Prácticas de Autómatas y Lenguajes. Curso 2017/18

Práctica 1: pila

Descripción del enunciado

Esta práctica tiene dos objetivos principales:

1. Que desarrolles con el lenguaje de programación C una librería para manejar una pila tan genérica como sea posible capaz de guardar como dato cadenas de caracteres.
2. Que esa librería esté preparada para integrarse en el futuro en un simulador de autómatas a pila.

Descripción de la pila.

Se te pide que implementes el tipo abstracto de datos pila (stack) que ya conoces de otros cursos.
A continuación se describen las decisiones de diseño más relevante que tiene que cumplir tu implementación.

- Es importante que sólo reserves memoria exactamente para las posiciones de la pila que están ocupadas, por lo tanto evita un esquema en el que reserves una cantidad de memoria inicial que vayas ocupando y utiliza memoria dinámica.
- Ten en cuenta que esta decisión determina la semántica de alguna de las funciones típicas de la pila. Aunque se detallará más adelante anota que esta decisión afecta al menos (ya que depende de tu implentación) a las siguientes funciones
    - Push, ya que tendrás que redimensionar la pila (para aumentar una posición)
    - Pop, ya que tendrás que redimensionar la pila (para reducir una poisición)
    - IsFull, ya que nunca lo será.
- Se te pide que tu pila sea genérica, es decir, capaz de albergar cualquier tipo de dato. Por eso en la implementación que te pedimos consideraremos
    - Que el tipo de dato elemental que almacena la pila es un puntero genérico
    - Que las funciones de manipulación específicas del dato almacenado en la pila se guarden en la propia pila de forma explícita mediante punteros a ellas. En principio es suficiente con tener
        - Una función para liberar el dato de tipo

```
typedef void (*destroy_element_function_type)(void*);
```

        - Una función para copiar el dato

```
typedef void (*(*copy_element_function_type)(const void*));
```

        - Y una función para imprimir el dato

```
typedef int (*print_element_function_type)(FILE *, const void*);
```

- A continuación se te muestra una posible definición (por favor, es importante que mantengas tanto los nombres de funciones como los tipos visibles desde fuera del módulo stack)

En stack.h

```
typedef struct _Stack Stack;

/* Tipos de los punteros a función soportados por la pila */

typedef void (*destroy_element_function_type)(void*);
typedef void (*(*copy_element_function_type)(const void*));
```

```
typedef int (*print_element_function_type)(FILE *, const void*);
```

En stack.c

```
#include "stack.h"


struct _Stack {

    /* COMPONENTES QUE UTILICES PARA IMPLEMENTAR LA PILA COMO POR EJEMPLO top
PARA APUNTAR A LA CIMA O item PARA GUARDAR LOS DATOS EN LA PILA */
    /* CUIDADO... EL TIPO DE DATO GUARDADO EN LA PILA (POR EJEMPLO SI item ES UNA
COLECCION) DEBE SER void * (LA COLECCIÓN LO SERÁ DE PUNTEROS TIPO void *  */


    /* ESTOS SON LOS PUNTEROS A LAS FUNCIONES NECESARIAS DE MANIPULACIÓN DEL DATO
*/
    destroy_element_function_type   destroy_element_function;
    copy_element_function_type      copy_element_function;
    print_element_function_type     print_element_function;
};
```

Y a continuación las cabeceras que debes respetar, con una breve descripción de su funcionalidad. Cualquier aspecto que quede abierto en la descripción de las mismas, puede ser cerrado por ti a tu conveniencia.

```
/**-------------------------------------------------------------------
Inicializa la pila reservando memoria y almacenando los tres punteros a función
pasados como parámetro (el primero para destruir elementos, el segundo para
copiar elementos y el tercero para imprimir elementos).
Salida: NULL si ha habido error o la pila si ha ido bien
-------------------------------------------------------------------*/
Stack * stack_ini(destroy_element_function_type f1, copy_element_function_type
f2, print_element_function_type f3);
/**-------------------------------------------------------------------
Elimina la pila
Entrada: la pila que se va a eliminar
-------------------------------------------------------------------*/
void stack_destroy(Stack *);
/**-------------------------------------------------------------------
Inserta un elemento en la pila.
Entrada: un elemento y la pila donde insertarlo.
Salida: NULL si no logra insertarlo o la pila resultante si lo logra
Debe hacer una copia en memoria nueva del dato que debe insertar y redimensionar
la colección de elementos para poder almacenar uno más
-------------------------------------------------------------------*/
Stack * stack_push(Stack *, const void *);
/**-------------------------------------------------------------------
Extrae un elemento en la pila.
Entrada: la pila de donde extraerlo.
Salida: NULL si no logra extraerlo o el elemento extraído si lo logra.
Nótese que la pila quedará modificada. De hecho debe ser redimensionada para
eliminar la posición que ya no se utiliza
-------------------------------------------------------------------*/
void * stack_pop(Stack *);
```

```
/**-------------------------------------------------------------------
Copia un elemento (reservando memoria) sin modificar el top de la pila.
Entrada: la pila de donde copiarlo.
Salida: NULL si no logra copiarlo o el elemento si lo logra
-------------------------------------------------------------------*/
void * stack_top(const Stack *);
/**-------------------------------------------------------------------
Comprueba si la pila esta vacia.
Entrada: pila. Salida:
TRUE si está vacía o FALSE si no lo esta
-------------------------------------------------------------------*/
Bool stack_isEmpty(const Stack *);
/**-------------------------------------------------------------------
Comprueba si la pila esta llena.
Entrada: pila. Salida: TRUE si está llena o FALSE si no lo esta
Recuerda, esta función es típica en cualquier implementación de stack, pero en la
que se te pide que realices debería devolver siempre FALSE
-------------------------------------------------------------------*/
Bool stack_isFull(const Stack *);
/**-------------------------------------------------------------------
Imprime toda la pila, colocando el elemento en la cima al principio de la
impresión (y un elemento por línea).
Entrada: pila y fichero donde imprimirla.
Salida: Devuelve el número de caracteres escritos (este valor de retorno es
irrelevante)
-------------------------------------------------------------------*/
int stack_print(FILE*, const Stack*);

/**-------------------------------------------------------------------
Tamaño de la pila
Entrada: pila.
Salida: Devuelve el número de elementos de la pila (0 si está vacío, -1 si hay
algún error)
-------------------------------------------------------------------*/
int stack_size(const Stack* );
```

Ejemplos

A continuación se muestra un ejemplo de programa principal

Observa que:

- Se ha supuesto que hay un tipo auxiliar (generic_string) que contiene las funciones de manipulación del string, dato que debe guardar la pila.

```
#include "stack.h"
#include "generic_string.h"

int main(int argc, char** argv)
{
	Stack * p_s_string1, * p_s_string2;
	char * aux_string;

	p_s_string1 = (Stack *) stack_ini(	destroy_p_string, copy_p_string,
print_p_string );
	p_s_string2 = (Stack *) stack_ini(	destroy_p_string, copy_p_string,
print_p_string );
```

```
        stack_push(p_s_string1, "hola 1");
        stack_push(p_s_string1, "hola 2");
        stack_push(p_s_string1, "hola 3");
        stack_push(p_s_string1, "hola 4");
        stack_push(p_s_string1, "hola 5");
        stack_push(p_s_string1, "hola 6");
        stack_push(p_s_string1, "hola 7");


        while(!stack_isEmpty(p_s_string1))
        {
                aux_string = stack_pop(p_s_string1);
                stack_push(p_s_string2, aux_string);
                destroy_p_string(aux_string);

        }

        stack_print(stdout,p_s_string2);

        stack_destroy(p_s_string2);
        stack_destroy(p_s_string1);

        return 0;

}
```

La salida esperada al ejecutar este programa sería la siguiente:

```
S (string): hola 1
S (string): hola 2
S (string): hola 3
S (string): hola 4
S (string): hola 5
S (string): hola 6
S (string): hola 7
```

El formato específico utilizado para imprimir cada elemento del stack se deja a tu criterio.

En esta práctica se pide:

- Cada grupo debe realizar una entrega
- Tu profesor te indicará la tarea Moodle donde debes hacer la entrega.
- Es imprescindible que mantengas los nombres especificados en este enunciado **sólo para las funciones que se piden de manera obligatoria**.
- Cualquier práctica que no compile (con los flags de compilación -Wall -ansi -pedantic), o no ejecute correctamente será considerada como no entregada.

- Cualquier práctica que deje “lagunas de memoria” será penalizada de manera explícita en función de la gravedad de esas lagunas.