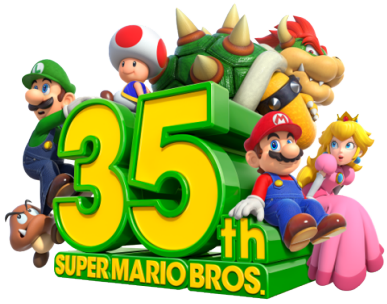
Universidad Francisco de Vitoria

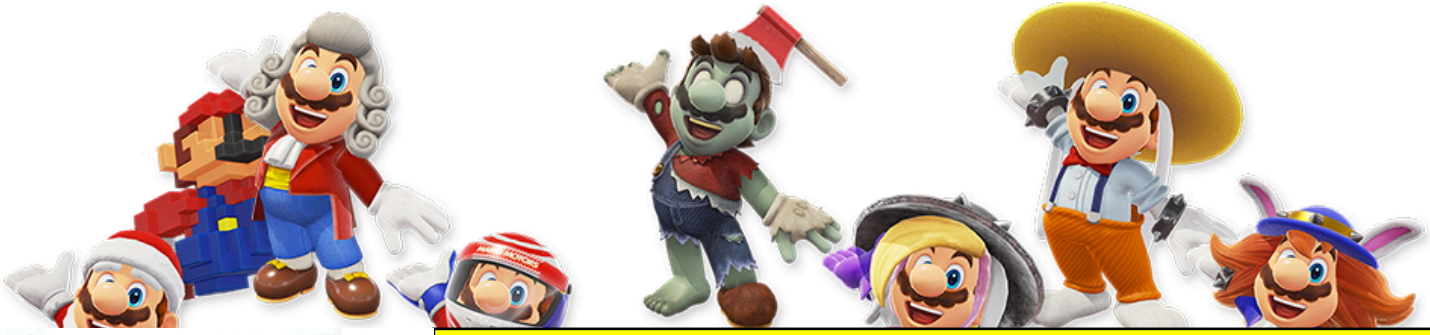
UFV Madrid



Universidad Francisco de Vitoria

Práctica
Disfraces de Mario

Introducción a la programación



PRÁCTICA 2: ARMARIO DISFRACES MARIO BROS

El objetivo de esta práctica es comprobar que el alumno ha comprendido y utiliza correctamente los arrays y las estructuras, así como las estructuras básicas del lenguaje (condicionales, bucles, declaración de variables, …).

Especificación del armario de disfraces de Mario Bros™

En esta práctica se trata de implementar el armario de disfraces de Mario Bros™ con las operaciones básicas CRUD (Create, Read, Update, Delete) para el manejo del mismo.

El elemento básico del armario es la siguiente estructura (el armario es una colección de disfraces):

```
#define MAX_CAD 100

struct disfraz{
    char nombre[MAX_CAD];
    short traje;
    short gorra;
    float precio;
};
```

La estructura disfraz tiene el nombre del disfraz, un campo que indica si Mario the comprado el traje del disfraz (almacena 0 si no tiene el traje del disfraz y 1 si ha comprado el traje), otro campo que indica si posee la gorra del disfraz (almacena 0 si no tiene la gorra del disfraz y 1 si ha comprado la gorra) y por último un campo para almacenar el precio del disfraz (precio del traje, de la gorra o del traje y la gorra en función de lo que haya comprado Mario en las tiendas Crazy Cap). .

Un disfraz se considera vacío (o nulo) si todos los campos de texto almacenan la cadena "?" y los campos numéricos almacenan un 0.

La operaciones disponibles en el armario son las siguientes (estas son las funciones y procedimientos que se deben implementar de forma obligatoria y son genéricas):

```
int insertarParte(struct disfraz armario[], int capacidadMaxima,
                  char nombreDisfraz[], float precioParte,
                  char tipo[]);
```

Esta función permite añadir en un *armario* que tiene una *capacidadMaxima* una parte de un disfraz, siempre y cuando haya un hueco libre en el armario y la parte del disfraz no se encuentre previamente almacenada. La parte del disfraz viene dada por su *nombreDisfraz* (el nombre se almacena en minúsculas, pero dos nombres de disfraz son iguales independientemente de mayúsculas y minúsculas), su *precioParte* y el *tipo* de parte del disfraz

Si el disfraz no estaba previamente en el armario se añade en la primera posición libre que se encuentre.

La función devuelve 1 si se ha podido insertar la parte del disfraz en el armario, devuelve 0 en caso que el armario esté lleno o el *tipo* de la parte sea incorrecto y devuelve -1 si la parte del disfraz ya se encontraba dentro del armario..

Si el almacenamiento de los disfraces se hace forma ordenada por nombre se obtendrá una puntuación superior en este apartado. LA ORDENACIÓN DEBE HACERSE EN LA FUNCIÓN insertarParte

```
void mostrarDisfraces(struct disfraz armario[], int
capacidadMaxima);
```

Esta función imprime en pantalla todos los disfraces (no vacíos) del armario acorde al siguiente formato

```
#orden nombre        precio Traje: [si/no]      Gorra: [si/no]
#orden nombre        precio Traje: [si/no]      Gorra: [si/no]
#orden nombre        precio Traje: [si/no]      Gorra: [si/no]
```

A continuación se muestra un ejemplo:

```
1       cientifico                    45.00     Traje: [si]     Gorra: [no]
2       esmoquin de villa chistera    20.00     Traje: [no]     Gorra: [si]
3       esqueleto                    380.00     Traje: [no]     Gorra: [si]
4       odyssey de mario               0.00     Traje: [si]     Gorra: [si]
```

En caso que el armario no tenga trajes almacenados, se debe mostrar en pantalla el mensaje “Armario vacio”.

```
struct disfraz buscarDisfraz(struct disfraz armario[], int
                                capacidadMaxima, char nombre[]);
```

Esta función busca en el *armario* un disfraz con el mismo nombre que el que se pasa en el parámetro *nombre* (NO DISTINGUE ENTRE MAYÚSCULAS Y MINÚSCULAS, el disfraz *Original* es igual que *original*). En caso de encontrarlo, devuelve la información del disfraz en un nuevo disfraz con la información encontrada en el armario.

Si no hay ningún disfraz con *nombre*, se devolverá un disfraz vacío (campos de texto con ‘?’ y campos numéricos con valor 0).

```
int borrarDisfraz(struct disfraz armario[], int capacidadMaxima,
                        char nombre[]);
```

Esta función borra del *armario* el disfraz cuyo *nombre* se pasa en el parámetro *nombre* (NO DISTINGUE ENTRE MAYÚSCULAS Y MINÚSCULAS, el

- Borrado compacto: elimina el disfraz y el hueco dejado en el armario por el borrado del mismo (el hueco libre está en el final del armario). Si se realiza este tipo de borrado se obtendrá una puntuación superior en este apartado.

Funcionamiento del armario

Asociado a las funciones anteriores (FUNCIONES GENÉRICAS Y QUE DEBEN FUNCIONAR PARA CUALQUIER TAMAÑO DE ARMARIO) hay que implementar el programa principal para la gestión del armario, acorde al siguiente menú de usuario.

```
1. Insertar parte
2. Buscar disfraz
3. Mostrar disfraces
4. Borrar disfraz
5. Salir
Introduce opcion:
```

Para definir el armario, vamos a suponer que tenemos un máximo de 50 trajes en el mismo (pero podría ser cualquier valor). Si se alcanza el numero máximo de trajes en el armario, no se podrán almacenar más trajes a partir de ese momento, mientras no se libere espacio, mediante el borrado de trajes. Cuando se inicia el armario, todos los disfraces deben tener en los campos de texto el símbolo '?' y en los campos numéricos un 0 (esto también debe ocurrir cuando se elimina un disfraz del armario)

Si se almacena el armario en un fichero de texto, antes de comenzar con el programa hay que cargar desde el fichero todos los disfraces almacenados. Si el fichero no existe, se debe crear uno nuevo con el nombre armario.txt y mostrar el menú de usuario. El fichero debe guardar cada ítem en una línea diferente, separando por ; cada uno de los campos (`nombre;traje;gorra;precio`)

El menú de la aplicación debe contemplar los diversos errores que pueda cometer el usuario y evitarlos (por ejemplo, introducir la opción 6).

El funcionamiento de cada una de las opciones es el que se describe a continuación

Insertar parte

El programa debe pedir los datos de la parte del disfraz que se quiere introducir por teclado, el usuario los teclea, si es posible almacenar el contacto se muestra el mensaje `"Parte insertada con exito"`, y si no es posible el programa debe mostrar el mensaje `"Error, la parte no es valida o armario esta llena"` o `"La parte ya estaba insertada"` según sea el caso.

```
nombre      precio      Traje: [si/no]   Gorra: [si/no]
```

Mostrar disfraces

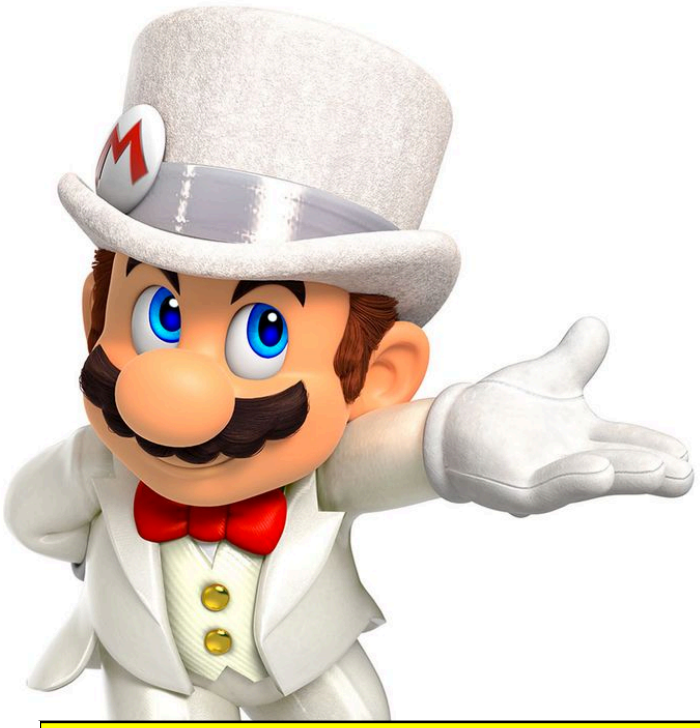
El programa debe mostrar por pantalla todos los disfraces almacenados en el armario acorde a lo especificado en la función ***mostrarDisfraces***.

Borrar disfraz

El programa debe pedir el nombre del disfraz a borrar, el usuario lo introduce por teclado. El programa elimina del armario el disfraz acorde a los especificado en la función ***borrarDisfraz*** e imprimir en pantalla el siguiente mensaje `"X disfraz(ces) borrado(s) con exito"` siendo X el numero de contactos eliminados en la función ***borrarPersona*** o imprime `"Error, no existen disfraces con ese nombre"` si no hay ningún disfraz que borrar.

Salir

El programa debe terminar. *Si se almacena les armario en un fichero de texto (armario.txt) se debe actualizar el fichero con todas las modificaciones realizadas en el mismo. Si el guardado del fichero se realiza de forma correcta se debe mostrar el mensaje* `"Armario guardado correctamente"` *y en caso contrario se mostrará el mensaje* `"ERROR: Armario no guardado"`



PRÁCTICA 2: ARMARIO DISFRACES MARIO BROS

Entrega

Esta práctica es de realización obligatoria para todos los alumnos. **Se realiza de forma obligatoria en grupos de 2 alumnos, excepto autorización por parte del profesor**.

Plazo de entrega: **23:59 horas del día 16 de diciembre de 2020**.

Habrá que entregar a través de Canvas el fichero **practica2.c** con el siguiente contenido:

- Un programa principal que contenga el menú para la gestión del armario.
- Cada una de las funciones descritas en el enunciado, acorde a los nombres dados en el mismo. Si una función no se implementa, debe devolver el valor -1 en el caso de funciones cuyo valor de retorno es un número entero. Si el valor de retorno es void, y la función no se implementa, se debe imprimir en pantalla `"Parte no realizada"`. Si el valor de retorno es un elemento de tipo struct debe devolver el disfraz vacío.
- Funciones adicionales que se utilicen para mejorar la estructura del código
- Si alguna de las partes no se implementa, el programa principal debe indicarlo en pantalla con el siguiente mensaje: `"Parte no realizada"`

La práctica SÓLO debe entregarla uno de los alumnos del grupo (el que aparezca en primer lugar en los nombres comentados)

Se considerará plagio entregar un código idéntico total o parcialmente al de otro compañero. Tampoco se admitirá un código idéntico a otro presente en cualquier página de Internet, incluso aun citando la fuente. El profesor dispone y usará herramientas de detección automática de plagio.

Puntuación

Esta práctica de entrega obligatoria se puntuará de 0 a 10, nota que se contabilizará en el 40% de la nota final de la asignatura en el caso de la evaluación continua (dentro de ese 40% esta práctica cuenta el 50%. Para poder considerar la nota de la práctica se debe obtener una nota igual o superior a 5). Las prácticas no entregadas se computarán como un 0.

Elementos a evaluar:

- **Incluir el nombre de los autores como las dos primeras líneas (comentadas) del fichero practica2.c**. Si para la correcta compilación del programa hay que incluir algún parámetro opcional al compilador, se debe incluir en una comentario debajo del nombre de alumno la correcta llamada al compilador.
- La no existencia de errores sintácticos en el código (es decir, **el código debe compilar**). En caso contrario, **una práctica cuyo código fuente dé errores de compilación se considerará suspensa directamente**.
- Correcto funcionamiento de los distintas funciones entregadas. Es decir, se valorará positivamente que no haya errores lógicos.
    - Sólo la función ***mostrarDisfraces*** debe hacer uso de la función ***printf***. En el resto de funciones está totalmente prohibido hacer uso de las funciones ***scanf*** y/o ***printf***
- Limpieza y claridad en el código.
- Presencia de comentarios en el código.
- Errores descritos en el documento "*Errores a evitar.pdf*" que se encuentra en Canvas junto con el enunciado de la práctica.

De forma numérica los pesos de cada parte de la practica son los siguientes.

insertarParte	1,5
mantener el armario ordenado por nombre de disfraz	1,5
mostrarDisfraces	1
buscarDisfraz	1,5
borrarDisfraz	1,5
mantener el armario sin huecos (armario compacto)	1,5
Programa principal	1,5
[EXTRA - OPCIONAL] almacenamiento en fichero de texto (lectura/escritura)	2