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.
```

Condiciones de 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 ejercicios en un zip (usar template como ayuda para el formato).

Ejercicio 3.1

Implementar 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`. Utilizar el siguiente prototipo:

```c
float mi_pow(float x, int y);
```

Ejercicio 3.2

Implementar una función que simule el tiro de un dado haciendo uso de la función estándar `rand()` para generar la secuencia pseudoaleatoria. Utilizar el siguiente prototipo:

```c
int tirar_dado();
```

Ayuda: Ver `man 3 rand`.

Ejercicio 3.3

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

- `0` si no es primo.
- `1` si es primo.
- `-1` en caso de error.

Utilizar el siguiente prototipo:

```c
int es_primo(int numero);
```

Nota: 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 3.4

Implementar 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 deberá retornar un `1`, caso contrario retornará `0`. Utilizar el siguiente prototipo:

```
int es_bisiesto(int anio);
```

Nota: Un año es bisiesto si es un número divisible por 4, pero no si es divisible por 100, excepto que también sea divisible por 400.

Ejercicio 3.5

Implementar una función que calcule el factorial de un número entero pasado como argumento. En caso no poder devolver el resultado correcto, se deberá retornar `0`. Utilizar el siguiente prototipo:

```
unsigned long factorial(int numero);
```

Nota: El factorial de un entero positivo `n` se define como el producto de todos los números enteros positivos desde `1` (es decir, los números naturales) hasta `n`.

Ejercicio 3.6

Implementar cinco funciones que computen las siguientes operaciones básicas: suma, resta, multiplicación, división y resto de la división. Utilizar los siguientes prototipos:

```
float sumar(float operando_a, float operando_b);

float restar(float operando_a, float operando_b);

float multiplicar(float multiplicando, float multiplicador);

float dividir(float dividendo, float divisor);

int resto(int dividendo, int divisor);
```

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

Ejercicio 3.7

Implementar una función que convierta un valor en grados Fahrenheit a grados Celsius y otra que efectúe el cálculo inverso. La fórmula que los relaciona es `°C = 5/9 * (°F - 32)`. Utilizar los siguientes prototipos:

```
float a_fahrenheit(float grados_celsius);

float a_celsius(float grados_fahrenheit);
```

Ejercicio 3.8

Implementar una función que convierta un valor en radianes a grados y otra que efectúe el cálculo inverso. La fórmula que los relaciona es `radianes = grados * (PI / 180)`. Utilizar los siguientes prototipos:

```
float a_radianes(float grados);

float a_grados(float radianes);
```

Ejercicio 3.9

Implementar una función que dado un número entero en sistema decimal, imprima por pantalla el polinomio correspondiente de su equivalente en sistema binario, octal o hexadecimal según se solicite. Utilizar el siguiente prototipo:

```
void descomponer_numero(int numero, int sistema);
```

Donde `numero` es el número en sistema decimal a evaluar y `sistema` podrá ser binario (`0`), octal (`1`) o hexadecimal (`2`) según corresponda. En caso de recibir un sistema no válido, la función imprimirá un mensaje de error.

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
```

- Si se invoca a la función también con `numero=61` pero `sistema=2` debería mostrar en pantalla:

```
D*16^0 + 3*16^1 = 61
```

Nota: Se deben realizar todas las conversiones en forma manual (por ejemplo, no usar modificadores de formato de `printf`).

Ejercicio 3.10

El algoritmo babilónico para calcular la raíz cuadrada de un número `x` consta de los siguientes pasos:

1. Se propone que el resultado de la raíz `b` vale `x`.
2. Se inicializa `h` en `1`.
3. Mientras que la diferencia entre `b` y `h` sea superior a nuestro margen de error (`0.001`):
    - El nuevo valor de `b` se calcula como el promedio entre `b` y `h`.
    - El nuevo valor de `h` se calcula como `x/b`.

Implementar una función que calcule la raíz cuadrada de `x` según este método. Utilizar el siguiente prototipo:

```
float raiz_cuadrada(float x);
```

Ejercicio 3.11

El valor aproximado del número de Euler (`e`) se puede obtener con la siguiente fórmula:

```
e = 1 + 1/1! + 1/2! + 1/3! + 1/4! + 1/5! + ...
```

Implementar una función que calcule el valor aproximado de `e` mediante un ciclo repetitivo que termine cuando la diferencia entre dos aproximaciones sucesivas difiera en menos de `10^-9`. Utilizar el siguiente prototipo:

```
double aproximar_e();
```

Ejercicio 3.12

Implementar una función que obtenga los factores positivos de un número positivo y los muestre por pantalla. Utilizar el siguiente prototipo:

```
void factores(int numero);
```

Ayuda: Los factores son aquellos números enteros que pueden dividir exactamente a otro número entero 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 3.13

Implementar una función que reciba tres números que corresponden a los lados de un triángulo. La función debe retornar:

- `0` si es equilátero.
- `1` si es isósceles.
- `2` si es escaleno.

Utilizar el siguiente prototipo:

```
int clasificar_triangulo_lados(float lado1, float lado2, float lado3);
```

Ejercicio 3.14

Implementar una función que reciba tres números que corresponden a los ángulos de un triángulo. La función debe retornar:

- `0` si es acutángulo.
- `1` si es rectángulo.
- `2` si es obtusángulo.

Utilizar el siguiente prototipo:

```
int clasificar_triangulo_angulos(float angulo1, float angulo2, float angulo3);
```

Ejercicio 3.15

Implementar una función que calcule (y retorne) el n-ésimo número de la sucesión de Fibonacci e imprima la serie hasta dicho número. La misma queda definida por las siguientes ecuaciones:

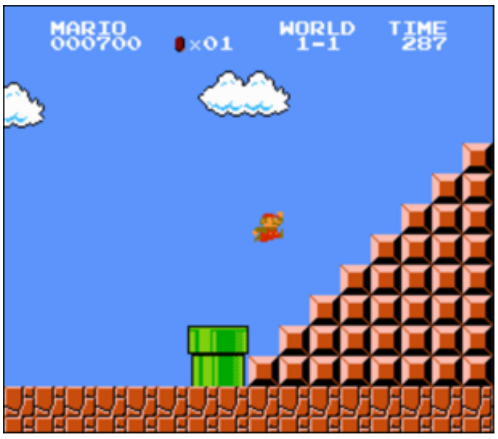
- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2)

Utilizar el siguiente prototipo:

```
int fibonacci(int n);
```

Ejercicio 3.16

Cerca del final del nivel 1-1 en Super Mario Bros de Nintendo, Mario debe saltar sobre pirámides de bloques adyacentes, como se muestra a continuación.



Codifiquemos dichas pirámides en C, aunque en texto, usando numerales (`#`) para los ladrillos, como se muestra a continuación (cada numeral es un poco más alto que ancho, por lo que las pirámides en sí también son más altas que anchas):

```
   #
  ##
 ###
####
```

Implementar una función que dibuje una pirámide, recibiendo como argumento el alto que deberá tener la misma. Utilizar el siguiente prototipo:

```
void piramide(int alto);
```

Ejercicio 3.17

Implementar una función generadora de respuestas aleatorias, la cual será utilizada posteriormente en un programa de preguntas y respuestas. Al variar el diálogo de la computadora en la instrucción asistida por computadora (CAI) se logra retener la atención del usuario, por lo cual el objetivo de la función es imprimir un comentario **aleatorio** según la contestación del usuario.

Comentario a las respuestas correctas:

- Muy Bien!
- Excelente!
- Buen trabajo!
- Correcto, felicitaciones!

Comentario a las respuestas incorrectas:

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

Utilizar el siguiente prototipo:

```
void generar_mensaje(int tipo_contestacion);
```

El argumento `tipo_contestacion` indicará si se deberá generar una respuesta positiva (`1`) o negativa (`0`).

Ayuda: Se sugiere hacer uso de la función estándar `rand()` para generar la secuencia pseudoaleatoria. Ver `man 3 rand`.

Ejercicio 3.18

Implementar 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 representarán años con lo cual en este caso el intervalo de ingreso posible es del 1 al 2022. Utilizar el siguiente prototipo:

```
void a_romanos(int anio);
```

Ayuda: Conviene separar en cifras primero para manejar cada conversión por separado. Para escribir cualquier número en números romanos, se emplea una combinación de siete letras (`I = 1`, `V = 5`, `X = 10`, `L = 50`, `C = 100`, `D = 500` y `M = 1000`) 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. Por ejemplo, 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
```

Ejercicio 3.19

Implementar una función que dado un número entero devuelva su valor absoluto. Utilizar el siguiente prototipo:

```
int abs(int n);
```

Ejercicio 3.20

Implementar una función que dada una dimensión imprima la matriz identidad de esa dimensión. Utilizar el siguiente prototipo:

```
void matriz_identidad(int dimension);
```

Ejercicio 3.21

Implementar una función que dadas dos rectas definidas por su pendiente y su ordenada al origen devuelva la abscisa en la que se intersectan. Validar lo que considere necesario. Utilizar el siguiente prototipo:

```
float interseccion_rectas(float m1, float b1, float m2, float b2);
```

Ejercicio 3.22

Implementar una función que implemente el algoritmo de Euclides para calcular el máximo común divisor de dos números `n` y `m`, dado por los siguientes pasos:

1. Teniendo `n` y `m`, se obtiene `r`, el resto de la división **entera** de `m/n`.
2. Si `r` es cero, `n` es el MCD de los valores iniciales.
3. Se reemplaza `m <- n`, `n <- r`, y se vuelve al primer paso.

Utilizar el siguiente prototipo:

```
int mcd(int n, int m);
```

Referencias

Algunos ejercicios fueron obtenidos y adaptados de:

- Guía de Trabajos Prácticos 2011 - Informática I - Departamento de Electrónica - UTN FRBA
- Guía de Ejercicios - Algoritmos y Programación I - UBA FIUBA
- Set de Problemas - CS50 - Harvard University
- Cómo programar en C/C++ y Java - Harvey M. Deitel y Paul J. Deitel