PRÁCTICA 1: Utilización del concepto de Tipo Abstracto de Dato

OBJETIVOS

El objetivo de esta práctica es utilizar los conceptos aprendidos en las clases de teoría sobre los Tipos Abstractos de Datos (TAD) y reflexionar sobre su utilidad y conveniencia.

Además de la documentación que los entornos de desarrollo proporcionan sobre el lenguaje de programación C, mediante la página de UAMX de la asignatura los profesores de los laboratorios proporcionarán las indicaciones adicionales para desarrollar el trabajo de cada una de las sesiones.

NORMATIVA DE ENTREGA

Como resultado de esta práctica debe entregarse un fichero empaquetado (.zip) que contenga todos los ficheros que permitan abrir, construir el ejecutable y ejecutar su proyecto **antes de la primera sesión de la práctica siguiente. Cada grupo puede entregar hasta las 12 de la noche del día anterior al que su profesor iniciará la siguiente práctica.** Este fichero empaquetado debe, por tanto, contener al menos los siguientes ficheros:

- README.txt: fichero de texto plano donde han de constar vuestros datos (nombres, e-mail, grupo de prácticas, número de grupo) y cualquier otra indicación adicional sobre la práctica. Sobre la descripción de vuestro trabajo es suficiente con que expliquéis las cuestiones que os hayan podido parecer relevantes de vuestra solución. No se trata de que repitáis las instrucciones del enunciado ni las indicaciones que vuestros profesores hayan hecho en clase. Sólo debéis centraros en las peculiaridades que vuestra solución contenga. Añade en éste fichero de forma clara la lista de comandos necesarios para generar los ejecutables.
- Los ficheros que deben entregarse en esta práctica se listan a continuación. Se adjunta a este enunciado un fichero .zip que contiene un esqueleto de estos ficheros. En particular se han proporcionado ya implementados las cabeceras y los programas de prueba de los TAD premiado y premios.
    - premiado.h : Cabecera del TAD premiado
    - premiado.c : Implementación del TAD premiado
    - premiado_test.c: Programa de prueba para el TAD premiado.
    - premiados.h: Cabecera del TAD premiados.
    - premiados.c: Implementation del TAD premiados.
    - premiados_test.c: Programa de prueba para el TAD premiados.
    - premio.h: Cabecera del TAD premio
    - premio.c: Implementación del TAD premio
    - premio_test.c: Programa de prueba para el TAD premio.
    - premios.h: Cabecera del TAD premios.

- premios.c: Implementación del TAD premios.
- premios_test.c: Programa de prueba para el TAD premios.
- const.h: Fichero de cabecera con constantes globales de la aplicación.
- main.c: Programa principal que incluye toda la funcionalidad de la práctica.

Tenga en cuenta que la práctica está dividida en 3 sesiones y que deberá entregar para cada sesión el zip correspondiente.

El nombre del fichero comprimido tiene que seguir la siguiente estructura:

lab<turno>_gr<numero_grupo>_p1_s<numero_sesion>.zip

Donde:

- <turno> es el turno asignado (4111, 4112, 4113 o 4114)
- <numero_grupo> es el que identifica a tu grupo dentro de tu laboratorio
- <numero_sesion> es el de la sesión correspondiente

Es obligatorio respetar todos aquellos nombres y formatos que se describen en el enunciado.

¡No olvidéis incluir vuestros nombres en los ficheros entregados!

CRITERIOS BÁSICOS DE EVALUACIÓN

Las siguientes normas deben ser consideradas en la realización de los problemas propuestos y la entrega del material:

1. *Estilo de programación*: debe ajustarse a las normas básicas de estilo de la programación estructurada en C y que ya se han presentado en asignaturas anteriores de programación.
2. *Documentación del código*: todas las funciones que se programen deberán estar convenientemente y brevemente comentadas, tanto en su cabecera (funcionalidad y parámetros de entrada/salida), como en su cuerpo (descripción de etapas relevantes). Se evitarán comentarios obvios o redundantes.
3. *Pruebas*: es especialmente importante el control de errores. Es justificable que un programa no admita valores muy grandes de los datos, pero no es justificable que el programa tenga un comportamiento anómalo con dichos valores.
4. *Comprobación y propagación de errores*: se deberán comprobar todos los errores susceptibles de producirse en el programa, incluyendo ausencia de archivo de datos de entrada, errores de formato en datos de entrada, fallo de funciones del sistema operativo (malloc, etc.). Se comprobará que los parámetros pasados a las funciones sean correctos (p.ej., no nulos). Se considerará fallo grave que el programa aborte su ejecución por un tratamiento de errores insuficiente o inadecuado.
5. *Liberación de memoria dinámica*: al finalizar los programas, éstos deberán haber liberado toda la memoria dinámica que hayan reservado durante su ejecución.
6. *Uso de Valgrind*. Es imprescindible que todos los ejercicios hayan sido probados usando Valgrind para asegurarse de que no existen problemas de memoria.

INTRODUCCIÓN

En esta práctica se propone crear un sistema base de datos para almacenar y recuperar información acerca de los premiados y de los premios otorgados en los Nobel durante todo su historia. El sistema tendrá las siguientes características, que se desarrollarán a lo largo de las tres partes que componen las prácticas de la asignatura:

- Lectura de datos de premios y premiados almacenados en ficheros de texto.
- Inserción de datos manualmente y su almacenamiento en memoria.
- La base de datos mantendrá la consistencia de los datos, comprobando circunstancias como que no haya premiados y premios repetidos, etc.
- El sistema ofrecerá una serie de consultas como, "búsqueda de premiados por premio", "búsqueda de los premiados por nombre", etc. y medirá el tiempo de ejecución.
- Se utilizará un índice para acelerar algunas consultas.

Para ofrecer una primera impresión de la aplicación que se pretende desarrollar, a continuación se muestra un extracto del menú de la aplicación.

```
1. Insertar datos manualmente
       1.1 Insertar premiado
       1.2 Insertar premio
2. Consultas
       2.1 Buscar premiados por nombre
       2.2 Buscar premios por nombre
       2.3 Buscar premiados de determinada premio
3. Crear índices
4. Salir

> Elige una opción:
```

1ª SEMANA: TAD premiado y premio

TAD premiado:

Se implementará el TAD premiado, que contendrá la información de cada premiado antes descrita. La estructura de datos y las operaciones se almacenarán en los ficheros premiado.h y premiado.c. También se implementará un TAD premiados para gestionar listas de premiados.

Las funciones que es necesario implementar para el TAD premiado son las siguientes:

- Crear el premiado, para lo cual todos los campos de la estructura del premiado deben inicializarse a valores por defecto (por ejemplo, a NULL los punteros y 0 los de tipo numérico):

```
premiado * new_premiado();
```

- Eliminar el premiado y liberar sus recursos:

```
void destroy_premiado(premiado * a);
```

- Funciones de modificación de los datos de los premiados:
  - Se deberá copiar la cadena nombre y país pasada como argumento. No se deberá asignar el puntero.

```
void set_premiado_id(premiado * a, int id);
void set_premiado_nombre(premiado * a, const char * nombre);
void set_premiado_pais(premiado * a, const char * pais);
void set_premiado_genero(premiado * a, const char * genero);
```

- Funciones de acceso a los datos de los premiados:

```
int get_premiado_id(premiado * a);
const char * get_premiado_nombre(premiado * a);
```

- Imprimir la información de un premiado por la salida estándar (`stdout`) en formato libre siempre que se muestren todos los campos:

```
void print_premiado(premiado * a);
```

TAD premio

Se implementará el TAD premio, que contendrá la información de un premio. Un premio tendrá un **código** representado como un entero, un **año** y una **categoría**. Además el TAD debe mantener una estructura *premios* para almacenar los premiados que participan en la premio. La implementación de esta lista se dejará para la semana siguiente.

Las funciones que es necesario implementar el TAD premio son las siguientes:

- Crear una premio, para lo cual todos los campos de la estructura deben inicializarse a valores por defecto (por ejemplo, NULL los punteros y 0 los de tipo numérico):

```
premio * new_premio();
```

- Eliminar una premio y liberar sus recursos:

```
void destroy_premio(premio *f);
```

- Al igual que en el TAD *premiados*, se proporcionarán una serie de funciones para modificar los campos del TAD, en este caso:

```
void set_premio_id(premio * f, int id);
void set_premio_anio(premio * f, int anio);
void set_premio_categoria(premio * f, const char * categoria);
```

- También se implementarán funciones para acceder a los valores de tales campos:

```
int get_premio_id(premio * f);
int get_premio_anio(premio * f);
const char * get_premio_categoria(premio * f);
```

- Se crearán funciones para añadir y buscar premiados (la implementación de estas dos funciones se puede dejar para la siguiente semana):

```
void add_premio_premiado(premio * f, premiado *a);
BOOL has_premio_premiado(premio * f, premiado *a);
```

- Imprimir la información de una premio por la salida estándar (`stdout`) en formato libre siempre que se muestren todos los campos:

```
void print_premio(premio * f);
```

Programa de prueba

Para probar el código que ha creado se proporcionan dos programas de prueba premiado_test.c y premio_test.c. Cada uno de estos programas incluye un *main* que ejecuta una serie de pruebas para los TAD premiado y premiados respectivamente. Es importante entender el código proporcionado y seguir un esquema similar para probar el resto del código que se implemente más adelante. A continuación se muestra un extracto del código de prueba de premiado_test.c.

```
#include <assert.h>
#include <string.h>
#include <stdio.h>
#include "premiado.h"

int test_set_get_nombre
{
  char content[] = {"Marie Curie"};
  int ret;

  premiado * a = new_premiado();

  set_premiado_nombre(a, content);
  ret = strcmp(get_premiado_nombre(a), content);

  destroy_premiado(a);

  return ret==0 ? 1 : 0;
}

int main(int argc, char**argv)
{
  assert(test_set_get_nombre());
  return 0;
}
```

La función *test_set_get_nombre*() realiza una serie de invocaciones de funciones del TAD premiado comprobando su correcto funcionamiento, por ejemplo, el valor establecido con *set_premiado_nombre* debe ser igual al obtenido al invocar a *get_premiado_nombre*. El resultado del test se almacena en la variable ret y es comprobado con la función *assert*() en el *main*. Por último, es importante ejecutar el programa de prueba utilizando la herramienta valgrind, para comprobar que no hay fugas de memoria ni accesos inválidos a memoria.

2ª SEMANA: TAD lista de premiados y lista de premios

Asociado a estos TAD se definirán unos TAD *premiados* y *premios* para representar un array de premiados y premios respectivamente.

TAD premiados:

La estructura de datos y las operaciones se almacenarán en los ficheros premiados.h y premiados.c que constará de las siguientes funciones:

- Funciones para la reserva y liberación de memoria. La función new_premiados creará la estructura *premiados* e inicializará sus campos adecuadamente (ej., el número de premiados será 0). La función destroy_premiados liberará la memoria reservada para la estructura y el array pero no liberará la memoria reservada para los premiados en sí.

```
premiados * new_premiados();
void destroy_premiados(premiados * as);
```

- Una función para añadir un nuevo premiado indicado por argumento a la colección de premiados. Devuelve el número total de premiados del array o bien NULL_ID si hay un error. No añadirá al premiado si éste ha sido previamente añadido a la colección (esto es, evita duplicados). No se debe copiar el premiado dentro de la lista, solo almacenar el puntero.

```
int add_premiado(premiados * as, premiado * a);
```

- Función que devuelva el número total de premiados en el array:

```
int get_n_premiados(premiados * a);
```

- Función que devuelva el premiado de la posición i-esima:

```
premiado * get_premiado_i(premiados * a, int index);
```

- Función para buscar un premiado según su identificador:

```
premiado * get_premiado_by_id(premiados * a, int id);
```

TAD premios:

Esta última parte También se implementará un TAD premios para gestionar listas de premios. Este TAD es equivalente al TAD premiados.

Programa de prueba

Prueba los TAD *premiados* siguiendo un esquema similar al propuesto para los TAD *premiado* y *premio*. No ha fichero de prueba para premios al ser equivalente al TAD premiados.

3ª SEMANA: Insertar datos y menú de la aplicación

Se pide implementar un programa principal que, usando los TAD y funciones anteriormente descritos, muestre un menú del sistema para realizar algunas operaciones básicas, que se describen a continuación:

- Inserción de premiados.
  - El sistema permitirá al usuario escribir los datos de uno o más premiados y almacenarlos en una colección de premiados que estará en memoria hasta la finalización del programa.
- Inserción de premios.
  - El sistema permitirá al usuario escribir los datos de una o más premios y almacenarlas en una colección de premios que estará en memoria hasta la finalización del programa.
  - Puesto que los datos de un premio incluyen sus premiados, el sistema preguntará iterativamente el código de los premiados que han realizado cada premio por cada premio que se quiera insertar.
- Consultas.
  - Listar todos los premios almacenados en el sistema.
  - Listar todos los premiados almacenados en el sistema.
  - Listar todos los premios que incluyan un determinado premiado (se puede utilizar la función *strstr* para encontrar una subcadena dentro de la cadena).

A continuación se muestra un ejemplo de uso de la aplicación:

1. Ejecución del programa

```
$ ./practica1

1. Insertar datos
   1.1 Insertar premiado
   1.2 Insertar premio
2. Consultas
   2.1 Todos los premiados
   2.2 Todos los premios
   2.3. Premios que incluyan a un premiado
3. Salir
Elige una opción >
```

2. Inserción de datos de premios

```
Elige una opción > 1.1
Introduce los datos del premiado:
  Identificador (número): 23
  Nombre (100 caracteres): Mario Vargas Llosa
  País: Perú
  Género: Hombre
Elige una opción > 1.1
Introduce los datos del premiado:
  Identificador (número): 11
  Nombre (100 caracteres): Marie Curie
  País: Polonia
  Genero: Mujer
Elige una opción > 1.1
Introduce los datos del premiado:
  Identificador (número): 12
  Nombre (100 caracteres): Pierre Curie
  País: Francia
  Genero: Hombre
```

3. Inserción de datos de premios

```
Elige una opción > 1.2
Introduce los datos del premio:
  Identificador (número): 100
  Categoria: Literatura
```

```
  Año: 2010
Inserta premiados por código (-1 para terminar):
  Codigo del premiado #2: 22
  No existe un premiado con ese código
  Codigo del premiado #2: 23
  Codigo del premiado #3: -1

Elige una opción > 1.2
Introduce los datos del premio:
  Identificador (número): 101
  Categoria: Física
  Año: 1903
Inserta premiados por código (-1 para terminar):
  Codigo del premiado #2: 11
  Codigo del premiado #2: 12
  Codigo del premiado #3: -1
```

4. Consultas (ejemplo de salida de una consulta)

```
Elige una opción > 2.1
Total premiados: 3
      23 Mario Vargas Llosa
      11 Marie Curie
      12 Pierre Curie

Elige una opción > 2.2
Total premiados: 2
      2010: Literatura:
          23 Mario Vargas Llosa
      1903: Física:
          11 Marie Curie
          12 Pierre Curie
```