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);
```

```c
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);
    }
}
```

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);
```

```c
void destruir (FILA_ENC f)
{
    NODO *aux;
    while (f->INICIO)
    {
        aux=f->INICIO;
        f->INICIO=f->INICIO->next;
        free(aux);
    }
    free(f);
}
```

Fila - 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.

```c
typedef struct nodo {
    int inf;
    struct nodo * next;
}NODO;
typedef struct {
    int ne;
    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);
int tam (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       int inf;
 3       struct nodo * next;
 4   }NODO;
 5   typedef struct {
 6       int ne;
 7       NODO *INICIO;
 8       NODO *FIM;
 9   }DESCRITOR;
10   typedef DESCRITOR * FILA_ENC;
11   void cria_fila (FILA_ENC *);
12   int eh_vazia (FILA_ENC);
13   void ins (FILA_ENC, int);
14   int cons (FILA_ENC);
15   void ret (FILA_ENC);
16   int cons_ret (FILA_ENC);
17   void destruir (FILA_ENC);
18   int tam (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;
    (*pf)->ne=0;
}
```

Fila - Alocação Encadeada

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

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

```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;
    f->ne++;
}
```

Fila - Alocação Encadeada

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

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

```c
void ret (FILA_ENC f)
{
    if (!f->INICIO)
    {
        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);
        f->ne--;
      }
}
```

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

```c
int cons_ret (FILA_ENC f)
{
    if (!f->INICIO)
    {
        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);
       f->ne--;
       return (v);
     }
}
```

Fila - Alocação Encadeada

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

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

```
1   int tam (FILA_ENC f)
2   {
3       return (f->ne);
4   }
```

Filas

Como vimos, uma fila nada mais é do que uma lista com uma disciplina de acesso.

Logo, podemos nos utilizar de todos os conceitos vistos em listas para implementarmos filas.

Por exemplo, podemos utilizar uma lista circular para armazenar uma fila.

Como exercício de fixação, implemente um TAD FILA, armazenado a mesma em uma lista circular.

UNIVASF

Exemplo de Aplicação de Filas

Filas – Exemplo de Aplicação

Uma aplicação interessante para filas é a ordenação por distribuição.

Seja uma lista l composta de n chaves, cada qual representada por um inteiro numa base $b > 1$. O problema consiste em ordenar essa lista.

O algoritmo utiliza b filas, denotadas por f_i, $0 \leq i \leq b-1$. Seja d o comprimento máximo da representação das chaves na base b. O algoritmo efetua d iterações, em cada uma das quais a tabela é percorrida.

Filas – Exemplo de Aplicação

A primeira iteração destaca, em cada nó, o dígito menos significativo da representação b-ária de cada chave. Se este for igual a k, a chave correspondente será inserida na fila f_k.

Ao terminar o percurso da tabela, esta se encontra distribuída pelas filas, que devem então ser concatenadas em sequência, isto é, f_0, depois f_1, f_2, etc.

Para essa tabela, já disposta numa ordem diferente da original, o processo deve ser repetido levando-se em consideração o segundo dígito da representação, ...

Filas – Exemplo de Aplicação

Observaremos agora um exemplo dessa ordenação, onde b=10 e d=2.

lista: **19 13 05 27 01 26 31 16 02 09 11 21 60 07**

Interação 1: 1ª distribuição (unidades simples)

lista: 19 13 05 27 01 26 31 16 02 09 11 21 60 07

$fila_0$: 60

$fila_1$: 01, 31, 11, 21

$fila_2$: 02

$fila_3$: 13

$fila_4$:

$fila_5$: 05

$fila_6$: 26, 16

$fila_7$: 27, 07

$fila_8$:

$fila_9$: 19, 09

lista: 60 01 31 11 21 02 13 05 26 16 27 07 19 09

Interação 2: 2ª distribuição (dezenas simples)

lista: ~~60 01 31 11 21 02 13 05 26 16 27 07 19 09~~

$fila_0$: 01, 02, 05, 07, 09

$fila_1$: 11, 13, 16, 19

$fila_2$: 21, 26, 27

$fila_3$: 31

$fila_4$:

$fila_5$:

$fila_6$: 60

$fila_7$:

$fila_8$:

$fila_9$:

lista: 01 02 05 07 09 11 13 16 19 21 26 27 31 60

Filas – Exemplo de Aplicação

Utilizando-se dos TAD´s LISTA_ENC e FILA_ENC, implemente a função ord_por_dist() a qual recebe como entrada uma referência para uma lista encadeada de valores inteiros e a ordena através do processo de ordenação por distribuição.

OBS: b=10 e d=4.