

GUÍA PRÁCTICA DE FUNCIONES

Resolvé los ejercicios de utilizando el lenguaje C. Asegurate de leer al menos dos veces los enunciados antes de intentar confeccionar las soluciones.

ENUNCIADOS

Definición y testeo de funciones

Definí las siguientes funciones, asegurándote su correcto funcionamiento con algunos tests.

- 1) **esPar**, que devuelva si un número entero dado como parámetro es par o no.
- 2) **obtenerResto**, que devuelva el resto del cociente entre dos números enteros dados como parámetros (sin usar el operador %).
- 3) **imprimirSimbolo**, que imprima por consola **n** veces un caracter en la misma línea. Tanto **n** como el caracter se reciben como parámetro.
- 4) **esVocal**, que devuelva si un caracter dado como parámetro es o no una letra vocal (contemplar mayúsculas y minúsculas).
- 5) **mostrarSucesion**, que imprima por consola la sucesión de números inclusiva entre **a** y **b**, dados como parámetros.
- 6) **esMultiplo**, que devuelva si un número entero es múltiplo de otro. Ambos son dados como parámetros.
- 7) **cantDivisores**, que devuelva la cantidad de divisores que posea un número entero dado como parámetro.
- 8) **esPrimo**, que devuelva si un número entero dado como parámetro es o no primo.
- 9) **mostrarNPrimos**, que muestre por la consola, separados por comas, los primeros **n** números primos. El valor de **n** se recibe como parámetro.
- 10) **esNumeroPerfecto**, que devuelva si un número entero dado como parámetro es o no perfecto.

Desarrollo de programas modularizados

- 11) Realizá un programa que permita al usuario ingresar el valor unitario de cierto artículo y la cantidad de artículos vendidos por un vendedor, del cual se sabe que gana un sueldo fijo de \$14000 más el 16% del monto total vendido. Con tales datos, la computadora debe calcular el monto a cobrar por el vendedor y mostrarlo.
- 12) Realizá un programa que permita al usuario ingresar su edad (entre **1** y **120** años) y su género ('F' o 'M'). La computadora debe indicar si la persona está o no en edad de jubilarse¹.
- 13) Realizá un programa que permita al usuario ingresar la cantidad de cierto producto y el precio unitario de dicho producto. Por cada carga debe preguntar si se desea seguir ingresando de la forma "**¿Deseás ingresar otro artículo? [S/N]**". La carga de datos finaliza cuando el usuario lo determine. La computadora debe mostrar el monto total del ticket.
- 14) Realizá un programa que permita al usuario ingresar los datos de cada uno de los 8 choferes de una empresa de viajes en ómnibus:

- Edad [25 a 40]
- Antigüedad [5 a 30]
- Turno ('M' | 'T' | 'N')

Todos los choferes obtendrán un bono de 500 dólares. Los que tengan una antigüedad superior a la mitad de su edad, cobrarán un 10% extra. Los del turno nocturno cobrarán otro 5% extra.

La computadora debe mostrar, de forma clara y ordenada:

¹ Las mujeres se jubilan con 60 años o más. Los hombres se jubilan con 65 años o más.

- A) El monto total que la empresa paga en concepto de bono para todos sus choferes.
- B) En cuál turno los choferes son, en promedio, más jóvenes.

Crear librería con funciones útiles

Luego de desarrollar ejercicios de forma modularizada, observamos que muchas operaciones en diferentes ejercicios parecen ser comunes. Por ejemplo: ¿habrá mucha diferencia entre un algoritmo que lea un entero validado entre 70 y 120 a otro que lea un entero validado entre 1 y 10?

Este tipo de hallazgo debe motivarnos a desarrollar, por única vez, una librería de funciones reutilizables a lo largo de toda la cursada, de manera tal de no repensar los mismos problemas de similar comportamiento por cada nuevo ejercicio a realizar.

Al incluir una librería de C como `<stdio.h>`, podemos obtener acceso a funciones como `scanf()` y `printf()`, definidas una única vez en tal librería y utilizadas por cualquier programador/a en cualquier solución a lo largo de la historia y el mundo, abstrayéndose de cómo fueron implementadas y cómo funcionan internamente.

Emularemos esta misma dinámica, creando una librería de funciones de entrada y salida de datos por consola, que nos sirvan a futuro tanto a nosotros como a cualquier colega.

Para ello, generará un archivo aparte llamado `utils.h`, que permitirá guardar las cabeceras (prototipos) de las funciones reutilizables que implementaremos luego, y copia este código:

```
#ifndef UTILS_H_INCLUDED
#define UTILS_H_INCLUDED
#include <stdio.h>
#include <string.h> // Este #include servirá para más adelante...

typedef char cadena[50]; // Para tratar los strings usando el tipo 'cadena'

int leerEntero(cadena mensaje);
float leerFloat(cadena mensaje);
char leerCaracter(cadena mensaje);
int leerEnteroEntre(int valorMin, int valorMax, cadena mensaje);
float leerFloatEntre(float valorMin, float valorMax, cadena mensaje);
int confirmaUsuario(cadena mensaje);

#include "utils.c" // Este #include traerá las implementaciones...
#endif // UTILS_H_INCLUDED
```

`utils.h`

A continuación, generará un nuevo archivo aparte, llamado `utils.c`, que tenga las implementaciones de las funciones anteriores, completando el código según lo que se describe a continuación:

```
int leerEntero(cadena mensaje) {
    /* Muestra el mensaje al usuario, lee un entero y lo retorna */
}

float leerFloat(cadena mensaje) {
    /* Muestra el mensaje al usuario, lee un float y lo retorna */
}
```

`utils.c`

```
char leerCaracter(cadena mensaje) {
    /* Muestra el mensaje al usuario, lee un char y lo retorna */
}

int leerEnteroEntre(int valorMin, int valorMax, cadena mensaje) {
    /* Muestra el mensaje al usuario. Luego lee enteros mientras éstos estén
    fuera del rango [valorMin, valorMax] informando al usuario de
    su error. Cuando finalmente lea un entero válido, lo retorna */
}

float leerFloatEntre(float valorMin, float valorMax, cadena mensaje) {
    /* Muestra el mensaje al usuario. Luego, lee floats mientras éstos estén
    fuera del rango [valorMin, valorMax] informando al usuario de
    su error. Cuando finalmente lea un float válido, lo retorna */
}

int confirmaUsuario(cadena mensaje) {
    /* Muestra el mensaje al usuario junto a la leyenda "[S/N]" que
    representa "Si" o "No". Luego, lee chars mientras éstos no sean los
    previstos ('S' o 'N', incluyendo minúsculas) informando al
    usuario de su error. Retorna si el usuario seleccionó que sí. */
}
```

Por último, vamos a testear nuestra librería para comprobar si funciona correctamente, creando un archivo aparte llamado **test-utils.c** y copiando el siguiente código:

```
#include "utils.h"

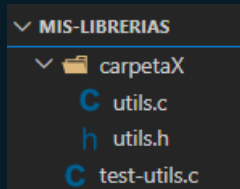
int main () {
    testLeerEntero();
    testLeerFloat();
    testLeerCaracter();
    testLeerEnteroEntre();
    testLeerFloatEntre();
    testConfirmaUsuario();
    return 0;
}

void testLeerEntero() {
    printf("Test de leerEntero\n");
    int edad = leerEntero("Edad?");
    printf("%d\n\n", edad);
}

void testLeerFloat() {
    printf("Test de leerFloat\n");
    float edad = leerFloat("Precio?");
```

test-utils.c

Debe ser la ruta relativa al archivo actual.
Por ejemplo, si fuera el caso de la imagen,
la ruta sería: "carpetaX/utils.h"



```
    printf("%f\n\n", edad);  
}  
void testLeerCaracter() {  
    printf("Test de leerCaracter\n");  
    char genero = leerCaracter("Genero?");  
    printf("%c\n\n", genero);  
}  
void testLeerEnteroEntre() {  
    printf("Test de leerEnteroEntre\n");  
    int edad = leerEnteroEntre(18, 60, "Edad?");  
    printf("%d\n\n", edad);  
}  
void testLeerFloatEntre() {  
    printf("Test de leerFloatEntre\n");  
    float edad = leerFloatEntre(5, 50.15, "Precio?");  
    printf("%f\n\n", edad);  
}  
void testConfirmaUsuario() {  
    printf("Test de confirmaUsuario\n");  
    int conf = confirmaUsuario("Llueve?");  
    printf("%d\n\n", conf);  
}
```