**2 - 1** Implemente uma pilha com vetor, conforme a interface dada *pilha.h*:

1. O TAD TPilha, com vetor de 100 elementos e a variável com a posição do topo;
2. Operação de criar;
3. Operação de empilhar;
4. Operação de desempilhar;
5. Operação que retorna tamanho;
6. Operação que retorna topo.

|  |
| --- |
| // stack.h  typedef struct Stack TStack;  TStack \*neww(int n);  void stackUp(TStack \*stack, int x);  int unStack(TStack \*stack);  int size(TStack \*stack);  int top(TStack \*stack);  // stack.c  #include <stdio.h>  #include <stdlib.h>  #include "stack.h"  struct Stack {  int \*vector;  int ts, qtd, top;  };  TStack \*neww(int n) {  TStack \*stack = (TStack \*)malloc(sizeof(TStack));  stack -> vector = malloc(sizeof(int) \*n);  stack -> qtd = n;  stack -> ts = -1;  return stack;  }  void stackUp(TStack \*stack, int x) {  if(stack -> ts < stack -> qtd -1) {  stack -> ts = stack -> ts +1;  stack -> vector[stack -> ts] = x;  }  }  int unStack(TStack \*stack) {  if(stack -> ts > -1) {  stack -> ts = stack -> ts -1;  return stack -> vector[stack -> ts + 1];  }  else  return -1;  }  int size(TStack \*stack) {  return stack -> qtd;  }  int top(TStack \*stack) {  return stack->vector[stack->top];  } |

**2 - 2** Inclua na interface pilha.h e implemente a operação “esta\_vazia()”, a qual retorna 0 se a pilha não estiver vazia e 1 em caso contrário.

|  |
| --- |
| // stack.h  int stackEmpty(TStack \*stack);  // stack.c  int stackEmpty(TStack \*stack) {  if(stack->top == -1) {  return 0;  }  return 1;  } |

**2 - 3** Desenvolva uma função que receba uma TPilha como parâmetro e inverta seu conteúdo, utilizando uma pilha auxiliar.

|  |
| --- |
| // stack.h  void \*reverseVector(TStack \*stack);  // stack.c  void reverse(TStack \*stack) {  TStack \*stackAux = neww(stack->qtd);  TStack \*stackAux2 = neww(stack->qtd);  int i,j, index;  for(i = 0; i <= stack->qtd; i++) {  stackUp(stackAux,unStack(stack));  }  for(j = 0; j <= stack->qtd; j++) {  stackUp(stackAux2,unStack(stackAux));  }  for(index = 0; index <= stack->qtd; index++) {  stackUp(stack,unStack(stackAux2));  }  } |

**2 - 4** Desenvolva uma função que receba uma TPilha e um valor n como parâmetros e remova a primeira ocorrência de n da pilha, se houver. A função poderá utilizar uma pilha auxiliar e só poderá utilizar as operações de pilha para manipular seus elementos.

|  |
| --- |
| // stack.h  void removeValue(TStack \*stack, int value);  // stack.c  void removeValue(TStack \*stack, int value) {  if(stackEmpty(stack) == 1) {  TStack \*stackAux = neww(stack->qtd);  int i,j;  for(i = 0; i <= stack->qtd; i++) {  if(stackTop(stack) == value) {  unStack(stack);  }  stackUp(stackAux,unStack(stack));  }  for(j = 0; j <= stack->qtd; j++) {  stackUp(stack, unStack(stackAux));  }  }  } |