75.41 Algortimos y Programación II Curso 4
TDA Pila

Implementada como vector dinámico

15 de septiembre de 2019

1. Enunciado

Se pide implementar una Pila como vector dinámico. Para ello se brindan las firmas de las funciones públicas a implementar y se deja a criterio del alumno la creación de las funciones privadas del TDA para el correcto funcionamiento de la Pila cumpliendo con las buenas prácticas de programación.

2. pila.h

```c
#ifndef __PILA_H__
#define __PILA_H__

#include <stdbool.h>

typedef struct pila {
        int tope;
        void** elementos;
        int tamanio;
} pila_t;

/*
 * Crea una pila, reservando la memoria necesearia para almacenar la
 * estructura.
 * Devuelve un puntero a la estructura pila_t creada o NULL si no pudo
 * crearla.
 */
pila_t* pila_crear();

/*
 * Apila un elemento.
 * Devuelve 0 si pudo o -1 en caso contrario.
 */
int pila_apilar(pila_t* pila, void* elemento);

/*
 * Desapila un elemento.
 * Devuelve 0 si pudo desapilar o -1 si no pudo.
 */
int pila_desapilar(pila_t* pila);

/*
 * Determina si la pila está vacia.
 * Devuelve true si está vacía, false en caso contrario.
 * Si la pila no existe devolverá true.
 */
bool pila_vacia(pila_t* pila);

/*
 * Devuelve la cantidad de elementos almacenados en la pila.
 * Si la pila no existe devolverá 0.
 */
int pila_cantidad(pila_t* pila);

/*
 * Devuelve el elemento en el tope de la pila o NULL
 * en caso de estar vacía.
```

```c
 * Si la pila no existe devolverá NULL.
 */
void* pila_tope(pila_t* pila);

/*
 * Destruye la pila liberando la memoria reservada para la misma.
 */
void pila_destruir(pila_t* pila);

#endif /* __PILA_H__ */
```

3. Compilación y Ejecución

El TDA entregado deberá compilar y pasar las pruebas dispuestas por la cátedra sin errores, adicionalmente estas pruebas deberán ser ejecutadas sin pérdida de memoria.

Compilación:

```
gcc *.c -o pila_vd -g -std=c99 -Wall -Wconversion -Wtype-limits -pedantic -Werror -O0
```

Ejecución:

```
valgrind --leak-check=full --track-origins=yes --show-reachable=yes ./pila_vd
```

4. Minipruebas

Se les brindará un lote de minipruebas, las cuales recomendamos fuertemente sean ampliadas ya que no son exhaustivas y no prueban los casos bordes, solo son un ejemplo de como apilar, desapilar y qué debería verse en la terminal en el **caso feliz**.

Cabe aclarar que el criterio de redimensionamiento de la pila queda a criterio del alumno.

En esta implementación, la pila se redimendiona al doble de tamaño.

Minipruebas:

```c
#include "pila.h"
#include <stdio.h>

int main(){
	pila_t* pila = pila_crear();

	char elemento1 = '!';
	pila_apilar(pila, &elemento1);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	char elemento2 = '2';
	pila_apilar(pila, &elemento2);
	char elemento3 = 'o';
	pila_apilar(pila, &elemento3);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	char elemento4 = 'g';
	pila_apilar(pila, &elemento4);
	char elemento5 = 'l';
	pila_apilar(pila, &elemento5);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	char elemento6 = 'A';
	pila_apilar(pila, &elemento6);


	printf("%c\n", *(char*)pila_tope(pila));
	pila_desapilar(pila);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	printf("%c\n", *(char*)pila_tope(pila));
	pila_desapilar(pila);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	printf("%c\n", *(char*)pila_tope(pila));
	pila_desapilar(pila);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	printf("%c\n", *(char*)pila_tope(pila));
	pila_desapilar(pila);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
	printf("%c\n", *(char*)pila_tope(pila));
	pila_desapilar(pila);
	printf("Tamanio pila: %i\n", pila_cantidad(pila));
```

```c
        printf("%c\n", *(char*)pila_tope(pila));
        pila_desapilar(pila);
        printf("Tamanio pila: %i\n", pila_cantidad(pila));

        pila_destruir(pila);
        return 0;
}
```

La salida por pantalla luego de correrlas con valgrind debería ser:

```
==25623== Memcheck, a memory error detector
==25623== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==25623== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==25623== Command: ./pila_vd
==25623==
Tamanio pila: 1
Tamanio pila: 3
Tamanio pila: 5
A
Tamanio pila: 5
l
Tamanio pila: 4
g
Tamanio pila: 3
o
Tamanio pila: 2
2
Tamanio pila: 1
!
Tamanio pila: 0
==25623==
==25623== HEAP SUMMARY:
==25623==     in use at exit: 0 bytes in 0 blocks
==25623==   total heap usage: 3 allocs, 3 frees, 1,128 bytes allocated
==25623==
==25623== All heap blocks were freed -- no leaks are possible
==25623==
==25623== For counts of detected and suppressed errors, rerun with: -v
==25623== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

5. Entrega

La entrega deberá contar con todos los archivos necesarios para compilar y ejecutar correctamente el TDA.

Dichos archivos deberán formar parte de un único archivo **.zip** el cual será entregado a través de la plataforma de corrección automática **Kwyjibo**.

El archivo comprimido deberá contar, además del TDA con:

- El archivo con las pruebas agregadas para comprobar el correcto funcionamiento del TDA.
- Un **Readme.txt** donde se deberá explicar qué es lo entregado, como compilarlo (línea de compilación), como ejecutarlo (línea de ejecución) y todo lo que crea necesario aclarar.
- El enunciado.