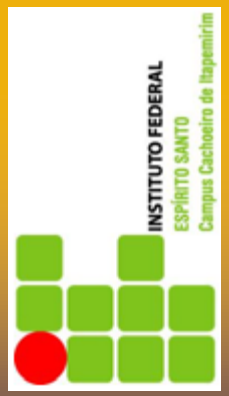
Programação 1
Filas

Rafael Vargas Mesquita

Filas

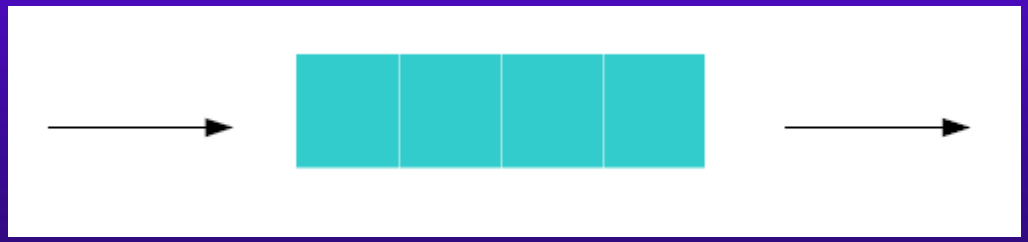
- Definição
  - A fila também é uma das estruturas de dados mais simples. O que diferencia a fila da pilha é a ordem dos elementos:
    - Na pilha “o último que entra é o primeiro que sai”.
    - Na fila “o primeiro que entra é o primeiro que sai”.
  - As filas são muito utilizadas na programação devido a sua simplicidade, um bom exemplo de uso de fila é a fila de impressão de arquivos no Windows.

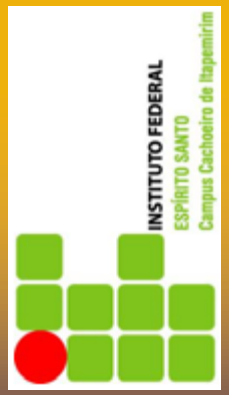


linguagem

Filas

- Representação Gráfica
  - Uma fila pode ser graficamente representada pela figura a seguir:



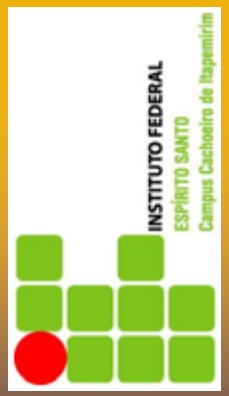


linguagem

Filas

- Formas de Implementar uma Fila
  - Podemos implementar uma fila de várias formas diferentes, por exemplo:
    - Fila implementada com vetor.
    - Fila implementada com lista simplesmente encadeada.

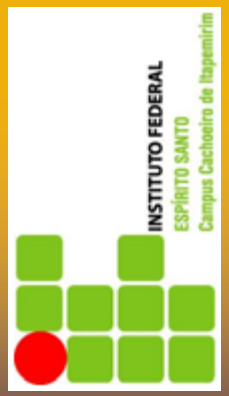
Obs.: Vamos abordar a implementação com lista



linguagem

Filas

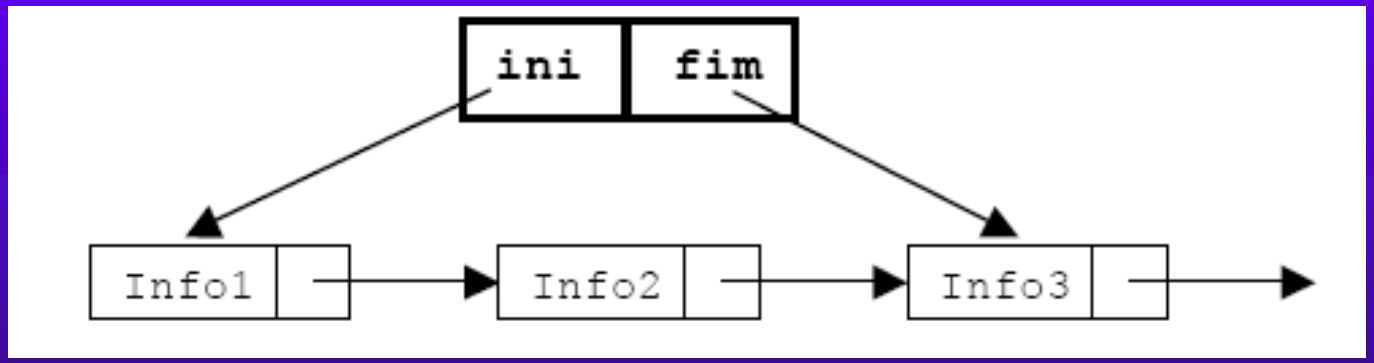
- Funcionalidades de uma Fila
  - Para utilizarmos uma fila é necessário implementar as seguintes funcionalidades:
    - – inicializa: inicializa a fila.
    - – insere: insere um elemento na fila.
    - – retira: retira um elemento da fila
    - – vazia: determina se a fila está vazia.
    - – libera: desaloca os elementos da fila.

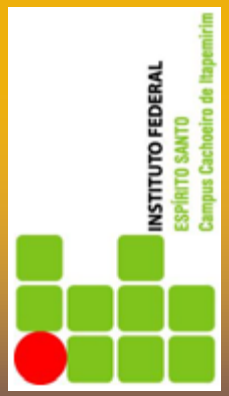


linguagem

Fila – Implementação com Lista

- A seguinte figura ilustra a estrutura de dados da fila com lista simplesmente encadeada:





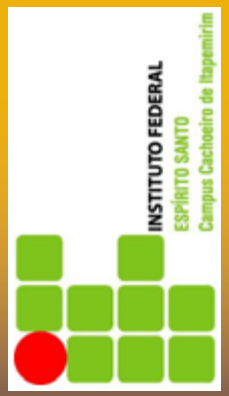
linguagem

Fila – Implementação com Lista

- Declaração

```
typedef struct no {
   int info;
   struct no* prox;
} No;

typedef struct fila {
   No* ini;
   No* fim;
} Fila;
```



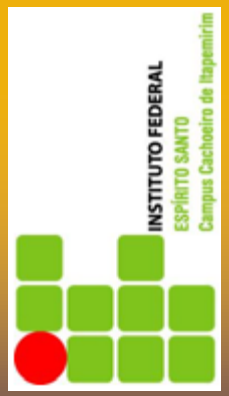
linguagem

Filas

- Funções
  - Função de Inicialização de uma Fila

```
/* função de inicialização: retorna fila vazia */
Fila* inicializa (){
  Fila *nova = (Fila *) malloc(sizeof(Fila));

  /* Inicializa os dados */
  nova->ini = NULL;
  nova->fim = NULL;
  return nova;
}
```



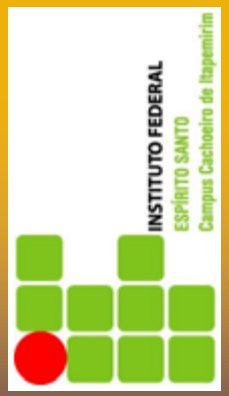
linguagem

Filas

- Funções
  - Função de Inserção de Elementos na Fila

```
// insere novo elemento na fila (sempre no final)
void insere (Fila *f, int v){
  No *novo = (No*) malloc(sizeof(No));
  novo->info = v;
  novo->prox = NULL;
  if (f->fim != NULL)
   f->fim->prox = novo;

  f->fim = novo;
  if (f->ini==NULL) /* fila antes vazia? */
   f->ini = f->fim;
}
```



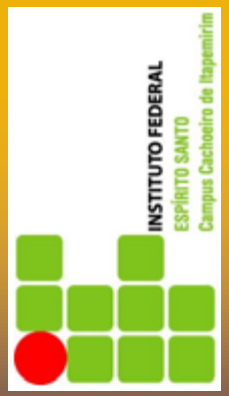
Filas

- Funções
  - Função de Remoção de Elemento da Fila

```
// retira um elemento da fila (sempre do inicio)
int retira (Fila* f){
   No* novo_ini;
   No* removido;
   int valor;

   removido = f->ini;
   novo_ini = f->ini->prox;

   valor = f->ini->info;
   f->ini = novo_ini;
   if (f->ini == NULL) /* fila ficou vazia? */
        f->fim = NULL;
   free(removido);
   return valor;
}
```



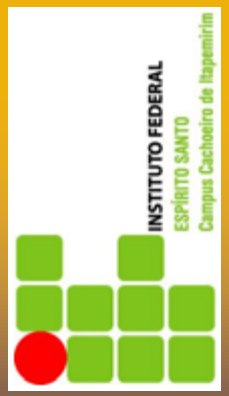
linguagem

Filas

- Funções
  - Função de Impressão de uma Fila

```
/* função imprime: imprime valores dos elementos */
void imprime (Fila *f){
   No *aux; /* variável auxiliar para percorrer a fila */

   for (aux = f->ini; aux != NULL; aux = aux->prox)
       printf("\t\tInfo = %d\n", aux->info);
}
```

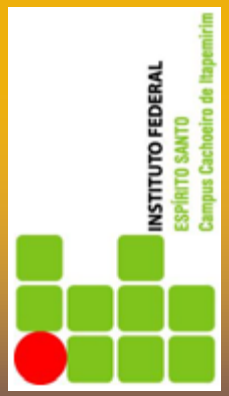


linguagem

Filas

- Funções
  - Função de Verificação de Fila Vazia

```
/* função vazia: retorna 1 se vazia ou 0 se não vazia */
int vazia (Fila *f){
   return (f->ini == NULL);
}
```

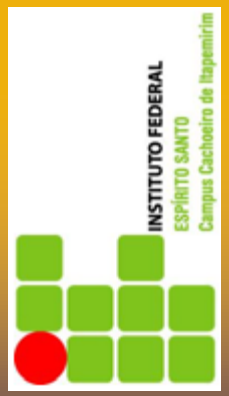


linguagem

Filas

- Funções
  - Função de Liberação de uma Fila

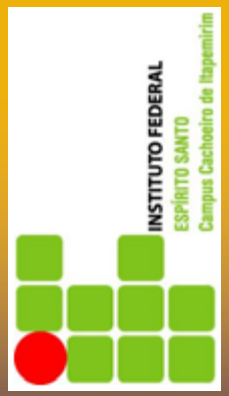
```
void libera (Fila* f){
  No* aux = f->ini;
  while (aux != NULL) {
    /* guarda referência para o próximo elemento */
    No* t = aux->prox;
    free(aux); /* libera a memória apontada por aux */
    aux = t; /* faz p apontar para o próximo */
  }
}
```



Filas

- Funções
  - A seguir um pequeno trecho de código utilizando as funcionalidades descritas até esse slide:

```
#include <stdio.h>
int main(){
   Fila *fila;
   fila = inicializa();
   insere(fila, 10);
   insere(fila, 20);
   printf("Primeiro elemento: %d \n", retira(fila));
   printf("Segundo elemento: %d \n", retira(fila));
   libera(pilha);
}
```



linguagem

Bibliografia

- SANTOS, Henrique José. Curso de Linguagem C da UFMG, apostila.
- FORBELLONE, André Luiz. Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados. São Paulo: MAKRON, 1993.