MEMORIA PRÁCTICA 2

EJERCICIO 1: Indicad qué modificaciones habría que hacer a la pila del ejercicio 4 para que la cima de la pila fuera de tipo puntero, en particular especificad qué ficheros habría que modificar y cuáles serían estos cambios:

En **stack_fp.h** no habría ningún cambio.

En **stack_fp.c** :

Cambio en la estructura de stack: *int top;* se cambia por *void **top;*

Cambio en stack_ini: *s->top = 0;* se sustitye por *s->top = NULL*

Cambio en stack_destroy(Stack *s): Antes había un bucle for para ir liberando los elementos de la pila uno a uno,

```
for (i=0; i< s->top; i++){
        s->destroy_element_function(s->ítem[i]);
}
```

Ahora ese bucle lo sustituimos por un while usando el puntero:

```
While (ps->top >= ps->datos){
        s->destroy_element_function(s->ítem[i]);
        ps->top --;
}
```

Y tras este bucle liberamos el puntero con *free(ps->top);*

En stack_push(Stack *s, const void *e) cambiamos :

```
s->item[s->top] = ec; (ec es una copia de e)
s->top ++;
```

por :

```
if(stack_isEmpty(s) ==TRUE) {
        s->top = &(s->ítem[0]);
        *(s->top) = ec;
}
s->top ++;
*(s->top) = ec;
```

En stack_pop(Stack *s): cambiamos

```
s->top --;
Return s->item[s->top];
```

Por:

```
Void *aux;
aux = s->copy_element_function(*(s->top));
*(s->top) = NULL;
s->top--;
Return aux;
```

En stack_top(Stack *s) es todo igual, except por un cambio:

```
ec = s->copy_element_function(s->item[s->top -1]);
```

se cambia por :

ec->s->copy_element_function((s->top));*

En stack_isEmpty(Stack *s) cambiaríamos :

If(s->top == 0)

Por:

If(s->top== NULL)

En stack_isFull(Stack *s) cambiamos :

If (s->top == MAX_STACK)

Por :

If(ps->top == &(s->datos[MAX_STACK -1]))

*En stack_print(FILE *f, Stack *s) habría que cambiar :*

for(i=(s->top -1); i>=0; i--){
temp = s->print_element_function(f, s->item[i]);

Por :

for(i=s->top; i!=&(s->item[0]); i--){
*Temp= s->print_element_function(f,*i);*

En el resto los archivos no sería necesario cambiar nada, pues precisamente es en lo que basa un TAD, el usuario se desentiende de la implementación en sí.

En el caso de haber implementado el ejercicio 3 así (pues en el enunciado de la memoria se decía que era el ejercicio 3 pero se nombraban los ficheros del ejercicio 4 y teníamos dudas),hay que tener en cuenta que tendríamos que haber implementado stack.c con s->top de la forma puntero a un elemento.
Las funciones principales su implementación sería similar a las de stack_fp.c.
Habría ligeros cambios ya que ahora tenemos una pila de nodos y no de punteros a void, esto simplifica el trabajo. Por ejemplo:

-En stack_print llamamos directamenete a element_print para imprimir los nodos.
-En stack_top llamamos a element_copy, en vez de escribir s->element_copy_function(x).
-En stack_push y stack_pop ocurre lo mismo que en la anterior.
-En stack_destroy para ir eliminando elemento a elemento llamamos a element_destroy(x), en vez de poner s->element_function_destroy(x).

Teniendo en cuenta que, excepto en el tres, en todos los ejercicios de la práctica tenemos una pila de elementos, los cambios a realizar serán los mismos que en p2_e3, es decir, cambios en stack.c .

EJERCICIO 2: Una aplicación habitual del TAD PILA es para evaluar expresiones posfijo. Describe en detalle qué TADs habría que definir/modificar para poder adaptar

la pila de enteros (P2_E1) en una que permita evaluar expresiones posfijo. Realiza el mismo ejercicio pero con la pila general (P2_E4).

En el ejercicio 1 dado que ya tenemos una pila de enteros no sería necesario crear ningún otro TAD, valdría con definir las tres funciones siguientes:

Bool esOperando(char c); que devuelve true si el carácter es un operando, un número

Bool esOperador(char c); que devuelve true si el carácter es un operador, un símbolo

Bool prioridad(char c1, char c2); que devuelve true si el primer carácter es de mayor prioridad

Con estas funciones, valdría con leer en un bucle los caracteres de la cadena en posfijo hasta llegar al ';' introduciendo los operandos en la pila y haciendo dos pops al encontrar un operador, para luego operarlos con la función operar e introducir el resultado en la pila. Hay que tener en cuenta que cuando realizamos los dos pops hay que transformar cada una de las dos cadenas a entero para poder operar, esto lo hacemos con la función *atoi* implementada en C.

Podríamos implementar sin embargo el TAD operador y/o operando que incluyesen las funciones mencionadas anteriormente, aunque como hemos visto no es necesario.

En el ejercicio 4 dado que la pila esta formada por punteros a void tampoco sería necesario un nuevo TAD, simplemente habría que, en la llamada a *stack_ini*, establecer las funciones de punteros a entero definidas en functions.c . Aun así, sería necesario añadir las 3 funciones anteriores a stack_fp.c

MEMORIA SOBRE LA PRÁCTICA:

EJERCICIO 2:

En el ejercicio 2 al final decidimos cambiar element.c, haciendo que los elementos fueran punteros a nodo. Durante la realización de este ejercicio, también pensamos hacer una pila de punteros a void para que pudiera valer tanto para enteros como para nodos, implementación que nos habría facilitado el ejercicio 4, sin embargo, decidimos no hacerlo porque no sabíamos usar punteros a funciones.

EJERCICIO 3:

Tuvimos distintas dudas a la hora de implementar el algoritmo que recorriese el grafo hasta que encontramos uno que funcionase correctamente. Nuestro algoritmo empieza en un primer nodo, mete en la pila todos los nodos a los que está conectado este y los pone en color gris, entonces va sacando sucesivamente nodos de la pila, marcándolos en color negro y metiendo en la pila los nodos a los que están conectados que no están ya marcados como negro, así visitamos todos los nodos posibles desde ese primero.

Para comprobar si un nodo es accesible o no basta con ver si lo metemos en la pila antes de que esta esté vacía.

EJERCICIO 4:

Durante la realización del diseño no hemos pensado varias alternativas, aunque si ha habido dificultades a la hora de implementar las nuevas funciones debido a la nueva estructura de la pila y a que no estábamos familiarizados con el uso de punteros a funciones.

CONCLUSIONES FINALES:

Esta práctica nos ha servido para terminar de entender el tema de pilas y también durante la realización de la práctica nos han ido surgiendo dudas que en clase no nos habíamos planteado.

El ejercicio 4 ha sido bastante útil ya que al no haberlo practicado en clase nos ha servido para saber cómo realizaríamos una pila genérica, que podemos reutilizar en distintos proyectos.

La parte de la memoria de la práctica también ha sido útil para repasar como sería la implementación si top fuera un puntero y no un entero, pues no lo hemos implementado así en ningún ejercicio y nos ha permitido ver si realmente lo sabemos hacer y lo entendemos.

Pareja 8

Javier Delgado del Cerro

Victoria Pelayo Alvaredo

Ignacio Rabuñal García