Atividade 5

```
/*****************************************************************************
//     pilhaEstatica.c
// Este programa gerencia PILHAs implementadas em arranjos
   (implementacao estatica).
// As PILHAs gerenciadas podem ter um numero de no maximo MAX elementos.
// Não usaremos sentinela nesta estrutura.
*****************************************************************************/
#include <stdio.h>
#include <malloc.h>  /* apenas para a funcao excluirElementoPilha2 */
#define true 1
#define false 0
#define MAX 50

typedef int bool;

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
} REGISTRO;

typedef struct {
  int topo;
  REGISTRO A[MAX];
} PILHA;


/* Inicialização da PILHA (a PILHA jah esta criada e eh apontada
pelo endereco em p) */
void inicializarPilha(PILHA* p){
  p->topo = -1;
} /* inicializarPILHA* /


/* Retornar o tamanho da pilha (numero de elementos) */
int tamanhoPilha(PILHA* p) {
  int tamanho = p->topo+1;
  return tamanho;
} /* tamanho */

/* Exibição da pilha */
void exibirPilha(PILHA* p){
  printf("Pilha: \" ");
  int i;
  for (i=p->topo;i>=0;i--){
    printf("%i ", p->A[i].chave); // soh lembrando TIPOCHAVE = int
  }
  printf("\"\n");
```

```
} /* exibirPilha* /

//************************** Atividade 5 **********************

/* Exibição da pilha (da base para o topo) */
void exibirPilhaInvertida(PILHA* p){
  printf("Pilha (da base para o topo): \" ");
  int i;
  for (i= 0;i <= p->topo;i++){
    printf("%i ", p->A[i].chave);
  }
  printf("\"\n");
} /* exibirPilhaInvertida* /

//************************** Atividade 5 **********************

/* Retornar o tamanho em bytes da pilha. Neste caso, isto nao depende do numero
   de elementos que estao sendo usados.   */
int tamanhoEmBytesPilha(PILHA* p) {
  return sizeof(PILHA);
} /* tamanhoEmBytes */

/* Busca Pilha - retorna posicao do primeiro elemento da pilha (topo) */
int buscaTopoDaPilha(PILHA* p){
  return p->topo;
} /* buscaTopoDaPilha * /

/* Destruição da PILHA */
void reinicializarPilha(PILHA* p) {
     p->topo = -1;
} /* destruirPILHA* /

/* inserirElementoPilha - insere elemento no fim da pilha   */
bool inserirElementoPilha(PILHA* p, REGISTRO reg){
     if (p->topo+1>= MAX) return false;
     p->topo = p->topo+1;
     p->A[p->topo] = reg;
     return true;
} /* inserirElementoPILHA* /

/* excluirElementoPilha - retorna e exclui 1o elemento da pilha
retorna false se nao houver elemento a ser retirado */
bool excluirElementoPilha(PILHA* p, REGISTRO* reg){
   if (p->topo == -1) return false;
   *reg = p->A[p->topo];
   p->topo = p->topo-1;
   return true;
} /* excluirElementoPILHA* /

/* retornarPrimeiroPilha
```

```
retorna a posicao do primeiro (topo) elemento da pilha e o valor de sua chave no
conteudo do endereco ch. Retorna -1 caso a pilha esteja vazia */
int retornarPrimeiroPilha(PILHA* p, TIPOCHAVE* ch){
    if (p->topo==-1)return -1;
    *ch = p->A[p->topo].chave;
    return p->topo;
}

/* excluirElementoPilha2 - versao alternativa da funcao excluir, que retorna
NULL caso a pilha esteja vazia; caso contrario aloca memoria e copia o
registro do 1o elemento da pilha, exclui este elemento e retorna o endereco do
registro alocado. */
REGISTRO* excluirElementoPilha2(PILHA* p){
   if (p->topo == -1) return NULL;
   REGISTRO* res = (REGISTRO*) malloc(sizeof(REGISTRO));
   *res = p->A[p->topo];
   p->topo = p->topo-1;
   return res;
} /* excluirElementoPILHA2*/

/* buscaSequencial - realiza uma busca sequencial na pilha (a partir do topo)
   e retorna a posicao do registro buscado na pilha, se encontrar, ou -1 caso
   contrário */
int buscaSequencial(PILHA* p, TIPOCHAVE ch){
  int i = p->topo;
  while (i>=0){
if (p->A[i].chave == ch) return i;
i--;
  }
  return -1;
} /* buscaSequencial */
```