Fila - Alocação Sequencial

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

```
 1  typedef struct
 2  {
 3      int N;
 4      int INICIO;
 5      int FIM;
 6      int val[MAX];
 7  }FILA_SEQ;
 8  void cria_fila (FILA_SEQ *);
 9  int eh_vazia (FILA_SEQ *);
10  int tam (FILA_SEQ *);
11  void ins (FILA_SEQ *, int);
12  int cons (FILA_SEQ *);
13  void ret (FILA_SEQ *);
14  int cons_ret (FILA_SEQ *);
```

```c
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.

```
 1  typedef struct
 2  {
 3      int N;
 4      int INICIO;
 5      int FIM;
 6      int val[MAX];
 7  }FILA_SEQ;
 8  void cria_fila (FILA_SEQ *);
 9  int eh_vazia (FILA_SEQ *);
10  int tam (FILA_SEQ *);
11  void ins (FILA_SEQ *, int);
12  int cons (FILA_SEQ *);
13  void ret (FILA_SEQ *);
14  int cons_ret (FILA_SEQ *);
```

```c
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.

```
1   typedef struct
2   {
3       int N;
4       int INICIO;
5       int FIM;
6       int val[MAX];
7   }FILA_SEQ;
8   void cria_fila (FILA_SEQ *);
9   int eh_vazia (FILA_SEQ *);
10  int tam (FILA_SEQ *);
11  void ins (FILA_SEQ *, int);
12  int cons (FILA_SEQ *);
13  void ret (FILA_SEQ *);
14  int cons_ret (FILA_SEQ *);
```

```c
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

```c
void gera_fila (FILA_SEQ *f, int m, int n)
{
   if (m>n)
   {
      printf ("\nERRO! Intervalo invalido.\n");
      exit (5);
   } else
      if (m==n){
         cria_fila (f);
         ins (f, m);
      } else {
         gera_fila (f, m, n-1);
         ins (f, n);
      }
}
```

Filas Encadeadas

UNIVASF

Fila - Alocação Encadeada

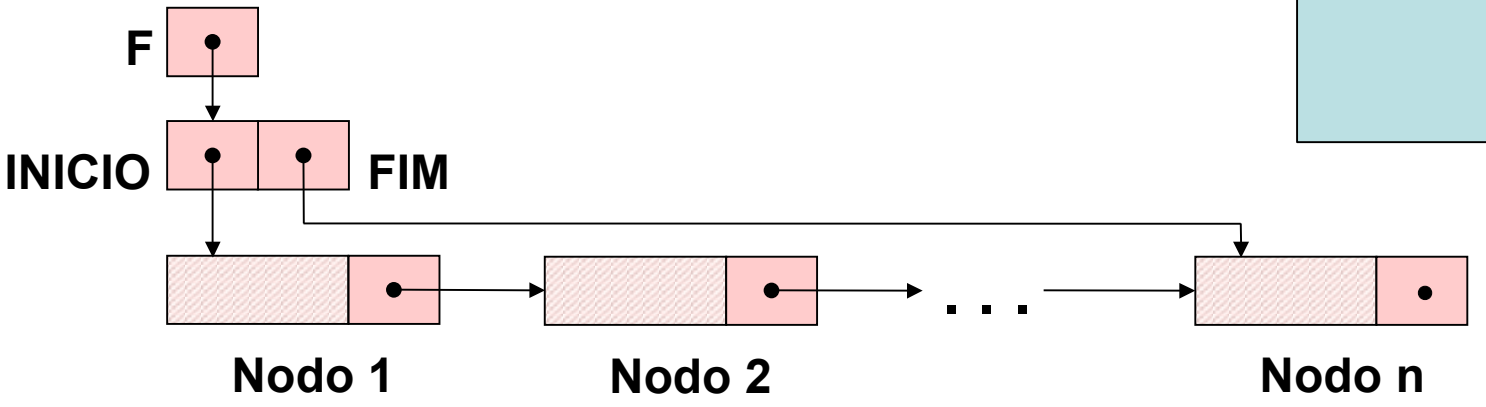
Como debatemos, a alocação sequência apresenta algumas desvantagens. Em virtude disso, podemos nós utilizar de uma lista encadeada dinâmica para armazenarmos uma fila.

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

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

UNIVASF

Fila - 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.

```c
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 destruir (FILA_ENC);
```

Fila - Alocação Encadeada

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

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

```c
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.

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

```c
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.

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

Fila - Alocação Encadeada

Dicas:

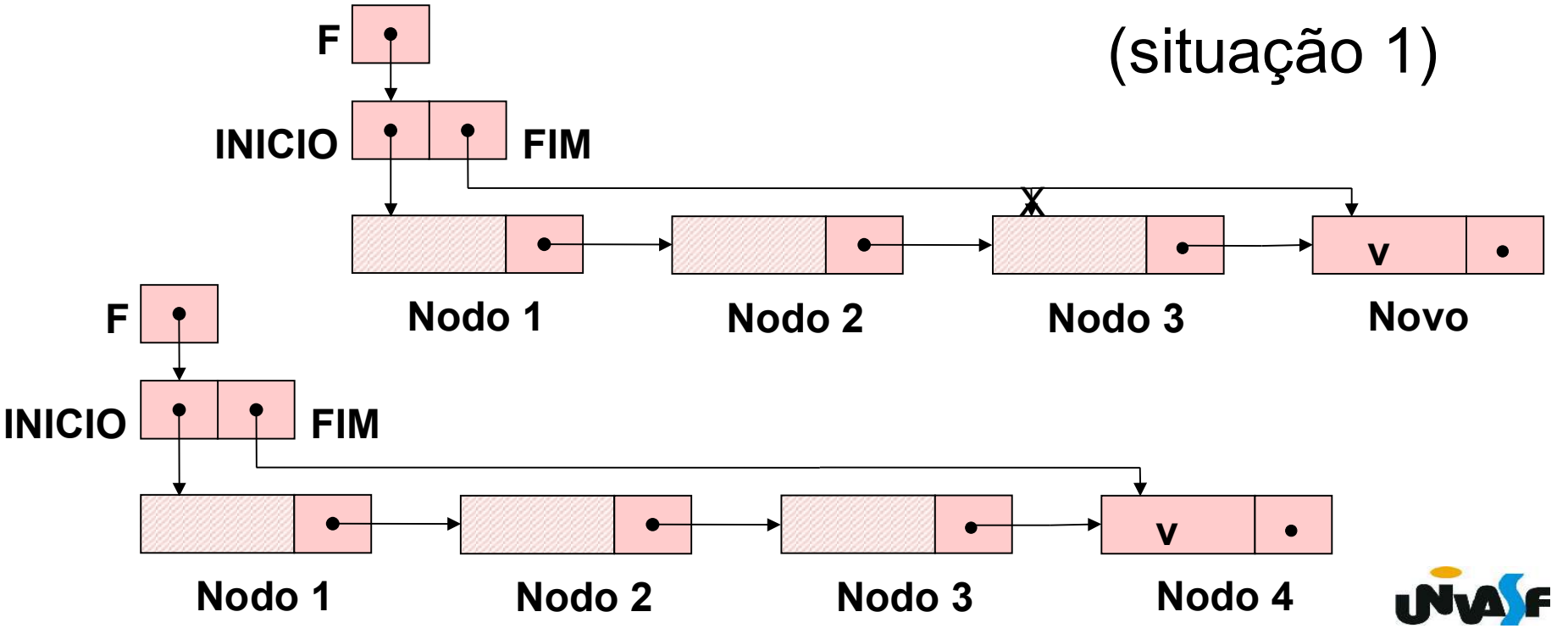
Tem espaço na memória para armazenar mais um elemento?

Todas as situações de inserção são tratadas da mesma forma?

UNIVASF

Fila - Alocação Encadeada

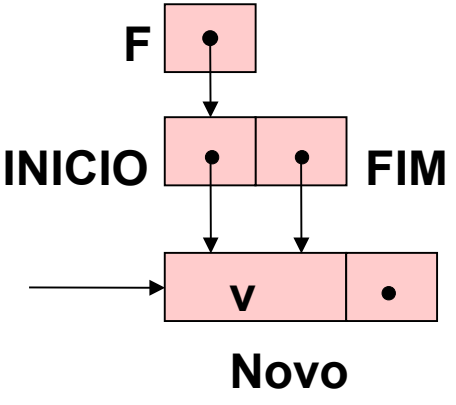
Esquema do processo de inserção de um elemento na fila encadeada.

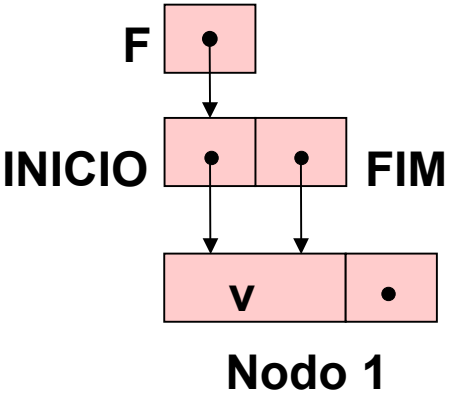


Fila - Alocação Encadeada

Esquema do processo de inserção de um elemento na fila encadeada.

(situação 2)





```c
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.

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

```c
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.

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

Fila - Alocação Encadeada

Dicas:

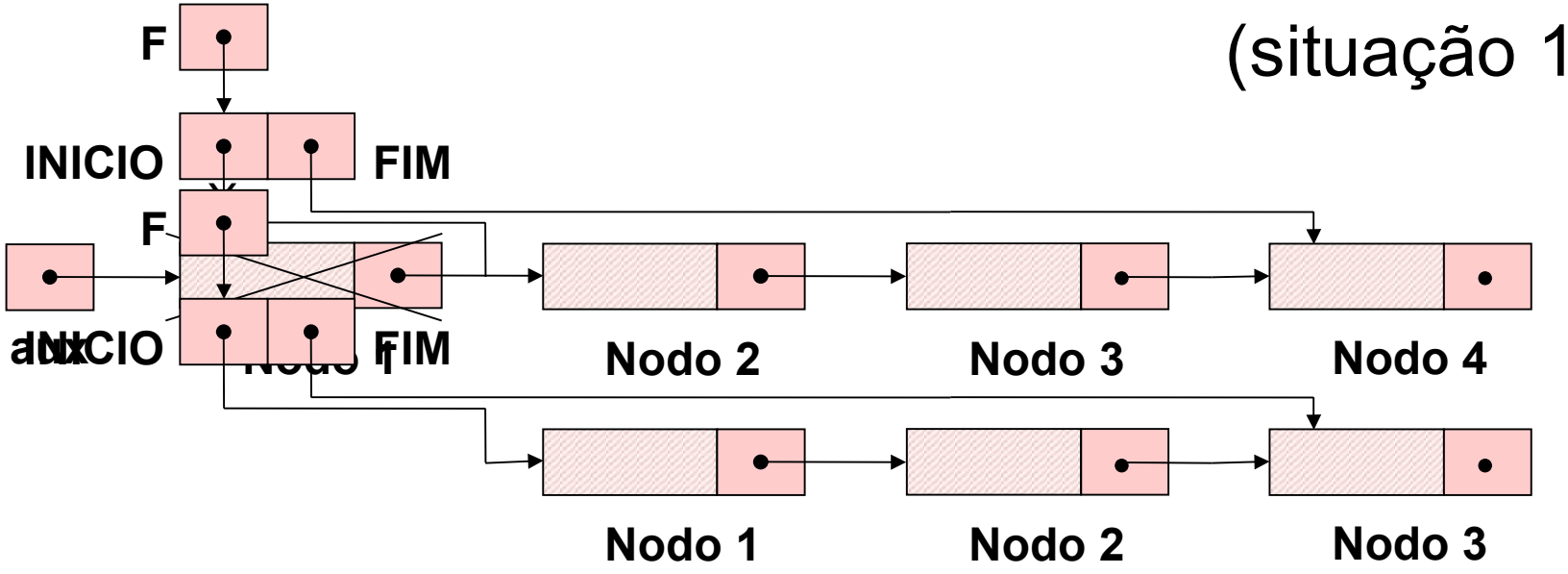
A FILA não é vazia?

Todas as situações de remoção são tratadas da mesma forma?

UNIVASF

Fila - Alocação Encadeada

Esquema do processo de retirada de um elemento na fila encadeada.

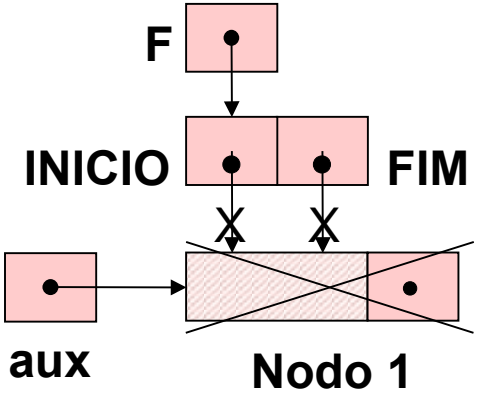


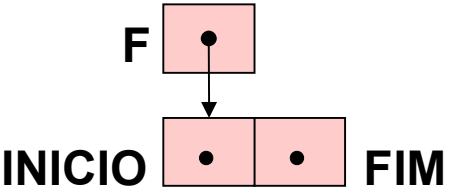
(situação 1)

Fila - Alocação Encadeada

Esquema do processo de retirada de um elemento na fila encadeada.

(situação 2)





```c
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.

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```

Fila - Alocação Encadeada

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

```
1   typedef struct nodo
2   {
3       int inf;
4       struct nodo * next;
5   }NODO;
6   typedef struct
7   {
8       NODO *INICIO;
9       NODO *FIM;
10  }DESCRITOR;
11  typedef DESCRITOR * FILA_ENC;
12  void cria_fila (FILA_ENC *);
13  int eh_vazia (FILA_ENC);
14  void ins (FILA_ENC, int);
15  int cons (FILA_ENC);
16  void ret (FILA_ENC);
17  int cons_ret (FILA_ENC);
18  void destruir (FILA_ENC);
```