Estruturas

Fila dinâmica

Entenda o mecanismo de implementação de filas com alocação dinâmica de memória

---

Definição da fila dinâmica

A definição de uma fila dinâmica é bastante similar à definição de uma pilha dinâmica. A diferença é que, agora, temos dois ponteiros em `FILA`, um para o `inicio` e outro para o `fim` da fila. Você verá que todo o processo acaba sendo mais simples do que fila estática.

```
typedef struct{
    int chave;
} REGISTRO;

typedef struct item{
    REGISTRO r;
    struct item* prox;
} ITEM;

typedef struct{
    ITEM *inicio;
    ITEM *fim;
} FILA;
```

Inicializando

A inicialização da `FILA` envolve basicamente definir os valores dos ponteiros de `inicio` e `fim` como `NULL`:

```
void inicializar(FILA *f){
    f->inicio = NULL;
    f->fim = NULL;
}
```

Inserindo elementos

A inserção de elementos também depende da função `criarItem`. Ela, no entanto, é idêntica à usada para a criação de itens da pilha.

```
ITEM* criarItem(REGISTRO r){
    ITEM* novo = (ITEM*) calloc(1,sizeof(ITEM));
    if(novo){
        novo->r = r;
        novo->prox = NULL;
    }
    return novo;
}
```

A diferença está na lógica da função inserir. Vejamos:

```
void inserir(FILA *f, REGISTRO r){
    ITEM *novo = criarItem(r);
    if(!novo) return;

    if(!f->fim){
        f->inicio = novo;
        f->fim = novo;
    }
    else{
        f->fim->prox = novo;
        f->fim = novo;
    }
}
```

Como numa fila a inserção acontece sempre no fim, primeiro verificamos se a fila possui um fim atualmente. São duas possibilidades:

- Se `f->fim == NULL` (não tem fim), o item a ser inserido será o primeiro. Logo, ele é tanto o início quanto o fim da fila;

- Se a fila já tem fim, precisamos informar ao antigo último item que agora tem mais alguém na fila. Depois, apontamos o fim da fila para o novo elemento.

Simples, não?

Remoção do elemento

Ao removermos um elemento, não apenas o excluímos da fila, mas o retornamos - para que ele possa ser utilizado por alguma outra função. Pelo menos essa é o entendimento que estamos usando nesta implementação.

Por isso, neste caso, a função `remover` retorna um `ITEM*` . A lógica dela é: se a fila tiver início, ou seja, não está vazia, faça o seguinte:

- Crie um ponteiro auxiliar que guarde o atual início da fila;
- Faça o início da fila apontar para o próximo item;
- Se a fila passar a não ter mais início, ela também não tem mais fim;
- Retorne o elemento que era o início da fila;

```
ITEM* remover(FILA *f){
    if(f->inicio){
        ITEM *aux = f->inicio;
        f->inicio = f->inicio->prox;
        if(!f->inicio) f->fim = NULL;
        return aux;
    }
    else return NULL;
}
```

Imprimindo a fila

O processo de imprimir talvez seja o mais simples. Começamos pelo início da fila. Se ele não for nulo, imprimimos o seu valor e, depois, seguimos para o próximo elemento.

```
void imprimir(FILA *f){
    ITEM *aux = f->inicio;
    while(aux){
```

```
        printf("%d\n", aux->r.chave);
        aux = aux->prox;
    }
}
```

Limpando a fila

Ao terminarmos de usar uma fila dinâmica, precisamos fazer a liberação do espaço ocupado por ela. No nosso caso, a necessidade envolvem o espaço ocupado pelos itens, que foram criados dinamicamente pela função `criarItem`. A função de liberar memória é basicamente uma adaptação da impressão:

```
void liberar(FILA *f){
    ITEM *aux = f->inicio;
    while(aux){
        ITEM *exc = aux;
        aux = aux->prox;
        free(exc);
    }
    inicializar(f);
}
```

Exemplo completo

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

typedef struct{
    int chave;
} REGISTRO;

typedef struct item{
    REGISTRO r;
    struct item* prox;
} ITEM;

typedef struct{
    ITEM *inicio;
    ITEM *fim;
```

```
} FILA;

void inicializar(FILA *f){
    f->inicio = NULL;
    f->fim = NULL;
}

ITEM* criarItem(REGISTRO r){
    ITEM* novo = (ITEM*) calloc(1,sizeof(ITEM));
    if(novo){
        novo->r = r;
        novo->prox = NULL;
    }
    return novo;
```



```
void inserir(FILA *f, REGISTRO r){
    ITEM *novo = criarItem(r);
    if(!novo) return;

    if(!f->fim){
        f->inicio = novo;
        f->fim = novo;
    }
    else{
        f->fim->prox = novo;
        f->fim = novo;
    }
}

ITEM* remover(FILA *f){
    if(f->inicio){
        ITEM *aux = f->inicio;
        f->inicio = f->inicio->prox;
        if(!f->inicio) f->fim = NULL;
        return aux;
    }
    else return NULL;
}

void imprimir(FILA *f){
    ITEM *aux = f->inicio;
    while(aux){
        printf("%d\n", aux->r.chave);
        aux = aux->prox;
    }
}
```

```c
void liberar(FILA *f){
    ITEM *aux = f->inicio;
    while(aux){
        ITEM *exc = aux;
        aux = aux->prox;
        free(exc);
    }
    inicializar(f);
}

int main(void){
    FILA f;
    inicializar(&f);
    inserir(&f, (REGISTRO){10});
    inserir(&f, (REGISTRO){20});
    inserir(&f, (REGISTRO){30});
    inserir(&f, (REGISTRO){40});

    /* Checar a remoção */
    ITEM *rem = remover(&f);
    if(rem){
        printf("Removido: %d\n", rem->r.chave);
        free(rem);
    }

    imprimir(&f);
    liberar(&f);

    return 0;
}
```


Estruturas
← Fila estática

Estruturas
Lista sequencial →