Alocação Encadeada

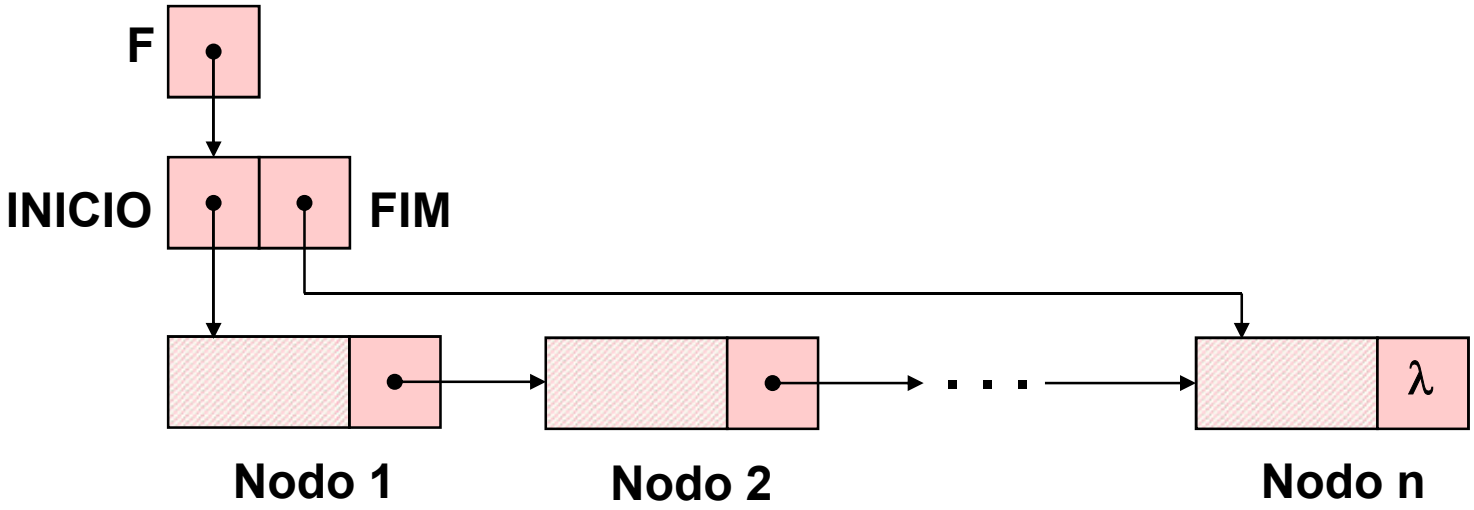
Com já discutimos, a alocação sequência apresenta algumas desvantagens. Em virtude disso, podemos nós utilizar de uma lista encadeada para armazenarmos uma fila.

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);
```

Fila - Alocação Encadeada

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

```
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);
```

```
void cria_fila (FILA_ENC *pf)
{
   *pf=(DESCRITOR *)malloc(sizeof(DESCRITOR));
   if (!*pf)
   {
      printf ("\nERRO! Memoria insuficiente!\n");
      exit (1);
   }
   (*pf)->INICIO=(*pf)->FIM=NULL;
}
```

Fila - Alocação Encadeada

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

```
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);
```

```
int eh_vazia (FILA_ENC f)
{
   return (f->INICIO == NULL);
}
```

Fila - Alocação Encadeada

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

```
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);
```

```
void ins (FILA_ENC f, int v) {
   NODO *novo;
   novo = (NODO *) malloc (sizeof(NODO));
    if (!novo)
    {
       printf ("\nERRO! Memoria insuficiente!\n");
       exit (1);
    }
   novo->inf = v;
   novo->next = NULL;
   if (eh_vazia(f))
      f->INICIO=novo;
   else
      f->FIM->next=novo;
   f->FIM=novo;
}
```

Fila - Alocação Encadeada

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

```
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);
```

```
int cons (FILA_ENC f)
{
   if (eh_vazia(f))
   {
      printf ("\nERRO! Consulta em fila vazia!\n");
      exit (2);
   }
   else
      return (f->INICIO->inf);
}
```

Fila - Alocação Encadeada

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

```
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);
```

```
void ret (FILA_ENC f) {
   if (eh_vazia(f))
   {
      printf ("\nERRO! Retirada em fila vazia!\n");
      exit (3);
   }
   else {
      NODO *aux=f->INICIO;
      f->INICIO=f->INICIO->next;
      if (!f->INICIO)
         f->FIM=NULL;
      free (aux);
     }
}
```

Fila - Alocação Encadeada

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

```
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);
```

```
int cons_ret (FILA_ENC f) {
   if (eh_vazia(f)){
      printf ("\nERRO! Consulta e retirada em fila vazia!\n");
      exit (4);
   }
   else {
      int v=f->INICIO->inf;
      NODO *aux=f->INICIO;
      f->INICIO=f->INICIO->next;
      if (!f->INICIO)
         f->FIM=NULL;
      free (aux);
      return (v);
   }
}
```

Alocação Encadeada

Como exercício, baseando-se no TAD anterior, defina e implemente o TAD FILA_ENC (de valores inteiros). Onde o descritor deve armazenar o número de elementos na fila e teremos a operação que determinará o tamanho da fila.