e `idade=65`:



2º) Se a pessoa não tiver direito ao atendimento preferencial (isto é, idade inferior à constante `IDADEPREFERENCIAL`). Exemplo de fila após a inserção do elemento com `id=2` e `idade=25`:

Caso contrário, a função deverá alocar memória para o novo elemento, preencher os campos *id* e *idade* com os valores passados como parâmetro e inserir esse elemento na fila, conforme a regra de atendimento preferencial. Isto é: se a idade da pessoa for maior ou igual à constante *IDADEPREFERENCIAL* (presente no arquivo *filapreferencial.h*), então esta pessoa deverá ser inserida após a última pessoa com direito ao atendimento preferencial e antes da primeira pessoa na fila que não tenha esse direito. Caso contrário, essa pessoa deverá ser inserida no fim da fila (após a última pessoa da fila). Após a inserção deste novo elemento na posição correta da fila, a função deverá retornar *true*. Note que há diversas situações específicas que devem ser tratadas: antes da inserção a fila pode estar sem pessoas (apenas com o nó cabeça), pode ter apenas pessoas sem direito ao atendimento preferencial, pode ter pessoas apenas com direito ao atendimento preferencial, ou pode ter pessoas nas duas categorias.

bool atenderPrimeiraDaFila(PFILA f, int id):* função que recebe o endereço de uma fila preferencial e o endereço de uma variável do tipo inteiro e retorna um valor booleano.

Esta função deverá retornar *false* se a fila estiver sem nenhuma pessoa (isto é, apenas com o nó cabeça).

Caso contrário, a função deverá colocar o identificador da primeira pessoa da fila (isto é, primeiro elemento após o nó cabeça) na variável apontada pelo endereço armazenado em *id*. Deverá também remover esta pessoa da fila, acertar os ponteiros necessários para a fila não ficar inconsistente, liberar a memória do elemento correspondente à pessoa que foi excluída/atendida e retornar *true*. Note que há diversas situações específicas que devem ser tratadas: antes do atendimento a fila pode ter, além do nó cabeça, uma única pessoa (com ou sem direito ao atendimento preferencial [ficando apenas com o nó cabeça após o atendimento]), pode ter apenas pessoas sem direito ao atendimento preferencial, pode ter pessoas apenas com direito ao atendimento preferencial, ou pode ter pessoas nas duas categorias.

bool desistirDaFila(PFILA f, int id): função que recebe o endereço de uma fila preferencial e o identificador de uma pessoa e retorna um valor booleano.

Esta função deverá retornar *false* se a pessoa com identificador igual a *id* não estiver na fila.

Caso contrário, a função deverá excluir da fila a pessoa cujo identificador possua valor igual à *id*, acertar os ponteiros necessários para a fila não ficar inconsistente, liberar a memória do elemento correspondente à pessoa que foi excluída (que abandonou a fila) e retornar *true*. Note que o nó cabeça não corresponde a uma pessoa e por isso nunca deverá ser excluído. Há diversas situações específicas que devem ser tratadas: a pessoa que abandonará a fila pode ser a única da fila (além do nó cabeça), pode ser a primeira com direito ao atendimento preferencial, a primeira sem direito ao atendimento preferencial, pode ser a última com direito ao atendimento preferencial, pode ser a última pessoa da fila, etc.

Informações gerais:

Os EPs desta disciplina são trabalhos individuais que devem ser submetidos pelos alunos via sistema TIDIA (ae4.tidia-ae.usp.br/) até as 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. Você não deverá alterar esse arquivo.
- `filapreferencial.c` que conterá a implementação das funções solicitadas (e funções adicionais, caso julgue 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, `12345678.c`) antes de submeter.

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

Nenhuma das funções que você implementará deverá imprimir algo. Para *debuggar* 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`. Adicionalmente, saiba que seu código será testado com uma versão diferente do arquivo `usaFilaPreferencial.c`. Suas funções serão testadas individualmente e em conjunto.

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