Estruturas de Dados I

Estruturas de Dados - Filas

Prof. Dr. Lidio Mauro Lima de Campos

limadecampos@gmail.com

Universidade Federal do Pará – UFPA
ICEN
PPGCC

Agenda

- Filas
  - Introdução
  - Operações Básicas
  - Filas Implementação por Vetor e Listas.

Funções de Acesso tipo Fila

- **typedef struct fila Fila;**

/*Função que cria uma Fila.*/

- **Fila* fila_cria(void);**

/*Testa se uma Fila é vazia.*/

- **int fila_vazia(Fila *f);**

/*Função que adiciona um elemento em uma Fila.*/

- **void fila_insere(Fila *f, int info);**

/*Função que remove um elemento de uma Fila.*/

- **int fila_remove(Fila *f);**

/*Função que imprime os elementos de uma Fila.*/

- **void fila_imprime(Fila *f);**

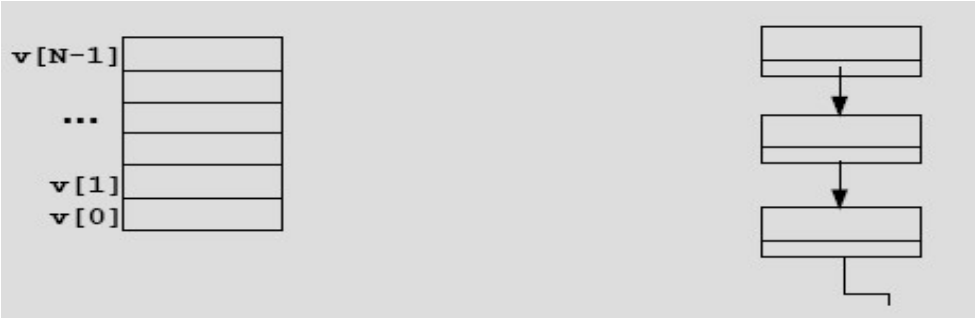
/*Libera o espaço alocado para uma Fila.*/

- **void fila_libera(Fila *f);**

Implementação de Filas

- Podemos implementar a interface fila de duas maneiras:

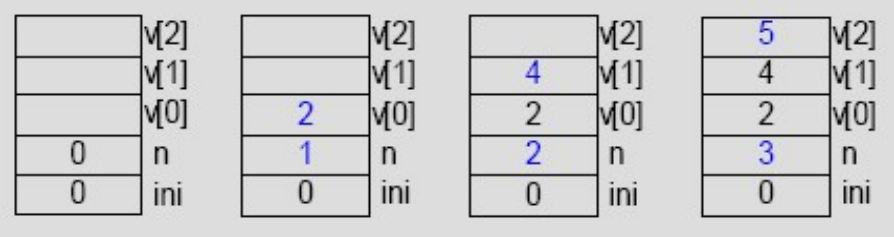
Vetor, quando o número máximo de elementos é Conhecido.	Lista, quando não se sabe o número máximo de elementos.



Implementação de Fila com Vetor

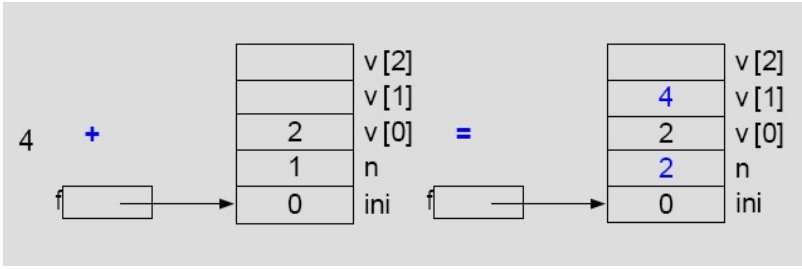
- A estrutura que representa a o tipo fila deve ser composto pelo vetor e pelo número de elementos armazenados.

#define N 3 **typedef struct fila** **{** **int n;** **int ini;** **int v[N];** **}Fila;**	**Fila f;** **f.n=0,** **f.ini=0;** **f.v[0] = 2; f.n++;** **f.v[1] = 4; f.n++;** **f.v[2] = 5; f.n++;**



Implementação de Fila com Vetor – Função Insere

```
voe fila_insere(Fila *f, int info)
{
  int fim;
  if(f->n==N)
 {
  printf("Capacidade da Fila Estorou!!!\n");
  exit(1);
 }
 fim=(f->ini+f->n)%N;
 f->v[fim]= info;
 f->n++;
}
```



Primeira Inserção

fim=(0+0)%100=0

f->v[0]=2; f->n=1;

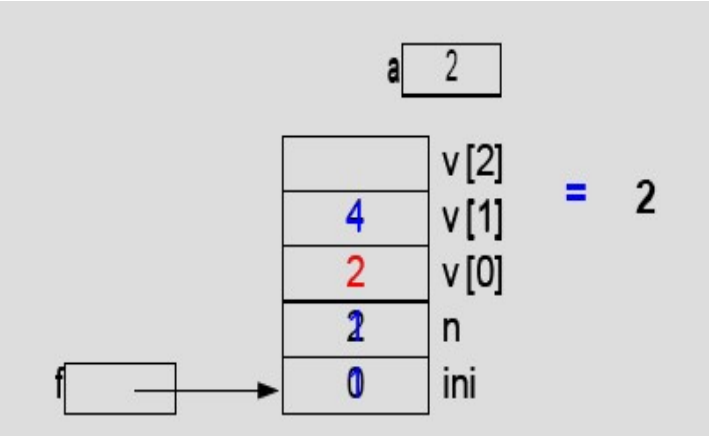
Segunda Inserção

fim=(0+1)%100=1 **//Resto da Divisão por 100**

f->v[1]=4; f->n=2;

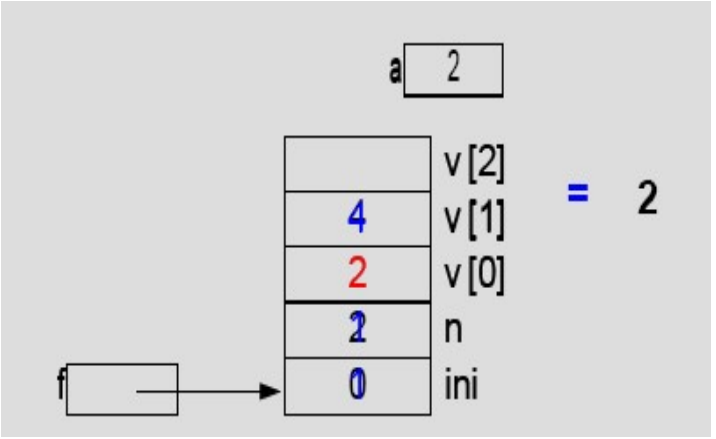
Implementação de Fila com Vetor – Função Fila Vazia e Remove

```
/*Testa se uma fila é vazia.*/
int fila_vazia(Fila *f)
{ return f->n==0; }
```



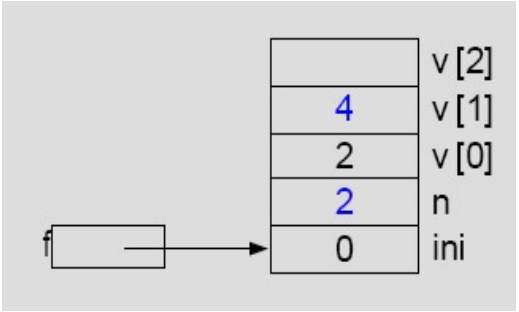
Implementação de Fila com Vetor – Função Fila Vazia e Remove

```
/*Função que remove um elemento de uma fila.*/
int fila_remove(Fila *f)
{ int a;
  if(fila_vazia(f))
  { printf("Fila Vazia!!!\n");
   exit(1);
  }
  a = f->v[f->ini];
  f->ini = (f->ini+1)%N;
  f->n--;
  return a;
}//a=f->v[f->ini]; a=f->v[0]=2;
//f->ini = (f->ini+1)%N;
//f->ini =(0+1) %3;
// f->ini =1;
//f->n=1
```



Implementação de Fila com Vetor – Função Fila Imprime e Libera

```
voe fila_imprime(Fila *f)
{ int i; int k;
  for(i = 0; i<f->n;i++)
 {
   k = (f->ini+i) % N;
   printf("%d\n",f->v[k]);
 }
}
//k = (f->ini+i)%N;
//k =(0+0) %3;
//k= 0;
voe fila_libera(Fila *f)
{
  free(f);
}
```



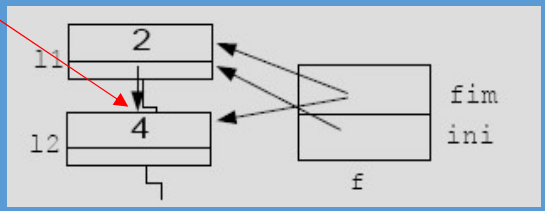
2
4

Implementação de Fila com Lista Encadeada

- A estrutura que representa o tipo fila é composta pelo início e pelo fim de uma lista.

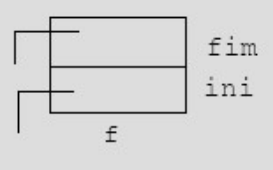
```
typedef struct lista Lista;
typedef struct fila Fila;

struct lista
{
    int info;
    Lista *prox;
  };

struct fila
{
 Lista *ini;
 Lista *fim;
};
```

```
Fila f;
Lista l1;
l1.info=2;
l1.prox=NULL;
f.ini=l1; f.fim=l1;
Lista l2;
l1.prox=l2;
l2.info=4;
l2.prox=NULL;
f.fim.prox=l2; //encadeamento l1 e l2
f.fim=l2;
```



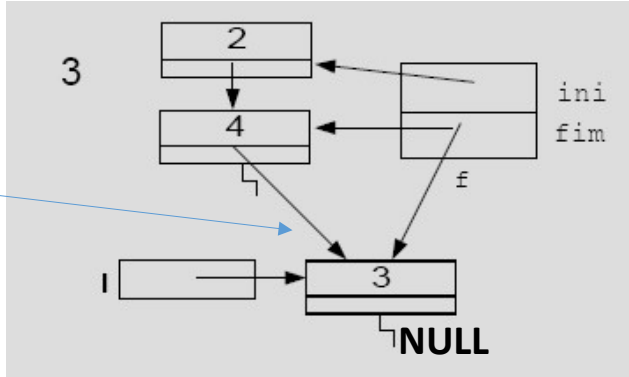
Implementação de Fila com Lista Função Cria

```
Fila* fila_cria(void)
{
 Fila *f = (Fila*)malloc(sizeof(Fila));
 if(f==NULL)
{
  printf("Memoria insuficiente!!!\n");
  exit(1);
}
f->ini = NULL;
f->fim = NULL;
return f;
}
```



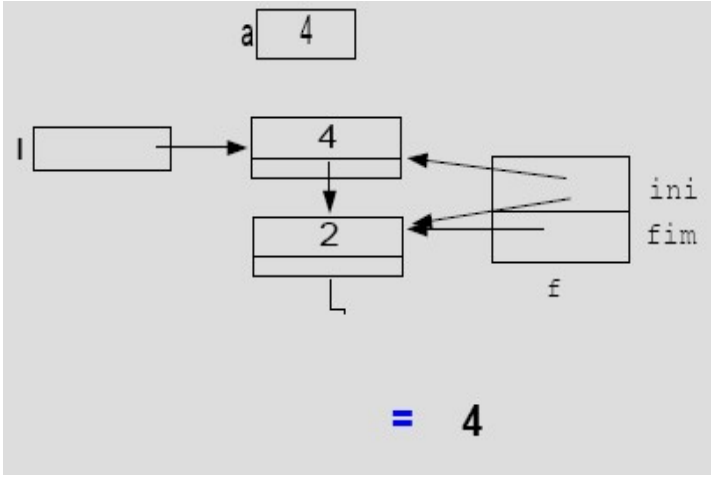
Implementação de Fila com Lista Função Insere

```
void fila_insere(Fila *f, int info)
{
  Lista *l = (Lista*)malloc(sizeof(Lista));
  if(l==NULL)
  {
   printf("Memoria insuficiente!!!\n");
   exit(1);
  }
  l->info = info;
  l->prox = NULL;
  if(!fila_vazia(f)) //fila não esta vazia
     f->fim->prox = l;// encadeamento do novo nó
   else //fila esta vazia
     f->ini = l;
   f ->fim = l; //f->fim aponta novo nó inserido
}
```



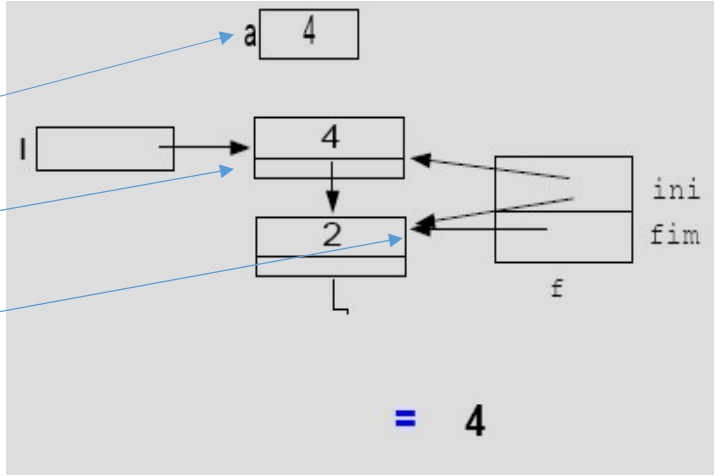
Implementação de Fila com Lista Função Vazia e Remove

```
/* fila_vazia
int fila_vazia(Fila *f)
{
  return f->ini==NULL;
}
```



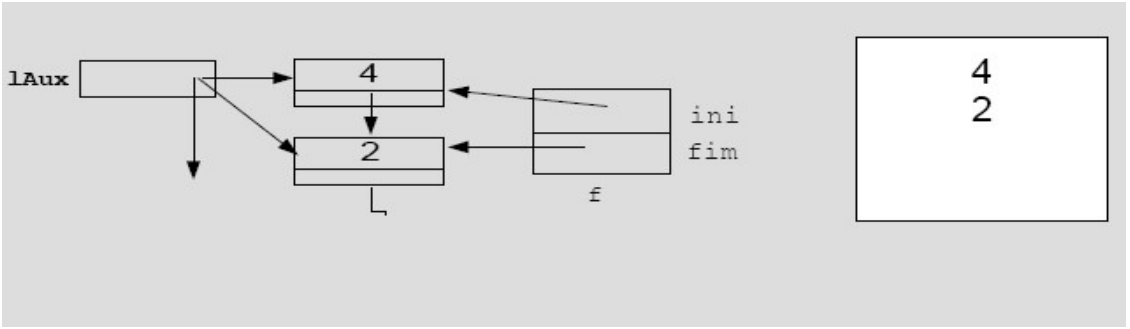
Implementação de Fila com Lista Função Vazia e Remove

```
/* fila_remove
int fila_remove(Fila *f)
{ Lista *l; int a;
  if(fila_vazia(f))
  { printf("Fila Vazia!!!\n");
    exit(1);
  }
  a = f->ini->info;
  l = f->ini;
  f->ini = f->ini->prox;
  free(l);
  if(fila_vazia(f)) //Verifica se a Fila ficou vazia
    f->fim = NULL;
  return a;
}
```



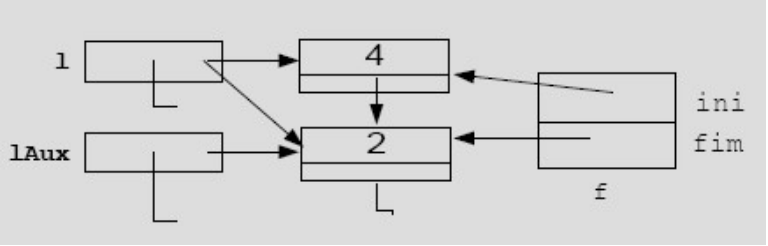
Implementação de Fila com Lista Função Imprime

```
/*Função que imprime os elementos de uma fila.*/
void fila_imprime(Fila *f)
{
  Lista *lAuz = f->ini;
  while(lAux!=NULL)
 {
    printf("%f\n",lAux->info);
    lAux = lAux->prox;
 }
}
```



Implementação de Fila com Lista Função Libera

```
/*Libera o espaço alocado para uma fila.*/
void fila_libera(Fila *f)
{
  Lista* l = f->ini;
  Lista* lAux;
  while(l!=NULL)
 {
   lAux = l->prox;
   free(l);
   l = lAux;
 }
free(f);
}
```



Exercícios

Questão #1

- Considere a estrutura de dados fila, do tipo FIFO. Entidades são inseridas nessa estrutura com a operação push( ) e removidas com a operação pop( ). A opção a seguir que mostra o conteúdo ordenado da fila após a seqüência de operações

push(8), push(7), push(5), push(2), pop( ), push(8), push(7),pop( ), push(5), push(2), pop( ), pop( ) é:

(A) 8578
(B) 8758
(C) 8752
(D) 2875
(E) 2758

8,7,5,2
7,5,2
7,5,2,8
7,5,2,8,7
5,2,8,7
5,2,8,7,5,2
8,7,5,2

Exercícios

Questão #2

- Uma das estruturas de dados utilizadas na programação de computadores funciona conforme o princípio conhecido como FIFO – "First In First Out" e uma como LIFO – "Last In First Out". Essas estruturas são denominadas, respectivamente:

(A) Lista Circular e Árvore

(B) Árvore e Lista Linear

(C) Pilha e Lista Circular

(D) Lista Linear e Fila

(E) Fila e Pilha

Exercícios

Questão #3

- A representação a seguir refere-se a um conjunto de elementos armazenados em um array. A remoção de um elemento desse conjunto segue a regra "o primeiro elemento que entra é o primeiro elemento que sai (FIFO)".

A representação acima refere-se a uma:

(A) pilha;

(B) fila;

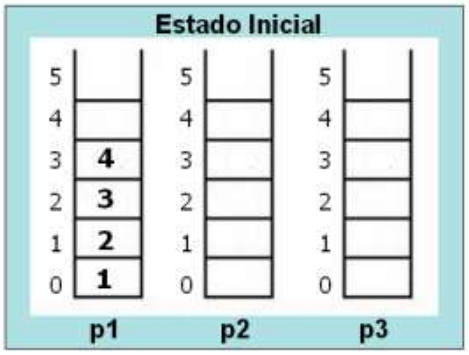
(C) lista encadeada;

(D) árvore;

(E) lista binária;

Exercícios

- Questão 4) Dado o estado inicial das pilhas p1, p2 e p3 na figura abaixo, mostre (desenhe as pilhas) o estado final dessas mesmas pilhas após as operações descritas no código abaixo. Considere que p1, p2 e p3 sejam instâncias da classe Stack (pilha com alocação sequencial) Caso não seja possível realizar alguma operação, escreva que não foi possível e ignore-a:




```
int temp = p1.pop();
p2.push(temp);
p3.push(p1.pop());
p2.push(p1.pop());
temp = p1.pop();
p3.push(temp);
p1.push(p2.pop());
p3.push(p2.pop());
p3.push(p1.pop());
```

Exercícios

- Questão 5) Sobre o tema, Estrutura de Dados, analise as assertivas e assinale a alternativa correta.

- I. Pilhas - São estruturas de dados do tipo LIFO (last-in first-out), onde o último elemento a ser inserido, será o primeiro a ser retirado. Assim, uma pilha permite acesso a apenas um item de dados - o último inserido. Para processar o penúltimo item inserido, deve-se remover o último.
- II. FILAS - São estruturas de dados do tipo FIFO (first-in first-out), onde o primeiro elemento a ser inserido, será o primeiro a ser retirado, ou seja, adiciona-se itens no fim e remove-se do início.
- III. Lista linear é uma estrutura de dados na qual elementos de um mesmo tipo de dado estão organizados de maneira sequencial. Não necessariamente, estes elementos estão fisicamente em sequência, mas a ideia é que exista uma ordem lógica entre eles.
- IV. Árvore é uma estrutura de dados que herda as características das topologias em árvore. Conceitualmente diferente das listas encadeadas, em que os dados se encontram numa sequência, nas árvores os dados estão dispostos de forma hierárquica. Uma árvore é formada por um conjunto de elementos que armazenam informações chamados nodos. Toda a árvore possui o elemento chamado raiz, que possui ligações para outros elementos denominados ramos ou filhos. Estes ramos podem estar ligados a outros elementos que também podem possuir outros ramos. O elemento que não possui ramos é conhecido como nó folha, nó terminal ou nó externo.

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Algoritmos – Teoria e Prática, Tradução da Segunda Edição. Campus, 2016.
- Ziviani, N. Projeto de Algoritmos Com Implementações em Pascal e C, Pioneira Thomson Learning, 4ed. 2009.
- http://www2.hawaii.edu/~suthers/courses/ics311f20/Notes/Topic-07.html
- U. Manber. **Algorithms: A Creative Approach**. . Addison-Wesley. 1989.
- J. Kleinberg e E. Tardos. **Algorithm Design**. . Addison Wesley. 2005
- Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados, Editora Campus* (2004)