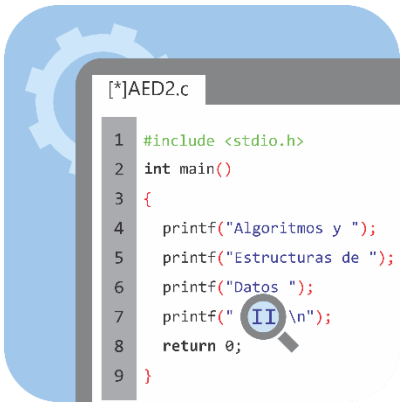

Algoritmos y Estructuras de Datos II

Práctico Nro 7: Recursividad.

OBJETIVOS:

Que el alumno:

- Se familiarice con los conceptos de Recursividad.
- Identifique las diferencias de la implementación iterativa versus recursiva.
- Aprenda a implementar algoritmos recursivos.

METODOLOGÍA

- Lectura de la conceptualización de recursividad.
- El alumno deberá resolver individualmente los ejercicios propuestos
- Se podrá realizar trabajos en grupos para consolidar conceptos, comprensión de lo solicitado y alternativas de solución.
- El alumno deberá codificar las soluciones que proponga de cada uno de los ejercicios propuestos en las clases prácticas de laboratorio.
- Interactuar en el aula virtual de la asignatura.

DURACIÓN

De acuerdo a la planificación de la asignatura, se deberán utilizar para la resolución de los ejercicios de esta serie, no más de dos clases prácticas.

EJERCICIOS PROPUESTOS

1. Escribir un programa que permita mostrar por pantalla una cuenta regresiva, a partir de un valor ingresado por teclado. Programar una función recursiva que, al llegar la cuenta a cero, informe que el tiempo se ha agotado.

2. La cátedra de AED II dispone de un programa con funciones que sirven para agilizar la corrección de exámenes, en este caso, requiere el desarrollo de una función que, a partir del ingreso de un número entero, y mediante la utilización de una función recursiva, permita visualizar el número binario equivalente por pantalla.

3. Escribir un programa que permita ingresar un número entero positivo y luego, mediante una función recursiva, muestre el número de forma invertida. Ej.: 123 - 321.

4. Se cuenta con el código de un programa que permite, a partir del ingreso por teclado de dos números enteros, ver por pantalla el resultado de la división entera de los mismos. Modificar la función "*division(int, int)*", de forma tal que continúe realizando la división y devolviendo el resultado, pero ahora lo haga de manera recursiva.

```c
#include <stdio.h>
int division(int, int);
int main (){
    int resultado, valor1, valor2;
    printf("Ingrese los valores a dividir: \n");
    printf("Valor 1: ");
    scanf("%d", &valor1);
    printf("Valor 2: ");
    scanf("%d", &valor2);
    resultado = division(valor1, valor2);
    printf("Resultado: %d / %d = %d", valor1, valor2, resultado);
}
int division(int p_a, int p_b){
    if (p_b != 0){
        return p_a / p_b;
    } else {
        return 0;
    }
}
```

Nota

La división matemática se define como una operación aritmética, que consiste en saber cuántas veces un número (divisor) está contenido en otra cifra (dividendo).

5. Modificar el programa codificado en el ejercicio 1 de esta serie, para que la cuenta regresiva se realice de a un segundo por vez.

Nota

La función **sleep** permite suspender la ejecución de un programa por un número de segundos determinados.

Su prototipo es: void sleep(time_t seconds);

Si el compilador no lo detecta automáticamente, agregar:
Para Linux: #include <unistd.h>
Para Windows: #include <windows.h>

6. Escribir un programa que, a partir del ingreso de dos números enteros positivos, calcule el producto de los mismos utilizando una función recursiva. Tener presente que la definición recursiva de la multiplicación de dos números a y b, se deriva de la definición de la multiplicación como una suma abreviada y la aplicación de la propiedad asociativa de la suma. La definición recursiva de la multiplicación es:

$$a * b = \begin{cases} a + (a * (b - 1)) & \text{si } b > 0 \\ 0 & \text{si } b = 0 \end{cases}$$

7. utilizando funciones recursivas, escribir un programa que calcule el máximo común divisor de dos números enteros positivos ingresados por teclado. El cálculo del máximo común divisor se basa en la siguiente propiedad de los números enteros:

$$m.c.d.(a,b) = \begin{cases} m.c.d.(a-b,b) & \text{si } a \geq b \\ m.c.d.(a,b-a) & \text{si } b > a \\ a & \text{si } b = 0 \\ b & \text{si } a = 0 \end{cases}$$

8. El siguiente pseudocódigo corresponde a una función que permite sumar los elementos de un vector en forma recursiva:

Función: sumaVec
Datos de entrada: un vector de números enteros, representado por la pareja de datos <*v*, *n*>. como precondición se considera que el vector ya tiene cargados un conjunto de valores válidos y que *n* es un valor comprendido entre el cero (el vector puede estar vacío) y NMAX (la dimensión máxima del array).
Datos de salida: la suma de los elementos del vector, es decir,
sumaVec(v, n) = ∑v[i], i = 0, ..., n

```
Función sumaVec ( v: tArray; n: Entero): Entero
Inicio
    Si (n = 0) Entonces
        { la solución para el caso base es directa }
        retorna 0;
    Sino
        { la solución para el caso general es recursiva }
        retorna sumaVec ( v, n-1 ) + v[n];
    Fin-Si
 Fin
```

Desarrollar el código C correspondiente al pseudocódigo dado, tener presente al codificar el manejo del índice del arreglo.

9. Escribir un programa que permita el ingreso de dos valores enteros (la base entera y el exponente entero positivo), calcule la potencia y muestre los resultados por pantalla. Utilizar función recursiva.

La definición recursiva de la operación exponenciación entera, es decir calcular la potencia de a^b, se deriva de la definición de la potencia como una multiplicación abreviada y la aplicación de la propiedad asociativa de la multiplicación. Entonces, la definición recursiva de exponenciación es:

$$a^b = \begin{cases} a * a^{b-1} & \text{si } b > 0 \\ 0 & \text{si } b = 0 \end{cases}$$

10. Escribir un programa que, mediante funciones, determine la suma de "N" números naturales. Mostrar la serie de los números desde 1 hasta N y el resultado de la suma. Utilizar recursividad.

11. Escribir un programa que permita ingresar una palabra y determinar si es palíndroma. Usar una función recursiva.

Nota

Un **palíndromo** es un término o una expresión que puede leerse tanto de izquierda a derecha como de derecha a izquierda (es decir, expresa lo mismo al ser leído de manera tradicional o al revés). Se trata del equivalente a lo que, respecto a los números, se conoce como capicúa.

12. Codificar un programa que, a partir del ingreso por teclado de un número entero positivo, permita calcular su factorial y visualizar el resultado por pantalla. Utilizar una función recursiva para el cálculo factorial.

$$n! = \begin{cases} 1 \rightarrow \text{Si } n = 0 \\ (n-1)! * n \rightarrow \text{Si } n > 0 \end{cases}$$

13. Codificar un programa que, a partir del ingreso de un número entero positivo, calcule y muestre los números de la sucesión de Fibonacci. Utilizar una función recursiva.

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{(n-1)} + F_{(n-2)}\ ,\ n > 1$$