H.L.C
El Lenguaje C++

Rafael Ruiz
gandano@gmail.com

Actividad 1.1

1. Como sabes, C++ soporta el paso de parámetros por referencia de forma nativa. Implementa las siguientes funciones, pasando por referencia aquellos parámetros que creas necesarios. Evitar utilizar el mecanismo normal de retorno de las funciones, o complementarlo con el paso de parámetros por referencia.

   - Suma de dos números reales
   - Área y perímetro de un rectángulo
   - Factoral de un número
   - Determinar si un número es par
   - Determinar si un número es primo
   - Calcular la suma, resta, multiplicación y división de dos números, indicando la función si ha ocurrido algún error.

2. Modificar las funciones anteriores utilizando parámetros por defecto

   - Suma de dos números reales (uno de ello es 0)
   - Área y perímetro de un rectángulo (base del rectángulo es 10)
   - Factorial de un número (5 por defecto)
   - Determinar si un número es par (2 por defecto)
   - Determinar si un número es primo (7 por defecto)
   - Calcular la suma, resta, multiplicación y división de dos números, indicando la función si ha ocurrido algún error. (Ambos factores a 1 por defecto)

3. Utilizar la sobrecarga de funciones para implementar las siguientes funciones en C++

   - Suma de dos números (reales, enteros, y char (bytes))
   - Área del rectángulo. Los valores pasados podrán ser enteros o reales.

4. Utilizando plantillas (templates) en C++ , calcular el menor de dos valores numéricos

5. Crear una clase llamada **Complejo** para representar y operar con este tipo de números, que están formados por dos valores reales llamados **parte real**, y **parte imaginaria**. La forma en la que se imprimirán será **parte real + parte imaginaria * i**, donde i es $\sqrt{-1}$. Los atributos de la clase serán de tipo **double**. La clase debe proporcionar un constructor que permita inicializar el objeto, y debe utilizar valores predeterminador en sus parámetros, además de los siguientes métodos:

   - Suma de complejos. Se suman las partes reales y las partes imaginarias.
   - Resta de complejos. La parte real del operando derecho se resta de la parte real del operando izquierdo, y la parte imaginaria del operando derecho se resta de la parte imaginaria del operando izquierdo.
   - Impresión de números complejos en la forma (a, b), donde a es la parte real, y b la parte imaginaria.

6. Crear una clase llamada **Racional** para operar con fracciones. Comprobar el funcionamiento correcto de todas ellas realizando combinaciones variadas. Utilizar tipos enteros para los atributos que representan el numerador y denominador. Crear los constructores adecuados para permitir la creación de objetos e inicializarlos, utilizando parámetros con valores predeterminados. La fracción resultante se debe almacenar en su forma reducida. Los métodos mínimos a implementar, son:

   - Suma de dos racionales.
   - Resta de racionales
   - Multiplicación de racionales
   - División de racionales
   - Impresión en formato a/b (a numerador y b denominador)
   - Impresión como un número real

7. Generar una clase **Rectángulo** con los atributos **longitud** y **ancho**, cada uno con un valor predeterminado de 1. Añadir métodos (funciones miembro) para calcular el perímetro y el área del rectángulo, además de las correspondientes **gets()/sets()** para ambos atributos, protegiendoles de posibles valores no permitidos. Estos deberánb ser valores reales comprendidos entre 0.0 y 20.0.

8. Mejorar la clase **Rectángulo** del ejercicio anterior, de forma que almacene las coordenadas de las cuatro esquinas o vértices del rectángulo. El constructor llamará a una función **set()** que tomará cuatro coordenadas y verificará que cada una de estas se encuentre comprendida en el primer cuadrante, y que ninguna coordenada **x**, **y** sea mayor que 20.0. La función **get()** verificará además que las coordenadas proporcionadas formen un rectángulo. Además añadir métodos para calcular la longitud, el ancho, el perímetro, y el área. La longitud es la mayor de las dos dimensiones. Incluir un método predicado (retorna bool) llamda **cuadrado** que determine si el rectángulo es un cuadrado.

9. Modificar el ejercicio anterior (Clase mejorada Rectángulo) para incluir un método (función miembro) denominado **dibujar()** que muestre el rectángulo (en consola usando como punto el carácter '*' como valor por defecto) de dimensiones 25 x 25 que representa el primer cuadrante en el que debe alojarse el rectángulo definido. Añadir otro método denominado **setCaracterRelleno()** para especificar el carácter con el que se completará el área. Contemplar otra función miembro llamada **setCaracterPerímetro()** para indicar el carácter a usar en el dibujo del perímetro, además de las opciones para moverlo y rotarlo dentro del primer cuadrante.

10. Crear una clase para trabajar con números enteros muy grandes, mediante un array de 40 digitos. Ofreces funciones miembro **entrada**, **salida**, **suma**, y **resta**. Para comparar objetos de este tipo, proporcionar métodos como **esIgual()**, **esDistinto()**, **esMayor()**, **esMenor()**, **esMayorIgual()**, **esMenorIgual()**. Todas estas funciones miembro serán predicados. Añadir otro predicado **esCero()**.

11. Crear la clase ***Fecha*** con tres atributos enteros para almacenar el día, mes y año. Varios constructores en el que se podrá especificar una fecha completa o incompleta. En caso de usar el constructor por defecto, se tomará la fecha actual leyéndola del sistema mediante las funciones adecuadas de la librería ***ctime***, además de incluir un constructor copia. Dotar a la clase de los métodos necesarios para comparar fechas, y determinar la igualdad, menor o mayor. Imprimir las fechas en distintos formatos tales como "DD/MM/AAAA", "21 de Abril de 2021", "MM/DD/AA", ...

12. Crear una clase ***Persona*** con los atributos *nombre*, *apellidos*, *fechaNacimiento*, *domicilio*, *teléfono* y *estudios*. Crear los constructores adecuados, destructor en caso de necesidad, los métodos *gets()* y *sets()* correspondientes con el control necesario sobre los atributos. Añadir un método denominado *edadActual()*, que calcule la edad en función del año de nacimiento y día actual. Contemplar el método *imprimir()*, para mostar los datos de una persona, además de *comparar()* que retornará un booleano indicando la igualdad entre dos instancias de la clase Persona.

13. Implementar las siguientes funciones para trabajar con el tipo string de C++

    - Una función **string subcadena(string cadena, int longitud, int posicion)** que retorne un fragmento de la cadena pasada como argumento con la longitud y desde la posición indicada.
    - Una función **string borraCaracter(string cadena, char caracter)** que elimine todas las apariciones de un carácter en la cadena especificada.
    - Una función **int buscarSubcadena(string cadena, string subcadena)** que busque la primera aparición de una subcadena dentro de la cadena espeficicada y retorne su índice de posición en caso de que exista, o -1 en caso contrario. Modificar la función (sobrecarga) para obtener el número de apariciones de la subcadena (Si vale 1, sería como la función original). Realizar otra función similar que retorne el numero de veces que aparece la subcadena en la cadena.

14. Implementar la función ***codifica()*** que realiza la modificación de la cadena recibida sumando un desplazamiento a cada carácter, pero teniendo en cuenta que el resultado debe ser una letra. La función debe trabajar con minúsculas y/o mayúsculas, y tener en cuenta que los caracteres que no sean letras no se deben modificar, permanecerán sin alterar.

15. Desarrollar una función ***esPalindromo()*** que retorna *true* si el parámetro de tipo *string* que se le pasa es palíndromo.

16. Diseña una función llamada ***crearPalindromo()*** que añada a un *string* el mismo *string* invertido, de forma que el resultado sea un palíndromo. Implementarlo con las siguientes dos funciones equilalentes :

    - *string crearPalindromo(string)*
    - *void crearPalindromo(string &)*

17. Elaborar una clase ***CuentaAhorro*** que utilice un miembro de clase para contener la Tasa de Interés Anual de cada uno de los clientes que tengan una cuenta de ahorros. Cada miembro de la clase debe contener el saldo. Proporcionar un método ***ultimoInteresMensual()*** que calcule el interés mensual multiplicando el saldo por la Tasa de Interés Anual dividida entre 12. Este interés debe sumarse al saldo. Proporcionar una función miembro (método) de clase ***modificarTasaInteres()*** para establecer el nuevo valor para la Tasa de Interés Anual. Escribir una aplicación para probar esta clase generando dos instancias del objeto ***CuentaAhorro***, denominadas ***ahorrador1*** y ***ahorrador2*** con saldos de 2000€ y 3500€ respectivamente. Establecer la Tasa de Interés Anual al 3 %, luego obtener un cálculo del interés mensual e imprimir los nuevos saldos de cada ahorrador. Modificar la Tasa de Interés Anual al 4 %, calcular el interés del siguiente mes e imprimir los nuevos saldos de cada ahorrador.

En todos los ejercicios se valorará el manejo de las características particulares de C++ en los casos en los que sean adecuadas