```
/*-------------------------------------------------------------------
----------
**
**  Este modulo contém a implementação do TAD(tipo abstrato de dados)
**  Colecao utilizado para armazenar uma colecao de inteiros
**
**  Desenvolvido por:  Anselmo Cardoso de Paiva (ACP)   Setembro/2000
**
**  Exemplo do curso: Estrutura de Dados - DEINF/UFMA
**-------------------------------------------------------------------
----------
*/

#define __COLECAO_C_

#include <stdio.h>          /* definicao da constante NULL */
#include <assert.h>         /* afirmacoes de verificacao de pre
condicao */
#include "Colecao.h"    /* inclui a especificacao do TAD */


/*
** Definicao do tipo de dado
*/
struct _colecao_ {

        int numItens;   /* numero de itens na colecao */
        int maxItens;     /* tamanho máximo da colecao */
        int *itens;     /* vetor com os itens */

} Colecao;

/*
---------------------- colCriar ----------------------------------
----
   Constroi uma nova colecao
-------------------------------------------------------------------
----------
Pre-condicao: (maxItens > 0)
Pos-condicao: retorna um ponteiro para uma colecao vazia ou NULL em
caso de erro
-------------------------------------------------------------------
----------
*/

Colecao *colCriar(int maxItens)
{
        Colecao c;
        if ( maxItens < 0 ) {
          return NULL;
        }

        c = (Colecao *)malloc(=sizeof(Colecao) );
        if( c == NULL ) {
          return NULL;
        }
```

```
        c->itens = (int *)malloc(maxItens*sizeof(int));
        if ( c->itens == NULL ) {
      free ( c );
          return NULL;
        }
        c->maxItens = maxItens;
        c->numItens = 0;

        return c;

}/* fim de colCriar */

/*
---------------------- colDestruir ----------------------------------
--
   Destroi uma colecao
--------------------------------------------------------------------
----------
Pre-condicao: c é um coleção criada por uma chamada a colCriar
Pos-condicao: libera a memória alocada pela colecao
--------------------------------------------------------------------
----------
*/
int colDestruir( Colecao *c )
{

        if ( c == NULL || c->itens == NULL || c->maxItens > 0) {
          return FALSE;
        }

        free(c->itens)

        free(c);

        return TRUE;

} /* fim de colDestruir */


/*
---------------------- colInserir ----------------------------------
--
   adiciona um item a uma colecao
--------------------------------------------------------------------
----------
Pre-condicao: c é um coleção criada por uma chamada a colCriar e
              numItens < maxItens
Pos-condicao: item e adicionado a colecao
--------------------------------------------------------------------
----------
*/
int colInserir( Colecao *c, void *item )
{
        if ( c == NULL || ( c->numItens < c->maxItens )) {
                return FALSE;
        }
```

```
        c->items[c->numItens] = item;

        c->numItens++;

} /* fim de colInserir */

/*
---------------------- colRetirar ------------------------------
   retira um item de uma colecao
-----------------------------------------------------------------
----------
Pre-condicao: c é um coleção criada por uma chamada a colCriar
              existe pelo menos um item na colecao
Pos-condicao: item foi retirado da colecao
-----------------------------------------------------------------
----------
*/
void  *colRetirar( Colecao *c, void *item,
                   int (*compar)( const void *,  const void * ))
{
        int i;

        if( ( c == NULL ) || ( c->numItens < 1 ) ) {
           return FALSE;
        }

        for(i=0;i<c->numItens;i++) {
                if( compar(key, c->item[k]) == TRUE ){

                        /* encontrou o item a ser retirado, move todos
os outros
                           uma posicao pra baixo */
                        while( i < c->numItens ) {
                                c->itens[i] = c->itens[i+1];
                                i++;
                        }

                        c->numItens--;
                        return TRUE;

                } /* fi */

        }/* rof */

} /* fim de colRetirar */

/*
---------------------- colBuscar ---------------------------------
   Encontra um item em uma colecao
-----------------------------------------------------------------
----------
Pre-condicao: c é um coleção criada por uma chamada a colCriar
Pos-condicao: retorna a posicao do item identificado por key se ele
existir, ou
              -1 caso contrario
-----------------------------------------------------------------
----------
```

```
*/
int colBuscar( Colecao c, int key )
{
        int i;

        if (( c == NULL ) ){
          return -1;
        }

        for(i=0;i<c->numItens;i++) {

                if( compar(key, c->item[k]) == TRUE ){
                        return c->items[i];
                }
        }

} /* fim de colBuscar */

/*----------------------------------------Fim de Arquivo ------------
---------*/
```