Guía
Ejercicios en C

Resuelva los siguientes ejercicios en lenguaje C.

Ejercicios Básicos

Secuencia

1) Hallar la superficie de un triángulo conociendo la base y la altura. Solicitarle los datos de entrada al usuario.

2) Calcular el sueldo de un operario ingresando por teclado la cantidad de horas que trabajó en el mes y el valor de la hora. Mostrarle el resultado al usuario con un mensaje adecuado.

Condicionales

3) Ingresar un número entero y decir si: a) es par o impar. b) es mayor, menor o igual a cero.

4) Solicitar al usuario el ingreso de una temperatura **(puede tener decimales por ejemplo 24.5)** y la unidad en la que se encuentra **(siendo solo un carácter F ó C)**. Luego el programa debe mostrar la temperatura ingresada, convertida en la otra unidad.
La relación entre temperaturas Celsius y Fahrenheit está dada por la fórmula:
C=5.0 / 9.0 * (F − 32)

Iteración

5) Solicitar un número entero positivo al usuario y calcular su factorial. En el caso de ingresar un número negativo mostrar un mensaje que diga “no se puede calcular el factorial del número ingresado”.
Recordar que por definición factorial(0)=1 y factorial(1)=1

6) Dada una serie de números ingresados de a uno. Indicar mayor, menor y promedio de la serie. El ingreso de números finaliza cuando el usuario ingresa 0.

7) Dado un número entero positivo ingresado por el usuario, procesarlo e indicar: (realizar un programa diferente para cada caso)

a) La cantidad de dígitos pares e impares que lo componen.
b) El menor y el mayor dígito del número.

Recordar uso de división y módulo:
Ejemplo:
111 / 10 = 11
111 % 10 = 1

Ejercicios Funciones

Importante: Luego de escribir cada función, probala, invocándola desde el bloque principal del programa, pasándole distintos valores para que la prueba contemple varias alternativas y así estar seguro que funciona adecuadamente.

1) Escribir una función que reciba un valor n, entero, y devuelva la suma de los valores entre 0 y n.
Ejemplos:
suma_n(5) = 15
suma_n(120) = 7260

2) Escribir una función que dado un número entero, devuelva un valor booleano que indique si dicho número es primo o no.

3) Escribir una función que reciba un valor entero y calcule el factorial del mismo. Si no se puede calcular el factorial del valor recibido, la función deberá devolver 0, de lo contrario deberá devolver el valor calculado.

4) Escribir una función que dado un tiempo expresado en segundos, devuelva por parámetros el equivalente en días, horas, minutos y segundos. Utilizar esta

función dentro de un programa que solicite el valor al usuario. Se debe validar que el valor ingresado sea entero positivo, de lo contrario, deberá mostrarse el mensaje: “Valor ingresado inválido”.

Ejemplo:
Valor ingresado representando una cantidad de segundos: 1234567
Resultado: Días: 14, Horas: 6, Minutos: 56, Segundos 7

Para verificar el resultado pueden ir a la siguiente web:
https://www.satsig.net/training/seconds-days-hours-minutes-calculator.htm

Ejercicios Arreglos

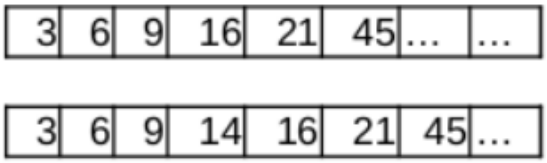
1) Dado un listado de números reales del cual no se conoce la cantidad, almacenar los números en un vector en el orden de entrada.
Informar la cantidad de números y el contenido del vector indicando la posición ocupada por cada número a partir de la primera posición.
Considerar una estructura de datos de tamaño físico máximo de 1000 posiciones.

2) Escribir una función en C, que reciba:
a) como primer parámetro un vector de números enteros;
b) como segundó parámetro, la cantidad de elementos en el vector;
c) como tercer parámetro deberá devolver la cantidad de valores negativos que hay en el vector recibido;
d) como cuarto parámetro, la cantidad de elementos positivos que hay en el vector recibido.

Escribir el programa que incluya a la función y las invocaciones con los siguientes ejemplos, y la respectiva impresión de los valores devueltos:

Probar el programa con los siguiente casos:
v1 = [2,8,1,-5,4] -> Negativos: 1 Positivos: 4
v2 = [-2,-15,-3] -> Negativos: 3 Positivos: 0
v3 = [0,0,10,12,23,55,1] -> Negativos: 0 Positivos: 5

3) Dado un vector **a** ordenado **ascendente** de longitud **ml** y un elemento **p** del mismo tipo que los elementos del vector, insertar **p** en el vector a de modo que siga ordenado. Validar previamente que en el vector haya espacio libre para guardar el nuevo dato. Se solicita resolver lo solicitado recorriendo una sola vez el vector y sin utilizar un arreglo auxiliar.
Ejemplo:
nuevo elemento p=14



Ejercicios Cadenas

1) Escribir una función para validar una nueva clave de acceso. La función deberá recibir una cadena de caracteres, que contendrá la clave candidata, que ya fue ingresada previamente por el usuario.
Devolverá true o false, dependiendo de si cumple o no, con las siguientes condiciones:
- La clave debe estar formada únicamente por, entre 6 y 12 caracteres numéricos
- La cantidad de dígitos pares debe ser mayor a la de los impares.

A los sumo debe recorrer una vez la cadena.
Evite realizar ciclos innecesarios.
Compruebe el correcto funcionamiento, incluyendo los siguientes casos de prueba:
validar("j20893") devuelve false
validar("20893a") devuelve false
validar("208X930") devuelve false
validar("20201") devuelve false
validar("23445776") devuelve false
validar("089010") devuelve true
validar("02784532132567") devuelve false
validar("027845320011") devuelve true

2) Escribir una función en C, que reciba una cadena que representa una palabra y devuelva si la misma es o no un palíndromo. Una palabra es un palíndromo, si se lee igual en ambos sentidos.
Probar la función con los siguientes casos de prueba:
- anilina (Es palíndromo)
- ojo (Es palíndromo)
- radar (Es palíndromo)
- reconocer (Es palíndromo)
- algoritmos (No es palíndromo)
- programas (No es palíndromo)

Evitar realizar ciclos innecesarios.

3) Escribir un programa modular en C, que solicite el ingreso de 3 oraciones, de no más de 50 caracteres cada una. Luego informar: 1. Cuál es la oración más larga. 2. Si hay al menos 2 oraciones iguales. 3. Solicitar el ingreso de una palabra o parte de una oración, e indicar si la misma se encuentra en las oraciones, y en cuales.

Ejercicios Memoria Dinámica

1) Escribir un programa el cual reserve memoria dinámica para almacenar un número entero (int), le solicite al usuario el ingreso de un número y se asigna dicho valor en la memoria reservada, luego mostrar dicho valor por pantalla Liberar la memoria reservada al finalizar el programa.

2) Escribir un programa el cual reserve memoria dinámica para almacenar una cierta cantidad de números enteros (n * int), este valor **n** debe ser ingresado por el usuario. Luego solicitarle que ingresé n valores enteros ingresados de a uno y almacenarlos en la memoria previamente reservada. Mostrar luego todos los valores ingresados. Liberar la memoria reservada al finalizar el programa.

Ejercicios Recursividad

1) Desarrollar un programa que calcule la factorial de un número en forma recursiva.

2) Desarrollar un programa que calcule y muestre por pantalla los primeros N términos de la sucesión de Fibonacci en forma recursiva. N es un número ingresado por el usuario.

3) Desarrollar una función recursiva para realizar una búsqueda binaria en un vector.

4) Desarrollar una función recursiva que compruebe si un número es binario. Un número binario está formado únicamente por ceros y unos.
Sugerencia: recordar el uso de operadores **% (mod)** y **/ (div)**
202 % 10 = 2
202 / 10 = 20

Ejemplos:
es_binario(101) -> true
es_binario(2) -> false
es_binario(20) -> false
es_binario(1) -> true
es_binario(0) -> true
es_binario(100000) -> true
es_binario(100009) -> false

5) Desarrollar una función recursiva que reciba un arreglo de enteros y su máximo lógico. La función debe retornar la cantidad de números pares presentes en el arreglo.

Ejemplo:
[23, 44, 68, 2, 24, 12] -> 5