**ÚLTIMA ACTUALIZACIÓN: 04/06/2021 00:30**

**prueba 3**

|  |
| --- |
| **#include <stdio.h>** **#include <stdlib.h>** */\*\** ***C O L A***  *\*/*  */\*-----------------------------\*/* **struct** **sNodo**{  **int** valor;  **struct** **sNodo** \***sig**; }; **typedef** **struct** **sNodo**\* **tNodo**;  */\*-----------------------------------------\*/* */\* \*\* con un puntero (sig) \*\* \*/* **struct** **s\_cola**{  tNodo frente; *//head (primero)*  tNodo final; *//tail (ultimo )* }; **typedef** **struct** **s\_cola** **t\_cola**;  */\*-----------------------------------------------------------------------\*/*  **int** **desencolar**(tNodo\* nodo); *// con un puntero (sig)*  **void** **encolar**(tNodo\*, **int**); *// con un puntero (sig)*  */\*-----------------------------\*/* **void** **queue**(t\_cola\*, **int**); *// con dos punteros (frente y final)* **int** **dequeue**(t\_cola\*); *// con dos punteros (frente y final)* */\*-----------------------------\*/* *//OJO SOLO FINES DE PRUEBA. NO SE PUEDE USAR EN LOS EXÁMENES* **void** **inspeccionarCola**(tNodo cola); *// con un puntero (sig)*  **void** **inspeccionarCola2**(t\_cola cola); *// con dos punteros (frente y final)*  */\*-----------------------------------------------------------------------\*/*  **void** **encolar**(tNodo\* nodo, **int** valor){  *// AGREGAR AL FINAL*   **if** ( \*nodo == NULL) {  \*nodo = (tNodo) malloc(**sizeof**(struct sNodo));  (\*nodo)->valor = valor;  (\*nodo)->sig = NULL;  }  **else** {  encolar(&(\*nodo)->sig, valor);  } }  **int** **desencolar**(tNodo\* nodo) {  *// ELIMINAR EL NODO DE ADELANTE (el primero)*    tNodo aux=NULL;  **int** valor=0;  **if**(\*nodo != NULL) {  aux = \*nodo;  valor= (\*nodo)->valor;  \*nodo=(\*nodo)->sig;  free(aux);  aux=NULL;  }  **return** valor; }  **void** **inspeccionarCola**(tNodo cola) {  *// OJOOOO NO HACER sólo lo usamos para inspeccionar el interior de la cola*  **while**(cola!=NULL) {  printf("\n cola=%d, <%p>",cola->valor,cola);  cola=(cola)->sig;  } }  **void** **inspeccionarCola2**(t\_cola cola) {  *// OJOOOO NO HACER sólo lo usamos para inspeccionar el interior de la cola*  tNodo aux = cola.frente;  **while**(aux!=NULL) {  printf("\n cola=%d, <%p>",aux->valor,aux);  aux=(aux)->sig;  } }  **void** **queue**(t\_cola\* cola, **int** valor){  *// con dos punteros (frente y final)*  tNodo aux = (tNodo) malloc(**sizeof**(struct sNodo));  aux->valor = valor;  aux->sig = NULL;  **if** (cola->frente == NULL && cola->final == NULL){  cola->frente = aux;  cola->final = aux;  }  **else** {  cola->final->sig = aux;  cola->final = aux;  } }  **int** **dequeue**(t\_cola\* cola){  **int** valor;  tNodo aux = cola->frente;  cola->frente = cola->frente->sig;  valor = aux->valor;  free(aux);  **if** (cola->frente == NULL) {  cola->final = NULL;  }  **return** valor; }  **int** **main**() {  */\*----------------------------\*/*  **unsigned** **int** n,c;  tNodo cola = NULL; *// con un puntero (sig)*   t\_cola cola2 = {NULL,NULL}; *// con dos punteros (frente y final)*   */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("ENCOLAR: con un puntero (sig)\n");  printf(" encolar() Agrega valores [6,3,26,23]\n");  printf("LUEGO: Hago una inspección de la cola\n");  printf(" Recordar que ispección no es una funcion valida\n");    printf("\n cola <%p>, antes de agregar",cola);  encolar(&cola, 6);  encolar(&cola, 3);  encolar(&cola, 26);  encolar(&cola, 23);  inspeccionarCola(cola); *// OJOOOO NO HACER EN EXAMEN*   printf("\n cola <%p> una vez completa",cola);    printf("\n--------------------------------------------------\n");  */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("DESENCOLAR: con un puntero (sig)\n");  printf(" desencolar() extrae el valor y lo retorna\n");  printf("ALGORITMO: vacia toda la cola\n");  **int** val;  tNodo dir;   **while**(cola!=NULL){  dir = cola;  val = desencolar(&cola);   printf("\n cola=%d, <%p>",val,dir);  }     printf("\n--------------------------------------------------\n");  */\*--------------------------------------------------\*/*   printf("\n\n==================================================\n");  printf("ENCOLAR: con dos punteros (frente y final)\n");  printf(" queue() Agrega valores [16,13,126,123]\n");  printf("LUEGO: Hago una inspección de la cola\n");  printf(" Recordar que ispección no es una funcion valida\n");    printf("\n cola.frente <%p>, cola.final <%p>, antes de agregar",cola2.frente,cola2.final);  queue(&cola2, 16);  queue(&cola2, 13);  queue(&cola2, 126);  queue(&cola2, 123);  inspeccionarCola2(cola2); *// OJOOOO NO HACER EN EXAMEN*  printf("\n cola.frente <%p>, cola.final <%p>, una vez completa",cola2.frente,cola2.final);    */\*--------------------------------------------------\*/*  printf("\n\n==================================================\n");  printf("DESENCOLAR: con dos punteros (frente y final)\n");  printf(" dequeue() extrae el valor y lo retorna\n");  printf("ALGORITMO: vacia toda la cola\n");  **int** val2;  t\_cola dir2;  **while**(cola2.final!=NULL && cola2.frente!=NULL){  dir2 = cola2;  val2 = dequeue(&cola2);  printf("\n cola=%d, <%p>",val2,dir2);  }   **return** 0; } |

**SALIDA**

|  |
| --- |
| ================================================== DESENCOLAR: con un puntero (sig)  desencolar() extrae el valor y lo retorna ALGORITMO: vacia toda la cola   cola=6, <00E51B70>  cola=3, <00E51B80>  cola=26, <00E51B90>  cola=23, <00E51BA0> --------------------------------------------------   ================================================== ENCOLAR: con dos punteros (frente y final)  queue() Agrega valores [16,13,126,123] LUEGO: Hago una inspección de la cola  Recordar que ispección no es una funcion valida   cola.frente <00000000>, cola.final <00000000>, antes de agregar  cola=16, <00E51B70>  cola=13, <00E51B80>  cola=126, <00E51B90>  cola=123, <00E51BA0>  cola.frente <00E51B70>, cola.final <00E51BA0>, una vez completa  ================================================== DESENCOLAR: con dos punteros (frente y final)  dequeue() extrae el valor y lo retorna ALGORITMO: vacia toda la cola   cola=16, <00E51B70>  cola=13, <00E51B80>  cola=126, <00E51B90>  cola=123, <00E51BA0>  << El programa ha finalizado: codigo de salida: 0 >> << Presione enter para cerrar esta ventana >> |