Estrutura de Dados
Estrutura de dados Linear: Fila

Alessandro Ferreira Leite

09/04/2012

Introdução

- Assim como a estrutura de dados Pilha, Fila é outra estrutura de dados bastante utilizada em computação.
- Um exemplo é a implementação de uma fila de impressão.
- Se uma impressora é compartilhada por várias máquinas, normalmente adota-se uma estratégia para determinar a ordem de impressão dos documentos.
- A maneira mais simples é tratar todas as requisições com a mesma prioridade e imprimir os documentos na ordem em que foram submetidos – o primeiro submetido é o primeiro a ser impresso.

Fila

Definição

Um conjunto ordenado de itens a partir do qual podem-se eliminar itens numa extremidade (chamada de início da fila) e no qual podem-se inserir itens na outra extremidade (chamada final da fila).

Representação

- Os nós de uma fila são armazenados em endereços contínuos.
- A Figura 1 temos a representação de uma fila com três elementos.

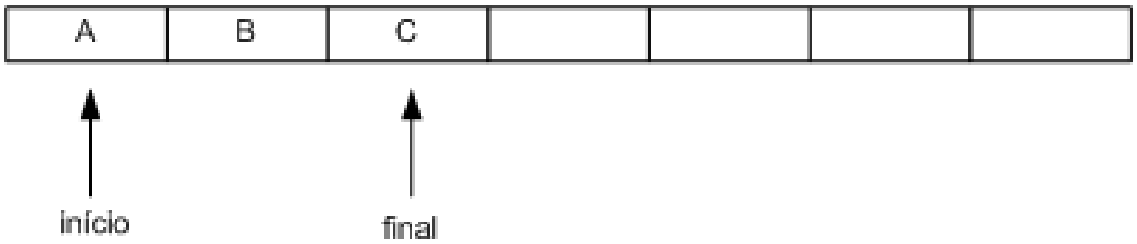

Figura: Exemplo de representação de fila

- Após a retirada de um elemento (*primeiro*) temos:

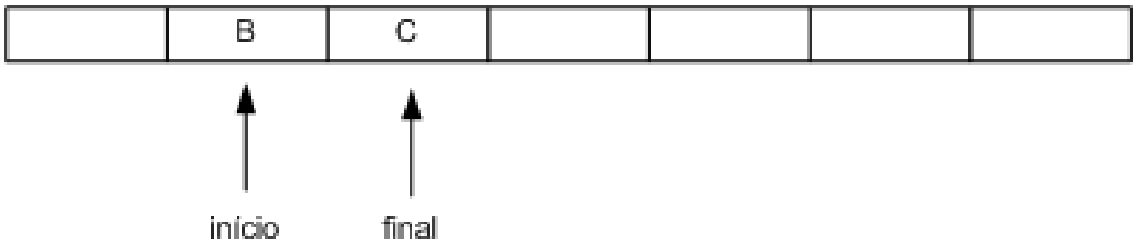

Figura: Representação de uma fila após a remoção do elemento "A"

Representação

- Após a inclusão de dois elementos temos:

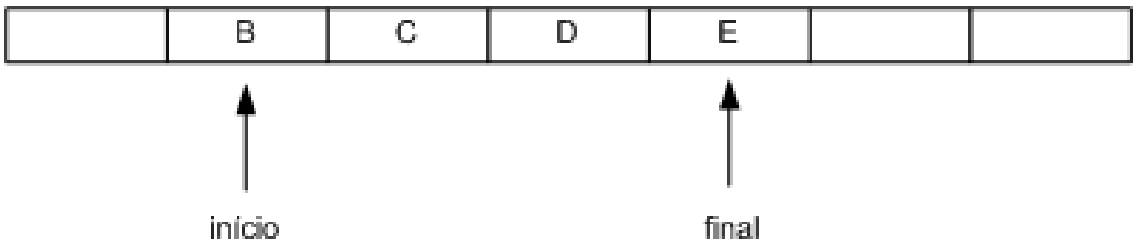

Figura: Representação de uma fila após a inclusão de dois elementos “D” e “E”

- Como podemos observar, a operação de inclusão e retirada de um item da fila incorre na mudança do endereço do ponteiro que informa onde é o início e o término da fila.

Representação

- Em uma fila, o primeiro elemento inserido é o primeiro a ser removido.
- Por essa razão, uma fila é chamada **fifo(*first-in first-out*)** - primeiro que entra é o primeiro a sair - ao contrário de uma pilha que é lifo (*last-in, first-out*)
- Para exemplificar a implementação em C, vamos considerar que o conteúdo armazenado na fila é do tipo inteiro.
- A estrutura de fila possui a seguinte representação:

```
struct fila{
   int elemento[N];
   int ini, n;
}
typedef struct fila Fila;
```

- Trata-se de uma estrutura heterogênea constituída de membros distintos entre si. Os membros são as variáveis *ini* e *fim*, que serve para armazenar respectivamente, o início e o fim da fila e o vetor *elemento* de inteiros que armazena os itens da fila.

Operações Primitivas

- As operações básicas que devem ser implementadas em uma estrutura do tipo Fila são:

Operação	**Descrição**
criar()	aloca dinamicamente a estrutura da fila.
insere(f,e)	adiciona um novo elemento *(e)*, no final da fila *f*.
retira(f)	remove o elemento do início da fila *f*.

Tabela: Operações básicas da estrutura de dados fila

Operações auxiliares

- Além das operações básicas, temos as operações "auxiliares". São elas:

Operação	**Descrição**
vazia(f)	informa se a fila está ou não vazia
libera(f)	destrói a estrutura, e assim libera toda a memória alocada

Tabela: Operações auxiliares da estrutura de dados fila

Interface do Tipo Fila

```c
typedef struct fila Fila;
/* Aloca dinamicamente a estrutura Fila, inicializando seus
 * campos e retorna seu ponteiro. A fila depois de criada
 * estará vazia.*/
Fila* criar(void);

/* Insere o elemento e no final da fila f, desde que,
 * a fila não esteja cheia.*/
void insere(Fila* f, int e);

/* Retira o elemento do início da fila, e fornece o
 * valor do elemento retirado como retorno, desde que a fila
 * não esteja vazia*/
int retira(Fila* f);

/*Verifica se a fila f está vazia*/
int vazia(Fila* f);

/*Libera a memória alocada pela fila f*/
void libera(Fila* f);
```

Implementação de Fila com Vetor

- Assim como nos casos da pilha e lista, a implementação de fila será feita usando um vetor para armazenar os elementos.
- Isso implica, que devemos fixar o número máximo de elementos na fila.
- O processo de inserção e remoção em extremidades opostas fará a fila "andar" no vetor.
- Por exemplo, se inserirmos os elementos 8, 7, 4, 3 e depois retiramos dois elementos, a fila não estará mais nas posições iniciais do vetor.

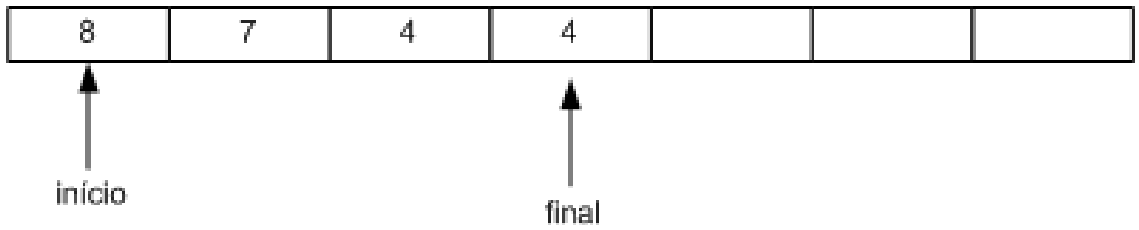

Figura: Fila após inserção de quatro elementos

Implementação de Fila com Vetor

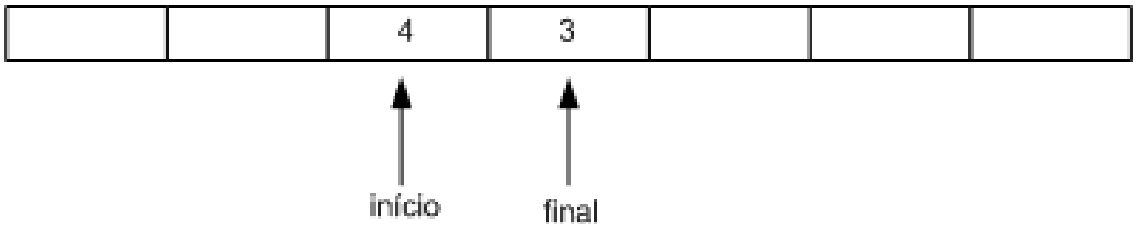

Figura: Fila após retirar dois elementos

- Com essa estratégia, é fácil observar que, em um dado instante, a parte ocupada pelo vetor pode chegar a última posição.
- Uma solução seria ao remover um elemento da fila, deslocar a fila inteira no sentido do início do vetor.
- Entretanto, essa método é bastante ineficiente, pois cada retirada implica em deslocar cada elemento restante da fila. Se uma fila tiver 500 ou 1000 elementos, evidentemente esse seria um preço muito alto a pagar.

Implementação de Fila com Vetor

- Para reaproveitar as primeira posições do vetor sem implementar uma "re-arrumação" dos elementos, podemos incrementar as posições do vetor de forma "circular".
- Para essa implementação, os índices do vetor são incrementados de maneira que seus valores progridam "circularmente".
- Dessa forma, se temos 100 posições no vetor, os índices assumem os seguintes valores:

$$0, 1, 2, 3, \cdots, 98, 99, 0, 1, 2, 3, \cdots, 98, 99, \cdots$$

Função de Criação

- A função que cria uma fila, deve criar e retornar o ponteiro de uma fila vazia;
- A função deve informar onde é o início da fila, ou seja, fazer $f->ini = 0$, como podemos ver no código abaixo.
- A complexidade de tempo para criar a fila é constante, ou seja, $O(1)$.

```
/* Aloca dinamicamente a estrutura Fila, inicializando seus
 * campos e retorna seu ponteiro. A fila depois de criada
 * estará vazia.
 */
Fila* criar(void)
{
    Fila* f = malloc(sizeof(Fila));
    f->n = 0;
    f->ini = 0;
    return f;
}
```

Função de Inserção

- Para inserir um elemento na fila, usamos a próxima posição livre do vetor, indicada por n.
- Devemos assegurar que há espaço para inserção do novo elemento no vetor, haja vista se tratar de um vetor com capacidade limitada.
- A complexidade de tempo para inserir um elemento na fila é constante, ou seja, $O(1)$.

```
/* Insere o elemento e no final da fila f.*/
void insere(Fila* f, int e)
{
   int fim;
   if (f->n == N){
      printf("Fila cheia!\n");
   }else{
      fim = (f->ini + f->n) % N;
      f->elementos[fim] = e;
      f->n++;
   }
}
```

Função de Remoção

- A função para retirar o elemento do início da fila fornece o valor do elemento retirado como retorno.
- Para remover um elemento, devemos verificar se a fila está ou não vazia.
- A complexidade de tempo para remover um elemento da fila é constante, ou seja, $O(1)$.

```
int retira(Fila* f)
{
  int e;
  if (vazia(f))
    printf("Fila vazia!\n");
  else{
    e = f->elementos[f->ini];
    f->ini = (f->ini + 1) % N;
    f->n--;
  }
  return e;
}
```

Exemplo de Uso da Fila

```
#define N 10
#include <stdio.h>
#include "fila.h"

int main(void)
{
    Fila* f = criar();

    int i;
    for (i = 0; i < N; i++)
      insere(f,i * 2);

    printf("\nElementos removidos: ");

    for (i = 0; i < N/2; i++)
     printf("%d ", retira(f));

    system("pause");
}
```

Referências

1. Tenenbaum, A. M., Langsam, Y., and Augestein, M. J. (1995). Estruturas de Dados Usando C. MAKRON Books, pp. 207-250.
2. Wirth, N. (1989). Algoritmos e Estrutura de dados. LTC, pp. 151-165.