Guía de Ejercicios #3 - Funciones

```
Advertencia

La resolución conjunta o grupal de los ejercicios aquí presentes no está
permitida, excepto en la medida en que puedas pedir ayuda a tus compañeros
de clase y a otras personas, y siempre que esa ayuda no se reduzca a que
otro haga el trabajo por vos.

El código fuente entregado por un estudiante debe ser escrito en su
totalidad por dicha persona.
```

En cuanto a la entrega:

Qué se entrega?	**Qué no se entrega?**
- Archivos fuente/source (.c)	- Archivos objeto (.o)
- Archivos encabezado/header (.h)	- Archivos ejecutables (programa, app, a.out, etc.)
- Bibliotecas específicas (.a)	

- Se deben entregar los tres ejercicios en un zip (usar template como ayuda para el formato).

Ejercicio 1.1

Implemente una función que calcule x elevado a la y sin utilizar la función estándar *pow()*. La función debe admitir exponentes enteros positivos, negativos, o de valor 0 (cero).
Utilice el siguiente prototipo de función de ser posible:

- **float mypow(float x, int y);**

Nota: La función debe retornar el resultado de la operación. Para referencias consultar 'man pow'.

Ejercicio 1.2

Escribir una función que simule el tiro de un dado haciendo uso de 'rand' para generar la secuencia aleatoria.

Ejercicio 1.3

Realice una función que reciba como argumento un número entero y determine si el mismo es primo. La función debe retornar:

- 1 (uno) si es primo.
- 0 (cero) si no es primo.
- 2 (dos) si se trata de un número negativo.
- 3 (tres) si el número es 1 (uno) o 0 (cero).

Nota: Utilice el siguiente prototipo de función de ser posible:

- **int es_primo(unsigned int numero);**

Ayuda: En matemáticas, un número primo es un número natural mayor que 1 que tiene únicamente dos divisores positivos distintos: él mismo y el '1'. Por el contrario, los números compuestos son los números naturales que tienen algún divisor natural aparte de sí mismos y del '1', y, por lo tanto, pueden factorizarse. El número '1', por convenio, no se considera ni primo ni compuesto.

Ejercicio 1.4

Realice una función que reciba un número entero que represente un año y determine si dicho año resulta ser bisiesto o no. De ser bisiesto la función debe retornar un 1 (uno).

Nota: Un año es bisiesto si es divisible por 4 y, si es divisible por 100 y también por 400. Es decir, que si es divisible por 100 y no por 400, no seria bisiesto (Ej, 1700, 2100, 2600, etc).

Ejercicio 1.5

Realizar una función que calcule el factorial de un número entero pasado como argumento. Se deben verificar todas las condiciones correspondientes para que el resultado sea correcto, en caso no poder devolver el resultado correcto, retornar 0 (cero). Realizar todas las validaciones q considere necesarias.

Nota: Utilice el siguiente prototipo de función de ser posible:

- **unsigned long factorial(unsigned int numero);**

Ayuda: El factorial de un entero positivo n, el factorial de n o n factorial se define en principio como el producto de todos los números enteros positivos desde 1 (es decir, los números naturales) hasta n.
Por ejemplo:
$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120.$$

Ejercicio 1.6

Realizar funciones que puedan hacer las cuatro operaciones básicas: suma, resta, multiplicación y división.

Nota: Tener en cuenta que no se considera válida la división por cero. En caso de dividir por cero la función deberá retornar 0 e imprimir un mensaje de error en la consola.

Ejercicio 1.7

Integre todas las funciones del ítem anterior para lograr una calculadora.

Ejercicio 1.8

Escriba una función que reciba un valor de temperatura en alguna escala, y la escala de temperaturas de destino, y que luego realice la conversión del valor.

Nota: Las escalas posibles son: Celsius, Fahrenheit y Kelvin. Si tiene dudas respecto a las escalas ver explicación de la guia de ejercicios 1.

Ejercicio 1.9

Desarrolle una función que dado un número entero decimal imprima por pantalla el polinomio correspondiente y devuelva la cantidad de dígitos de su equivalente en binario,octal y hexadecimal según se solicite. Utilice el siguiente prototipo de función de ser posible:

- **int descomponer_numero(int numero, int sistema);**

Donde <numero> es el valor decimal a evaluar y < sistema> es binario(0), octal(1) o hexadecimal(2) según corresponda.
En caso de recibir un sistema no válido, la función retornará -1.
Ejemplos:

- Si se invoca a la función con numero=61 y sistema=0 debería mostrar en pantalla:
  1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 1*2^4 + 1*2^5 = 61 y retornar el valor 6.
- Si se invoca a la función también con numero=61 pero sistema=2 debería mostrar en pantalla:
  D*16^0 + 3*16^1 = 61 y retornar el valor 2.

Nota: Debe realizar todas las conversiones en forma manual

Ejercicio 1.10

Realizar un programa que reciba dos numeros enteros positivos. Si el primer número es par debera calcular el factorial del mismo pero, si es impar debera calcula la potencia del primero elevado por el segundo.

Ayuda: Puede invocar a las funciones hechas en los puntos 1.10 y 1.1 en caso de haberlas hecho.

Ejercicio 1.11

Realizar una función que dado un número entero en base 10 (decimal) que reciba como argumento realice su conversión a base 2 (binario) y la muestre por pantalla.

Ejercicio 1.12

Realizar una función que dado un número entero en base 10 (decimal) que reciba como argumento realice su conversión a base 8 (octal) y la muestre por pantalla.

Ejercicio 1.13

Realizar una función que dado un número entero en base 10 (decimal) que reciba como argumento realice su conversión a base 16 (hexadecimal) y la muestre por pantalla.

Ejercicio 1.14

Utilizar los puntos 1.12, 1.13 y 1.14 para realizar un programa que invoque a estas funciones y al recibir un número indique sus conversiones correspondientes. La función debe devolver una variable de tipo long double.

Nota: Si el número no es entero o no es posible calcular su conversión, se debera informar por pantalla.

Ejercicio 1.15

Realizar una función que informe cuales son las raíces de una ecuación cuadrática. Los argumentos de la función deben ser los componentes de la ecuación.

Ayuda: Recuerde que puede calcular las raices usando la fórmula de Bhaskara

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Nota: De tener raíces no reales la función debera informar que la raiz encontrada es un número complejo.

Ejercicio 1.16

Realizar una función que obtenga los factores positivos de un numero positivo y los muestre por pantalla.

Ayuda: Los *factores* pueden dividir exactamente a un número sin un residuo o decimal. Por ejemplo, 30 dividido entre 10 es 3, y 30 dividido entre 15 es 2, así que 2, 3, 10 y 15 son todos factores de 30.

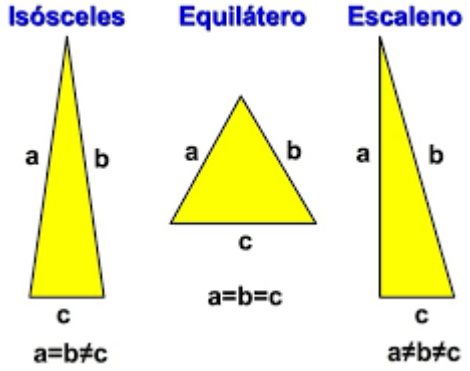
Ejercicio 1.17

Realice una función que reciba tres números que correspondan a los lados de un triángulo y determine si el mismo es: equilatero, escaleno o isósceles.

Nota: De ser posible, utilice el prototipo:

- **int obtener_tipo_triangulo_lados (float lado1, float lado2, float lado3);**

Ayuda:



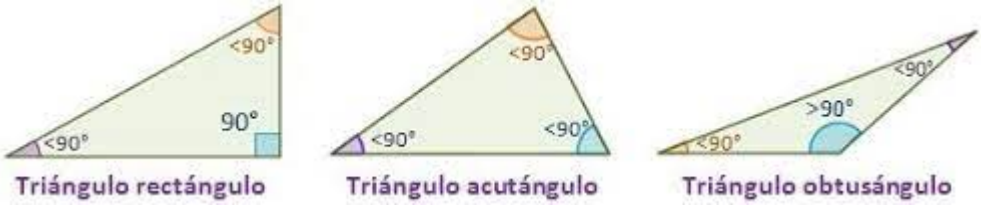
Ejercicio 1.18

Realice una función que reciba tres números que correspondan a los ángulos de un triángulo y determine si el mismo es: rectángulo, acutángulo u obtusángulo.

Nota: Utilice este prototipo de ser posible:

- **int obtener_tipo_triangulo_angulos (float angulo1, float angulo2, float angulo3);**

Ayuda:



Ejercicio 1.19

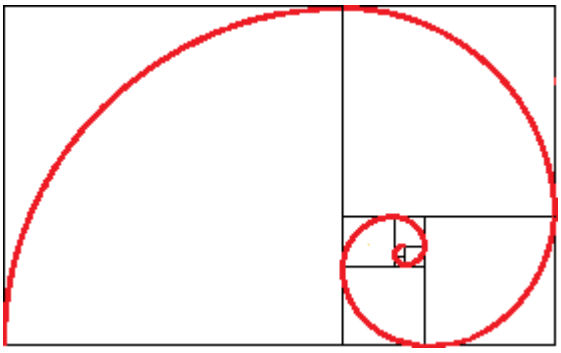
Realice una función que encuentre cuál es el valor númerico entero (que se encuentra entre -50 y 50) en que coinciden la escala Celsius y Fahrenheit.

Ayuda: La manera más óptima para encontrarlo es usar alguna estructura de iteración.

Ejercicio 1.20

Crear una función que emplee alguna estructura de iteración para calcular un término de la serie Fibonacci que reciba como argumento.
Utilice el siguiente prototipo de función de ser posible:

- **int fibonacci (int termino);**



0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Ayuda: La sucesión de Fibonacci es una sucesión definida por recurrencia. Esto significa que para calcular un término de la sucesión se necesitan los términos que le preceden.
Se proporcionan los dos primeros términos: a0=0 y a1=1. Los siguientes se calculan con la siguiente fórmula:

$$a_{n+1} = a_{n-1} + a_n\,, \quad n \geq 1$$

Ejercicio 1.21

Realizar una función que reciba una letra y un número, y escriba un “triángulo” formado por esa letra, que tenga como anchura inicial la que se ha indicado. Por ejemplo, si la letra es 'B' y la anchura es 4, debería escribir:

BBBB
BBB
BB
B

Ejercicio 1.22

Escriba un programa que sirva para ayudar a un alumno de primaria a aprender a multiplicar. Use *rand* para producir dos enteros positivos de un dígito. A continuación debería escribir una pregunta como la siguiente:
¿Cuánto es 6 veces 7?
Si la respuesta es correcta deberá llamar a una función generadora de respuestas y, solicita otra multiplicación. Si la respuesta es incorrecta deberá llamar a la función generadora de respuestas y luego, permitirá que el alumno vuelva a intentar la misma multiplicación hasta que la conteste correctamente. El programa finalizará con un saludo de despedida cuando el alumno haya respondido 10 preguntas correctamente.
En cuanto a la función generadora de respuestas:
Al variar el diálogo de la computadora en la instrucción asistida por computadora (CAI) se logra retener la atención del alumno. Por lo cual el objetivo de la función es imprimir un comentario aleatorio según la respuesta del alumno.
Respuestas a las contestaciones correctas:

- Muy Bien!
- Excelente!
- Buen trabajo!
- Felicitaciones por tu empeño

Respuestas a las contestaciones incorrectas:

- No, volve a intentarlo por favor.
- Incorrecto. Proba una vez más.
- No te rindas!
- No, seguí intentando!

Ayuda: Puede usar el generador de números aleatorios para elegir un número del 1 al 4 y seleccionar una respuesta apropiada para cada una de las contestaciones.

Ejercicio 1.23

Escriba una función que, dado un número entero que recibe como argumento, imprima dicho número en su equivalente en números romanos.
Los números que recibe la función representan años con lo cual en este caso el intervalo de ingreso posible es del 1 al 2021.
Validar el ingreso de datos dentro de la función. Es decir, de no ser posible calcular el número, la función deberá informarlo.

ROMANO	DECIMAL
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Ayuda: Separar en cifras primero para manejar cada conversión por separado.
Para escribir cualquier número en números romanos, se emplea una combinación de las siete letras anteriores y se deben cumplir las siguientes reglas:

- Si a la derecha de una letra romana se escribe otra igual o menor, el valor de ésta se suma a la anterior.
  VI = 6; XXI = 21; LXVII = 67
- Una letra colocada antes de alguna otra que sea mayor, le resta la unidad de la menor a la mayor. Ejs, La letra "X", precediendo a "L" o "C", resta diez unidades.
  IV = 4; IX = 9; XL = 40; XC = 90; CD = 400; CM = 900
- En ningún número se puede poner una misma letra más de tres veces seguidas.
  XIII = 13; XIV = 14; XXXIII = 33; XXXIV = 34
- Si entre dos cifras cualesquiera existe otra menor, ésta restará su valor a la siguiente.
  XIX = 19; LIV = 54; CXXIX = 129

Referencias

Algunos ejercicios fueron obtenidos y adaptados de:

- Guía de Trabajos Prácticos 2011 - Informática I - Departamento de Electrónica - UTN FRBA
- Cómo programar en C/C++ y Java Libro de Harvey M. Deitel