



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: **Manuel Enrique Castañeda Castañeda**

Asignatura:

Estructura de Datos y Algoritmos I

Grupo:

12

No de Práctica(s):

12.

Integrante(s):

- **Lemus Ambrosio Aline Andrea**
- **Reyes Fuentes José Manuel**
- **Sánchez Alvirde Andrés Iván**

*No. de Equipo de cómputo
empleado:*

-

No. de Lista o Brigada:

12

Semestre:

2021-2

Fecha de entrega:

21 de agosto de 2021

Observaciones:

Calificación:

Estructura de Datos y Algoritmos I

Práctica 12
Recursividad

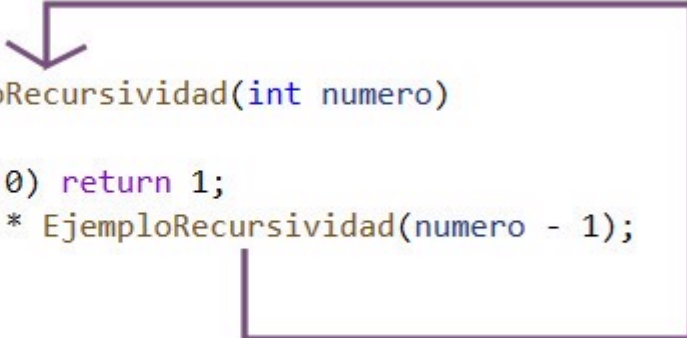
INTRODUCCIÓN

En esta práctica pudimos ver la Recursividad en algunos programas en los cuales el profesor nos apoyó en la realización de los mismos; pero sabremos un poco más acerca de la Recursividad en los programas de Dev c + +.

Se dice que una función es recursiva cuando se define en función de si misma, su propósito es dividir un problema en problemas más pequeños

No todas las funciones pueden llamarse a sí mismas, sino que deben estar diseñadas especialmente para que sean recursivas, de otro modo podrían conducir a bucles infinitos, o que el programa termine de una forma no favorable

Cuando creamos un método recursivo debemos tener en cuenta que este tiene que terminar por lo que dentro del método debemos asegurarnos de que no se está llamando a sí mismo todo el rato, Lo que quiere decir que el ciclo es finito.



```
1 reference
public int EjemploRecursividad(int numero)
{
    if (numero == 0) return 1;
    return numero * EjemploRecursividad(numero - 1);
}
```

Podemos utilizar recursividad para reemplazar cualquier tipo de bucle.

Recursividad directa vs indirecta.

Cuando en una subrutina hay llamadas a ella misma se habla de recursividad directa, en contraposición, cuando se tienen varias subrutinas y éstas se llaman unas a otras formando ciclos se dice que la recursión es indirecta.

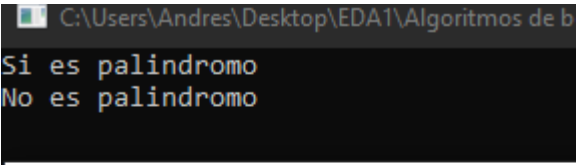
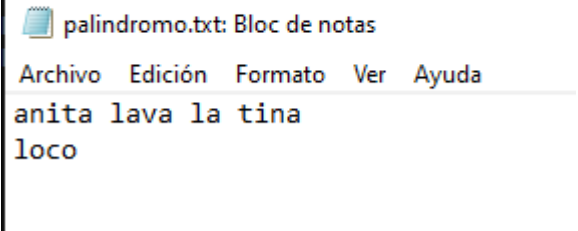
Subrutina IA → Subrutina IA → Subrutina IA

Subrutina IA → Subrutina → Subrutina C → Subrutina D → Subrutina IA

En general, cuando encaramos la tarea de escribir un programa para resolver un problema no hay razón para buscar una solución recursiva. La mayoría de los problemas pueden resolverse de una manera directa usando métodos no recursivos. Sin embargo, otros pueden resolverse de una manera más lógica y elegante mediante la recursión.

DESARROLLO

1. Palíndromo usando recursividad

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include <stdio.h> #include <string.h> #include <stdlib.h> int espalindromo(char* cadena); int main(){ char cad[30]; FILE *ap=fopen("palindromo.txt","r");//accediendo a un archivo de texto while(feof(ap)==NULL){ fgets(cad,100,ap);//leyendo cadena del archivo if(espalindromo(cad)){ printf("Si es palindromo\n"); }else{ printf("No es palindromo\n"); } } } int espalindromo(char *cadena){ int j=0,i; if(strlen(cadena)==1){ return 1; } if(cadena[0]==cadena[strlen(cadena)-2]){ for(i=0;i<strlen(cadena)-1;i++){ cadena[i]=cadena[i+1]; j=1; } for(i=j;i<strlen(cadena)-1;i++){ cadena[i]=NULL; } espalindromo(cadena); }else{ return 0; } }</pre>	 

2. Factorial usando recursividad

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include<stdio.h> int factorialNumero(int n) { if (n > 1) { return n * factorialNumero(n-1); } else { return 1; } } int main() { int j; printf("Recursividad para obtener el Factorial.\n\n"); printf("Ingrese el numero al cual se le obtendra su factorial:\n"); scanf("%d",&j); int resultado = factorialNumero(j); printf("\nEl resultado es: %d \n\n", resultado); }</pre>	<pre>Recursividad para obtener el Factorial. Ingrese el numero al cual se le obtendra su factorial: 10 El resultado es: 3628800</pre>

OBSERVACIONES (Individuales)

- **Lemus Ambrosio Aline Andrea**

En esta práctica pude aprender sobre la recursividad y como es que esta sirve en los códigos, ya que esta es de utilidad para hacer que un código sea compacto y funcional, si bien me perdí un poco en el cómo usarla, me pude apoyar con videos externos de la clase y la clase en que lo vimos, también me llegué a trabar un poco al realizar los ejercicios y cuando no encontraba mi error mis compañeros me ayudaban, logrando así resolver los ejercicios.

- **Reyes Fuentes José Manuel**

La práctica me permitió comprender más acerca de la recursividad ya que por lo visto en clase está nos permite hacer nuestro código más compacto, para poder comprender bien su funcionamiento tuve que ver el video de la clase y apoyarme un poco con otros videos, finalmente pude comprender cómo es que funciona.

- **Sánchez Alvirde Andrés Iván**

Lo que más se me complicó fue entender el concepto de recursividad y esto porque se me hacía raro el llamar a la función dentro de sí misma. Tuve que ir haciendo el paso por paso en el dev c++ para entender cómo iba trabajando el código para devolver el valor que necesitaba, también viendo videos de apoyo me ayudaron a complementar más el funcionamiento de la recursividad.

CONCLUSIONES (Individuales)

- **Lemus Ambrosio Aline Andrea**

Con esta práctica pude comprender el uso de la recursividad en los códigos, además de aprender la manera correcta en que esta se usa, algo que si me confundía y al mismo tiempo me sorprendía era ver cómo llamamos a la función dentro de sí misma, por suerte con ayuda del video de clase y videos externos, pude resolver mis dudas.

- **Reyes Fuentes José Manuel**

Al estar realizando la práctica junto con el profesor me pude dar cuenta de cómo era la utilización de la recursividad, al menos para mi fue un tanto interesante debido a como con una sola función se podría llamar así misma para estar constantemente evaluando la parte necesaria del programa, sin la necesidad de entrar con otras funciones como un while o un for, para mi al menos fue sorprendente el como iba dividiendo el problema y solucionarlo de una forma bastante favorable, en lo que se refiere en uso de líneas de código.

- **Sánchez Alvirde Andrés Iván**

Para concluir quiero decir que me pareció muy interesante la recursividad ya que ayuda a que el código sea mucho más pequeño y compacto haciendo que todo un proceso se haga en una sola función, esto también ayuda a resolver el ejercicio de una manera mucho más fácil ya que se va dividiendo por partes el problema.