75.41 Algortimos y Programación II Curso 4
TDA Pila

Implementada como vector dinámico

25 de marzo 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_VECTOR_DINAMICO_H__
#define __PILA_VECTOR_DINAMICO_H__

#include <stdbool.h>
#include <stdlib.h>

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

/*
 * Crea una pila, reservando la memoria necesearia para almacenar la estructura.
 * Devuelve una refencia a la estructura pila_t creada
 */
pila_t* crear_pila();

/*
 * Apila un elemento. En caso de ser necesario expande la estructura.
 * Devuelve 0 si pudo encolar o 1 si no pudo.
 */
int apilar(pila_t* pila, void* elemento);

/*
 * Desapila un elemento. En caso de ser necesario contrae la estructura.
 * Devuelve 0 si pudo desapilar o 1 si no pudo.
 */
int desapilar(pila_t* pila);

/*
 * Determina si la pila está vacia.
 * Devuelve true si está vacía o false si no.
 */
bool vacia(pila_t* pila);

/*
 * Devuelve el elemento en el tope de la pila o NULL en caso de estar vacía.
 */
void* tope(pila_t* pila);

/*
 * Destruye la pila liberando la memoria reservada por el vector dinamico.
 * Devuelve 0 si pudo destruirla o 1 si no pudo.
 */
int destruir_pila(pila_t* pila);

#endif /* __PILA_VECTOR_DINAMICO_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 que 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_vector_dinamico.h"
#include <stdio.h>

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

        char elemento_1 = '2';
        char elemento_2 = 'o';
        char elemento_3 = 'g';
        char elemento_4 = 'l';
        char elemento_5 = 'A';

        for (int i = 0; i < 3; i++) {
                apilar(pila, &elemento_1);
                apilar(pila, &elemento_2);
                apilar(pila, &elemento_3);
                apilar(pila, &elemento_4);
                apilar(pila, &elemento_5);
                printf("Tamanio pila: %i\n", pila->tamanio);
        }

        for (int i = 0; i < 5; i++) {
                printf("%c\n", *(char*)tope(pila));
                desapilar(pila);
        }
        printf("Tamanio pila: %i\n", pila->tamanio);

        for (int i = 0; i < 5; i++) {
                desapilar(pila);
        }
        printf("Tamanio pila: %i\n", pila->tamanio);

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

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

```
==17682== Memcheck, a memory error detector
==17682== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==17682== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==17682== Command: ./pila_vd
==17682==
Tamanio pila: 10
Tamanio pila: 10
Tamanio pila: 20
A
l
g
o
2
Tamanio pila: 20
```

```
Tamanio pila: 10
==17682==
==17682== HEAP SUMMARY:
==17682==     in use at exit: 0 bytes in 0 blocks
==17682==   total heap usage: 5 allocs, 5 frees, 1,368 bytes allocated
==17682==
==17682== All heap blocks were freed -- no leaks are possible
==17682==
==17682== For counts of detected and suppressed errors, rerun with: -v
==17682== 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.