Estruturas de Dados

Fila *por Alocação Dinâmica de Memória*

PUC Minas, *campus* de Poços de Caldas

Curso de Bacharelado em Ciência da Computação

ED – Estruturas de Dados

Estrutura de Dados Fila

Definição

Estrutura de dados que respeita o seguinte critério de utilização:

todo novo elemento é inserido no final da fila; e

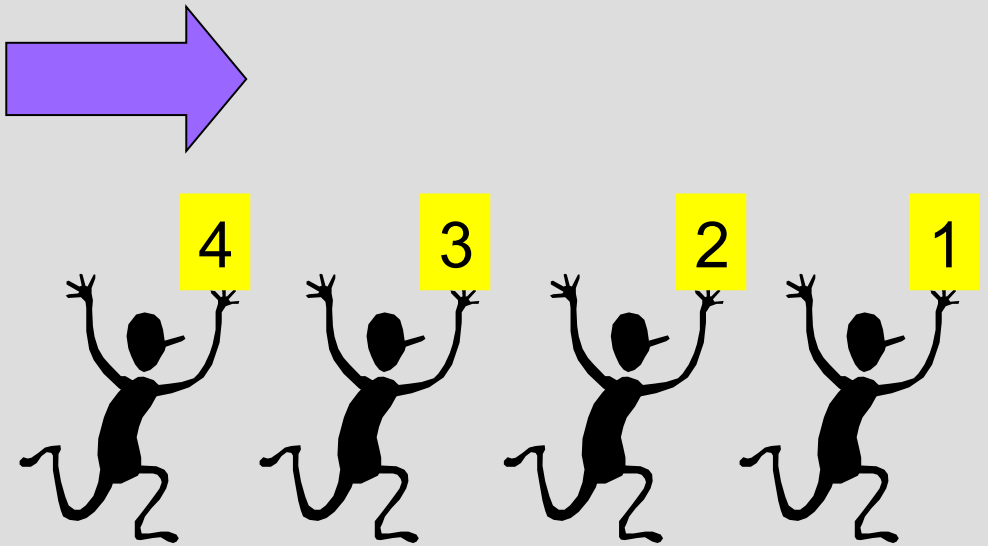
só pode ser consumido o primeiro elemento da fila.

FIFO

First In - First Out

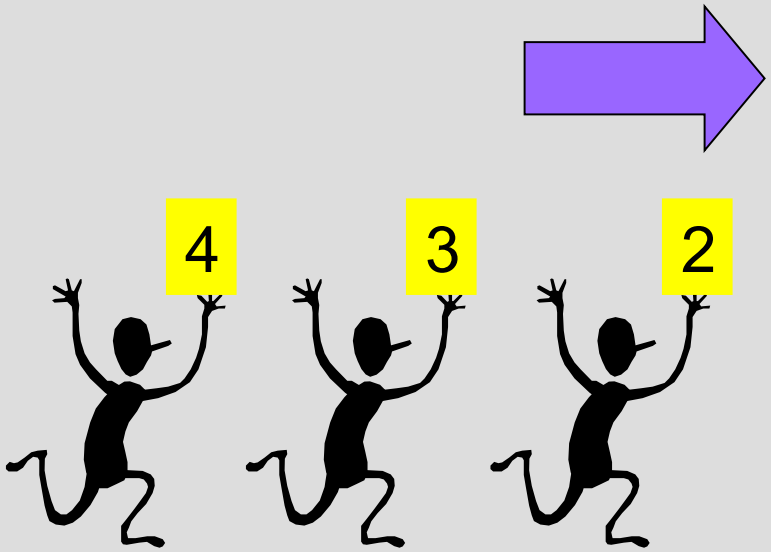
Estrutura de Dados Fila

Simulação



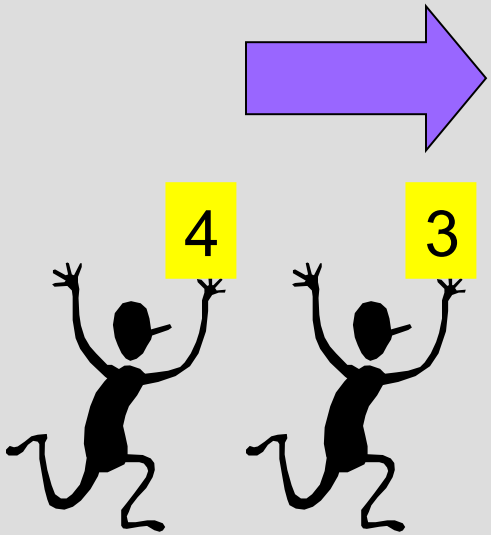
Estrutura de Dados Fila

Simulação



Estrutura de Dados Fila

Simulação



Estrutura de Dados Fila

Simulação

Estrutura de Dados Fila

Funções Essenciais para Manipulação de Filas

create(&f): Criar e Inicializar a Fila;

insert(&f,e): Inserir o elemento "e" no final da Fila

remove(&f,&e): Remover o primeiro elemento da Fila, retornando o resultado no parâmetro "e";

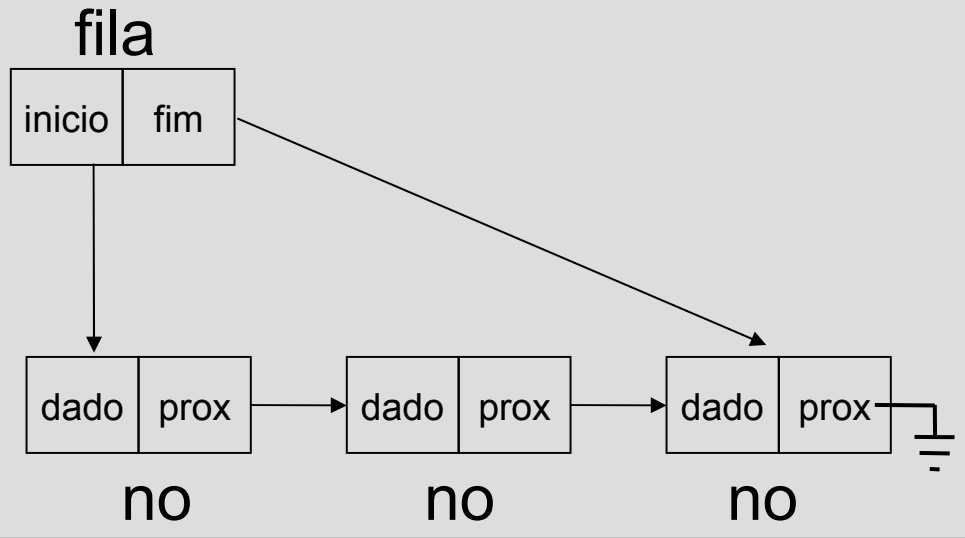
isEmpty(f): Verificar se a Fila está vazia;

Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
void create(fila *q);

int isEmpty(fila q);

int insert(fila *q, int d);

int removee(fila *q, int *d);
```

```
struct no
{
 int dado;
 struct no *prox;
};

typedef struct
{
 struct no *inicio;
 struct no *fim;
} fila;
```



Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
  fila f;
  int N;
  create(&f);
  insert(&f,12);
  insert(&f,320);
  insert(&f,413);
  printf("Ordem de chegada");

  if (!isEmpty(f))
  {
    removee(&f,&N))
    printf("%d", N);
  }
}
```

O programa armazena uma sequência de números inteiros e imprime, no final, a ordem de inserção na fila.

Estrutura de Dados Fila

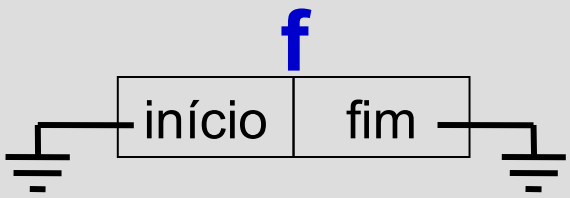
Implementação com Alocação Dinâmica

```
main( )
{
  fila f;
  int N;
  create(&f);
  insert(&f,12);
  insert(&f,320);
  insert(&f,413);
  printf("Ordem de chegada");

  if (!isEmpty(f))
  {
    removee(&f,&N))
    printf("%d", N);
  }
}
```

Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
  fila f;
  int N;
  create(&f);
  insert(&f,12);
  insert(&f,320);
  insert(&f,413);
  printf("Ordem de chegada");

  if (!isEmpty(f))
  {
    removee(&f,&N))
    printf("%d", N);
  }
}
```



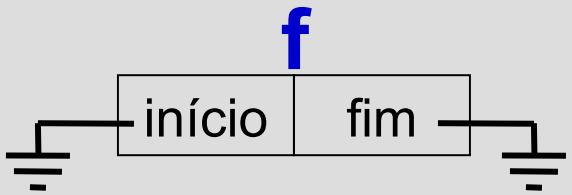

```
void create(fila *q)
{
 q->inicio=NULL;
 q->fim=NULL;
}
```

Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
   fila f;
   int N;
   create(&f);
   insert(&f,12);
   insert(&f,320);
   insert(&f,413);
   printf("Ordem de chegada");

   if (!isEmpty(f))
   {
      removee(&f,&N))
      printf("%d", N);
   }
}
```

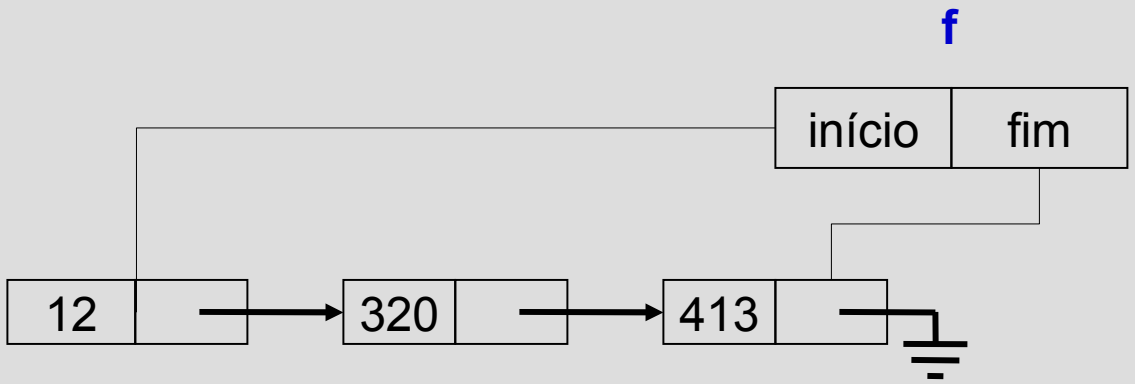
```
int insert (fila *q, int d)
{
   struct no *aux;
   aux = (struct no*)malloc(sizeof(struct no));
   if (aux==NULL)
       return (FALSE);

   aux->dado=d;
   aux->prox=NULL;

   if (q->inicio==NULL)
       q->inicio=aux;
   if (q->fim!=NULL)
       q->fim->prox=aux;

   q->fim=aux;
   return(TRUE);
}
```



Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
   fila f;
   int N;
   create(&f);
   insert(&f,12);
   insert(&f,320);
   insert(&f,413);
   printf("Ordem de chegada");

   if (!isEmpty(f))
   {
      removee(&f,&N))
      printf("%d", N);
   }
}
```



Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
  fila f;
  int N;
  create(&f);
  insert(&f,12);
  insert(&f,320);
  insert(&f,413);
  printf("Ordem de chegada");

  if (!isEmpty(f))
  {
    removee(&f,&N))
    printf("%d", N);
  }
}
```

```
int removee(fila *q, int *d)
{
 struct no *aux;
 if (q->inicio==NULL)
    return (FALSE);

 aux=q->inicio;
 q->inicio=aux->prox;
 *d=aux->dado;
 free(aux);
 return(TRUE);
}
```

Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
main( )
{
  fila f;
  int N;
  create(&f);
  insert(&f,12);
  insert(&f,320);
  insert(&f,413);
  printf("Ordem de chegada");

  if (!isEmpty(f))
  {
    removee(&f,&N))
    printf("%d", N);
  }
}
```

Estrutura de Dados Fila

Implementação com Alocação Dinâmica

```
int removee(fila *q, int *d)
{
 struct no *aux;
 if (q->inicio==NULL)
    return (FALSE);

 aux=q->inicio;
 q->inicio=aux->prox;
 if (q->inicio == NULL)
    q->fim = NULL;
 *d=aux->dado;
 free(aux);
 return(TRUE);
}
```

Implementação Correta!