Práctica 4: Programación Modular

Fundamentos de Informática I

Grado en Ingeniería Informática y Tecnologías Virtuales

Curso académico 2021/2022

Alfonso Carlos Martínez Estudillo

Objetivos

Con la realización de esta práctica se pretende que el alumnado:

- Se familiarice con la programación modular de estructuras condicionales, vectores, matrices y estructuras.
- Repase conceptos vistos en prácticas y temas anteriores.
- Siga las recomendaciones del profesor acerca de la tabulación del código, espaciado, documentación y nombre de las variables.

Muy importante

- Todos los programas deben ser resueltos haciendo uso de funciones.
- Se utilizará la cuarta forma para la organización de los ficheros, es decir, fichero de cabecera.h, fichero de definición de funciones cabecera.cpp y fichero principal main.cpp.
- Todos los ejercicios tienen que tener su makefile que creará el ejecutable a partir de códigos objeto y bibliotecas.

Estructuras de Control

1. Ejercicio 1

Implemente una función que determine si un año es o no bisiesto. Recuerde que son bisiestos todos los años cuya cifra es múltiplo de 4 excepto los que siendo múltiplos de

100 no lo son de 400. Así 1900 no fue bisiesto, pero 2000 sí. Utilice dicha función para mostrar todos los años bisiestos comprendidos entre dos años introducidos por teclado.

2. Ejercicio 2

Codifique una función en C++ que calcule el factorial de un número dado por teclado. Posteriormente, implementar una función que calcule el número combinatorio haciendo uso de la función factorial definida previamente.

3. Ejercicio 3

Escriba una función que calcule la potencia de una base b elevado a un exponente e, no está permitido el uso de la función **pow**. Posteriormente, utilice la función para mostrar las n primeras potencias de una base y un exponente, todos ellos introducidos por teclado.

4. Ejercicio 4

Implemente una función que determine si un número es primo o no. Posteriormente, muestre por pantalla los números primos comprendidos entre dos números introducidos por teclado.

5. Ejercicio 5

Implemente una función que determine si un número es perfecto o no. Se dice que un número es perfecto si es igual a la suma de sus divisores menores que ese número. Por ejemplo, 6 es perfecto ya que 6 = 1 + 2 + 3, 28 es perfecto ya que 28 = 1 + 2 + 4 + 7 + 14. Posteriormente, muestre por pantalla los números perfectos comprendidos entre dos números introducidos por teclado.

Vectores

El tamaño máximo de los vectores será de 100 elementos.

6. Ejercicio 6

Implemente un programa en C++ que realice las siguientes operaciones en el siguiente orden:

1. Introducir el número de elementos del vector.
2. Introducir los elementos del vector.
3. Mostrar los elementos del vector.
4. Calcular la suma de los elementos del vector.
5. Calcular la media de los elementos del vector.
6. Calcular la varianza de los elementos del vector.
7. Determinar el valor máximo de los elementos del vector.
8. Determinar el valor mínimo de los elementos del vector.

7. Ejercicio 7

Codifique un programa que permita invertir los elementos de un vector. Se podrá hacer usando un vector auxiliar o no, pero en cualquier caso, al final el vector original tiene que quedar invertido.

8. Ejercicio 8

Un polinomio de grado N se puede representar en programación como un conjunto de N elementos reales, donde cada elemento sería el coeficiente asociado a un término del

polinomio. Implemente un programa que realice de manera secuencial las siguientes operaciones:

1. Leer un polinomio de grado N introducido por el usuario.
2. Mostrar el polinomio por pantalla de manera adecuada.
3. Evaluar el polinomio en un punto X introducido por el usuario.

9. Ejercicio 9

Realice un programa que maneje un vector de enteros de tamaño N a través de un menú con cinco opciones:

- Añadir un elemento al vector (comprobando que el vector no esté lleno).
- Eliminar un elemento del vector (comprobando que no esté vacío). Realiza los desplazamientos de elementos necesarios para que no queden huecos en el vector.
- Mostrar el contenido del vector.
- Contar el número de divisores de 5.
- Terminar.

Matrices

El tamaño máximo de las filas y columnas será de 100 elementos.

10. Ejercicio 10

Escriba un programa que lea las dimensiones de una matriz de m filas y n columnas e intercambie dos filas, a y b cualesquiera. El programa debe mostrar la matriz antes y después del intercambio.

11. Ejercicio 11

Dada una matriz de reales de m filas y n columnas, implemente un programa que calcule la media de los elementos de cada columna y las almacene en un vector de n elementos.

12. Ejercicio 12

Escribe un programa que rote k posiciones hacia la derecha los elementos de una fila f de una matriz.

13. Ejercicio 13

Implemente un programa que busque un elemento indicado por el usuario en una matriz de m filas y n columnas. El programa devolverá la fila y la columna donde se encuentre el elemento a buscar o -1, -1 si no se ha encontrado dicho elemento.

Estructuras

14. Ejercicio 14

Escriba un programa que utilizando la siguiente estructura para representar matrices, lea una matriz, la muestre por pantalla y calcule la suma de los elementos que hay por debajo de su diagonal principal. Si la matriz no es cuadrada no se realizará ninguna operación.

```
struct matriz{
  int nFil;
  int nCol;
  int m[100][100];
}
```

15. Ejercicio 15

Implemente un programa haciendo uso de estructuras, que lea un vector de números complejos y calcule el complejo media de todos ellos. El programa realizará secuencialmente las siguientes operaciones:

1. Leer los números complejos en un vector.
2. Mostrar los números complejos introducidos por teclado.
3. Sumar los números complejos del vector.
4. Calcular la media a partir de la suma anterior.