UNIVERSIDAD NEBRIJA

ESTRUCTURA DE DATOS Y ALGORITMOS - GRUPO C

PRÁCTICA Nº2

8 de marzo 2023

INSTRUCCIONES

1. No se permitirá la consulta de ningún otro tipo de material durante la realización de la práctica.

2. La entrega de la práctica sólo se admitirá a través de la **actividad** disponible en el campus virtual de la asignatura de prácticas de EDA del grupo C antes de la hora de finalización de la sesión de prácticas.

3. Aunque las prácticas se tengan que realizar en grupos de dos integrantes, para su evaluación, **ambos deberán hacer la entrega a través de su campus virtual, la misma práctica**. En otro caso, la práctica quedará **sin evaluar** y supondrá un **0 en la calificación del estudiante que no la haya entregado**.

4. Se debe entregar **los ficheros** correspondientes a cada ejercicio en formato **.cpp, sin comprimir**, con el nombre de ***P2EDAGC_Ex,*** donde la ***x*** debe sustituirse por el número del ejercicio correspondiente. Por ejemplo, P2EDAGC_E1 será el nombre del ejercicio número 1 de la práctica 2 del grupo C.

5. El fichero entregado debe **incluir, al principio del cada fichero entregado, el nombre de los integrantes del equipo**.

6. **El incumplimiento de alguna de las instrucciones sobre la realización/entrega de la Práctica supondrá su descalificación.**

IMPORTANTE:

- Todas las opciones deberán ser resueltos de forma **algorítmica**, es decir, la **solución** propuesta tendrá que ser **general** y **no particular** para unos determinados datos/valores.

- Todos los **ejercicios resueltos sin utilizar funciones** cuando sea apropiado se valorarán con una **nota máxima del 60% sobre la calificación prevista**.

- Los métodos/funciones de las clases que se indican en el enunciado de los ejercicios deben **mantener la definición de su interfaz**, es decir, **nombres de las funciones, número y tipo de argumentos/parámetros** indicados**.**

- Los ficheros entregados tienen que **compilar**.

- Se recomienda y será valorado de forma positiva que los nombres de las variables/objetos/clases/atributos/etc. sean **significativos** y **sustantivos**.

- Se recomienda y será valorado de forma positiva que los nombres de las funciones y los métodos sean **significativos** y **expresiones verbales**.

- Se pueden **diseñar e implementar otras funciones/métodos auxiliares o complementarias** a las solicitadas en el enunciado, siempre que tengan sentido.

- **Se pueden utilizar funciones propias de otras librerías de C++** (iostream, array, vectores, etc.), pero **NO para sustituir contenidos propios de la asignatura** (listas, pilas, colas, algoritmos de ordenación, etc.).

- Se recomienda una **primera lectura del enunciado de la práctica** para planificar su realización.

- Conviene ir **probando los diferentes recursos programados** de forma gradual. Programar todo seguido y realizar las pruebas al final cuando quedan 10 minutos para la entrega, suele acabar con **errores de compilación sin resolver en la entrega** y por lo tanto con una calificación de 0 puntos.

EJERCICIO

Escriba un programa en C++11, ***P2EDAGC_E1***, utilizando los **tipos de datos**, las **estructuras de control** y las **funciones** necesarias, que muestre por pantalla un **menú** con las siguientes opciones:

1. **Guardar libros en Pila y Cola.**
2. **Mostrar libros en Pila y Cola.**
3. **Mostrar libros entre dos posiciones de la Pila.**
4. **Calcular importe de libros vendidos.**
0. **Salir programa.**

El menú será una **función** (llamada ***menu***) que, **sin recibir ningún parámetro**, devolverá la **opción seleccionada por el usuario**. La función solicitará la opción al usuario que tendrá que validarse. Así, si la **opción indicada por el usuario no es válida**, se mostrará el mensaje "**La opción no es válida. Por favor, vuelva a seleccionar otra opción**" y **presentará de nuevo el menú anterior** para que el usuario pueda **repetir el proceso** seleccionando otra opción hasta que sea válida **(0,25 puntos)**.

Si el usuario selecciona la **opción 0**, el programa terminará mostrando el mensaje: "**Gracias y hasta pronto**". Siempre que se seleccione cualquier otra opción el programa, desde la función ***main***, deberá realizar y ejecutar dicha operación indicada. Después, volverá a **presentar de nuevo el menú anterior**, llamando a la función ***menu***, para que el usuario pueda **repetir el proceso** seleccionando otra opción **(0,25 puntos)**.

La descripción de las opciones del menú es la siguiente:

1. Guardar libros en Pila y Cola. (1,5 puntos)

Esta opción se tendrá que realizar a través de, al menos, una **función general.** Su objetivo es **guardar datos en las dos estructuras que se utilizarán en las demás opciones de la práctica**. Para ello, se solicitará al usuario los datos correspondientes a un libro (título y precio), según la descripción de **la clase Libro** que se incluye en el enunciado, para guardarlos tanto en **una Pila** como en **una Cola** que también son descritas posteriormente. Después, se preguntará al usuario si desea añadir nuevos libros a las estructuras. Si la respuesta es

afirmativa, se repetirá el proceso. En caso contrario, la entrada de datos terminará.

2. **Mostrar libros en Pila y Cola. (1,5 puntos)**

   Esta opción se tendrá que realizar a través de, al menos, una **función general.** Su objetivo es mostrar por pantalla los datos (título y precio) de los libros guardados en las estructuras Pila y Cola. Primero, se **mostrará un mensaje indicando que se van a mostrar los datos de la Pila** y, a continuación, se **mostrará el contenido de la Pila**. Después, se **mostrará un mensaje indicando que se van a mostrar los datos de la Cola** y, a continuación, se **mostrará el contenido de la Cola.** La **Pila** y la **Cola de libros deben mantener su contenido original después de mostrarlo por pantalla.**

3. **Mostrar libros entre dos posiciones de la Pila. (2 puntos)**

   Esta opción se tendrá que realizar a través de, al menos, una **función general y utilizando únicamente las operaciones básicas definidas en el TAD Pila.** Su objetivo es **determinar los libros que se encuentran entre dos posiciones concretas de la Pila, ambas incluidas**. Para ello, **se solicitarán al usuario dos valores que corresponden a las posiciones de la Pila**. Si los valores no son correctos para realizar la búsqueda (por ejemplo, no son válidos según el número de libros contenidos en la Pila, son valores negativos, etc.), se mostrará un **mensaje para informar al usuario**. Por el contrario, sí son correctos, se **localizarán los libros situados entre ambas posiciones (ambas incluidas) para mostrar por pantalla los títulos de los mismos**. La **Pila de libros debe mantener su contenido original después de realizar la operación de esta opción.**

   NOTA: Se considera que el libro ubicado en la parte superior de la Pila (CIMA) está en la posición 1 y el siguiente en la posición 2, y así sucesivamente hasta que se llega al libro que ocupa la posición más baja de la Pila (**enésima posición** si hay **n libros** en la Pila). Por tanto, si se solicita mostrar los libros entre las posiciones 5 y 8, se mostrarán los títulos correspondientes a los libros situados en **las posiciones 5, 6, 7 y 8 de la Pila.**

4. Calcular importe de libros vendidos. (1,5 puntos)

Esta opción se tendrá que realizar a través de, al menos, una **función general y utilizando únicamente las operaciones básicas definidas en el TAD Cola.** Su objetivo es **calcular el importe total de los libros que se van a facturar en un momento determinado** (libros situados en la Cola). Para ello, se irán sumando los importes correspondientes a cada libro que se encuentra en la Cola. Finalmente, se mostrará **un mensaje por pantalla indicando el importe total de libros vendidos**. La **Cola de libros debe mantener su contenido original después de realizar la operación de esta opción.**

Las **clases que se pueden utilizar** y sus **correspondientes interfaces** (no se pueden modificar) son las siguientes:

```
class CLibro {
    private:
        string titulo;
        int pvp;
    public:
        CLibro():titulo(""), pvp(0){}
        CLibro(string const &t, int p):titulo(t),pvp(p){}

        const string &getTitulo() const;
        void setTitulo(const string &newTitulo);
        int getPvp() const;
        void setPvp(int newPvp);
};
```

```
class Nodo {
    private:
        CLibro dato;
        shared_ptr<Nodo> next;
    public:
        Nodo():next(nullptr){}
        Nodo(CLibro const &d, shared_ptr<Nodo> const &ptr):dato(d),next(ptr){}

        const CLibro &getDato() const;
        void setDato(const CLibro &newDato);
        shared_ptr<Nodo> getNext() const;
        void setNext(const shared_ptr<Nodo> &newNext);
};
```

```
class Stack {
    private:
        shared_ptr<Nodo> front;
    public:
        Stack():front(nullptr){}

        bool empty() const;
        void push(const CLibro &dato);
        void pop();
        const CLibro &top() const;

        void mostrarPila() const;
        void reconstruirPila(Stack &p);
};
```

```
class Cola {
    private:
        shared_ptr<Nodo> first, end;
    public:
        Cola():first(nullptr), end(nullptr){}

        bool empty() const;
        void push(const CLibro &dato);
        void pop();
        const CLibro &front() const;

        void mostrarCola() const;
};
```

EJERCICIO 2

Analice la **complejidad** (coste temporal) de las siguientes **funciones recursivas**: **(1,5 puntos)**

NOTA: La solución del ejercicio deberá de incluirse en un fichero llamado, ***P2EDAGC_E2***.

a.-

```
int funcion1 (int n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return (n*funcion1(n-1));
    }
}
```

b.-

```
int funcion2 (int n) {
    int x;

    if (n <= 1) {
        return 1;
    }
    else {
        x = funcion2 (n / 2) + funcion2 (n / 2);
        for (int i{1}; i <= n; i++) {
            x = x + 1;
        }
        return x;
    }
}
```

EJERCICIO 3

Escriba una función en C++11, ***P2EDAGC_E3***, que permita **determinar el menor elemento de un vector de n elementos** de t**ipo int** y analice su **complejidad** (coste temporal). **(1,5 puntos)**