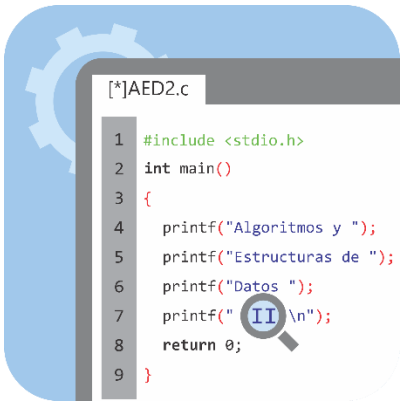

Algoritmos y Estructuras de Datos II

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.
- Implemente soluciones recursivas para el tratamiento de estructuras de datos no lineales (árboles y grafos).
- Continúe en el desarrollo de implementar las soluciones de problemas con un enfoque estructurado.

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 (2) clases prácticas.

Ejercicios propuestos

Ejercicio 1: Se desea codificar un programa que permita mostrar por pantalla una cuenta regresiva, a partir de un valor (a descontar) ingresado por teclado. Programar una función recursiva que, al llegar la cuenta a 0 informe que el tiempo se ha agotado.

Ejercicio 2: La cátedra de AED I necesita para completar un programa que agilice la corrección de exámenes, un módulo que a partir del ingreso de un numero entero, y mediante la utilización de una función recursiva, muestre el número binario equivalente por pantalla.

Ejercicio 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.

Ejercicio 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.

```
#include <stdio.h>
int division(int, int);
int main (){
	int resultado, valor1, valor2;
	printf("Ingrese los valores a dividir: \n");
	scanf("%d", &valor1);
	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;
	}
}
```

Ejercicio 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 librería dos.h incluye funciones que le permitirán manejar el retardo en segundos.

Ejercicio 6: Escriba 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:

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

Ejercicio 7: utilizando funciones recursivas, escriba 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:

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

Ejercicio 8: Se cuenta con el pseudocódigo correspondiente a una función que permite sumar los elementos de un vector en forma recursiva. Desarrolle el código correspondiente.

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

Nota: Tener presente al codificar en el lenguaje C, el manejo del índice del arreglo.

Ejercicios complementarios

Ejercicio 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 ab, 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:

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

Ejercicio 10: Escriba un programa que, mediante funciones, determine la suma de los "N" números naturales. Muestre la serie de los números desde 1 hasta N y el resultado de la suma. Utilice recursividad.

Ejercicio 11: Escriba un programa que ingrese una palabra y determine si es palindroma. Use función recursiva.

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.

Ejercicio 12: Codifique un programa que, a partir del ingreso por teclado de un número entero positivo, permita calcular su factorial y muestre el resultado por pantalla.
Utilice una función recursiva para el cálculo factorial.

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

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

$\mathbf{F_0} = 0$

$\mathbf{F_1} = 1$

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