# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação cria_fila() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
void cria_fila (FILA_SEQ *f)
{
   f->N = f->INICIO = 0;
   f->FIM = -1;
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação eh_vazia() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
int eh_vazia (FILA_SEQ *f)
{
   return (!f->N);
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação tam() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
int tam (FILA_SEQ *f)
{
   return (f->N);
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação ins() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
void ins (FILA_SEQ *f, int v)
{
   if (f->N == MAX)
   {
      printf ("\nERRO! Estouro na fila.\n");
      exit (1);
   }
   f->FIM= ((f->FIM)+1) % MAX;
   f->val[f->FIM]=v;
   f->N++;
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação cons() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
int cons (FILA_SEQ *f)
{
   if (eh_vazia(f))
   {
      printf ("\nERRO! Consulta na fila
   vazia.\n");
      exit (2);
   }
   else
      return (f->val[f->INICIO]);
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação ret() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
void ret (FILA_SEQ *f)
{
   if (eh_vazia(f))
   {
      printf ("\nERRO! Retirada na fila
   vazia.\n");
      exit (3);
   }
   else
   {
      f->INICIO= (f->INICIO+1) % MAX;
      f->N--;
   }
}
```

# Fila - Alocação Sequencial

Com base no que foi visto implemente a operação cons_ret() que compõem o TAD FILA_SEQ.

```
typedef struct
{
    int N;
    int INICIO;
    int FIM;
    int val[MAX];
}FILA_SEQ;
void cria_fila (FILA_SEQ *);
int eh_vazia (FILA_SEQ *);
int tam (FILA_SEQ *);
void ins (FILA_SEQ *, int);
int cons (FILA_SEQ *);
void ret (FILA_SEQ *);
int cons_ret (FILA_SEQ *);
```

```
int cons_ret (FILA_SEQ *f)
{
   if (eh_vazia(f))
   {
      printf ("\nERRO! Consulta e retirada na
    fila vazia.\n");
      exit (4);
   }
   else
   {
      int v=f->val[f->INICIO];
      f->INICIO= (f->INICIO+1) % MAX;
      f->N--;
      return (v);
   }
}
```

## Alocação Sequencial - Exercício

Implemente, no TAD FILA_SEQ, utilizando recursividade, a seguinte operação:

void gera_fila (FILA_SEQ *f, int m, int n);

a qual utilizando-se das operações do TAD FILA_SEQ produz uma fila de inteiros correspondente a [m..n].



UNIVASF

## Alocação Encadeada

Com já discutimos, a alocação sequência apresenta algumas desvantagens. Em virtude disso, podemos nós utilizar de uma lista encadeada para armazenarmos uma fila.

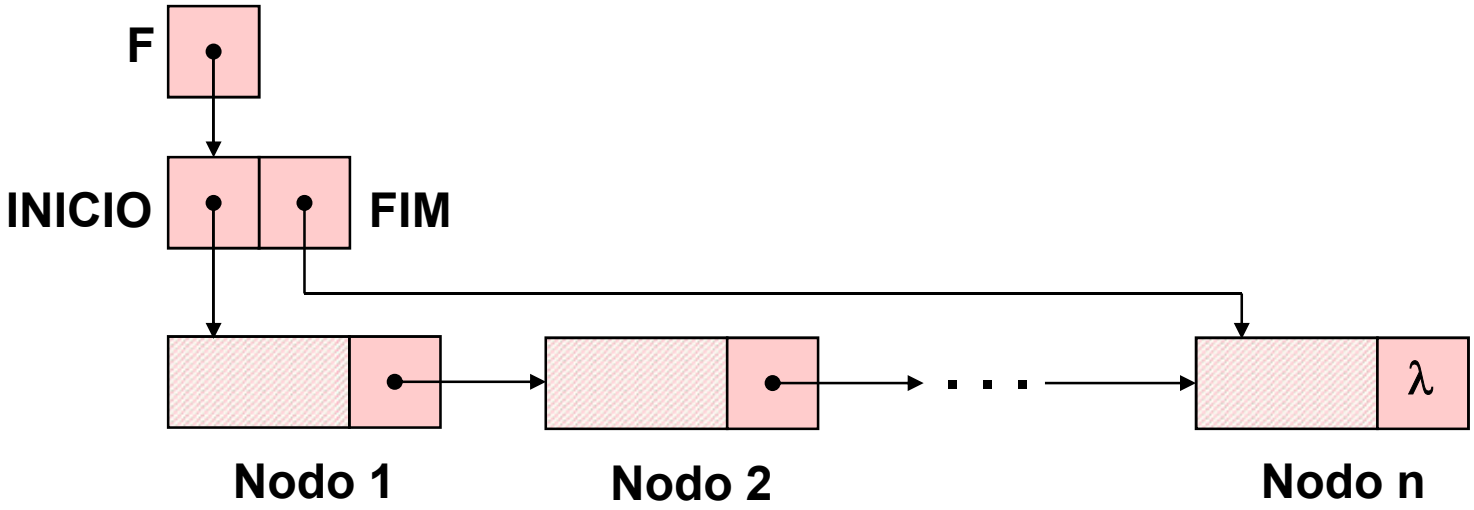
Como operaremos em ambas as extremidades da lista, devemos facilitar o acesso ao último nodo.

Uma estratégia, muito utilizada, é a utilização de uma representação baseada em um descritor contendo duas referências, ao primeiro e ao último nodo.



UNIVASF

# Alocação Encadeada




Desta forma, definiremos e implementaremos, agora, o TAD FILA_ENC (de valores inteiros).

**Obs.**: Se para a aplicação se fizer relevante o descritor pode armazenar, também, o número de elementos na fila.

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef struct
{
   NODO *INICIO;
   NODO *FIM;
}DESCRITOR;
typedef DESCRITOR * FILA_ENC;
void cria_fila (FILA_ENC *);
int eh_vazia (FILA_ENC);
void ins (FILA_ENC, int);
int cons (FILA_ENC);
void ret (FILA_ENC);
int cons_ret (FILA_ENC);
```