	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrian Ulises Mercado

Asignatura: Estructura de Datos y Algoritmos I

Grupo: 13

No de Práctica(s): 12

Integrante(s): Narváez Campos Alejandro Tomás

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 2020-1

Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

INTRODUCCION:

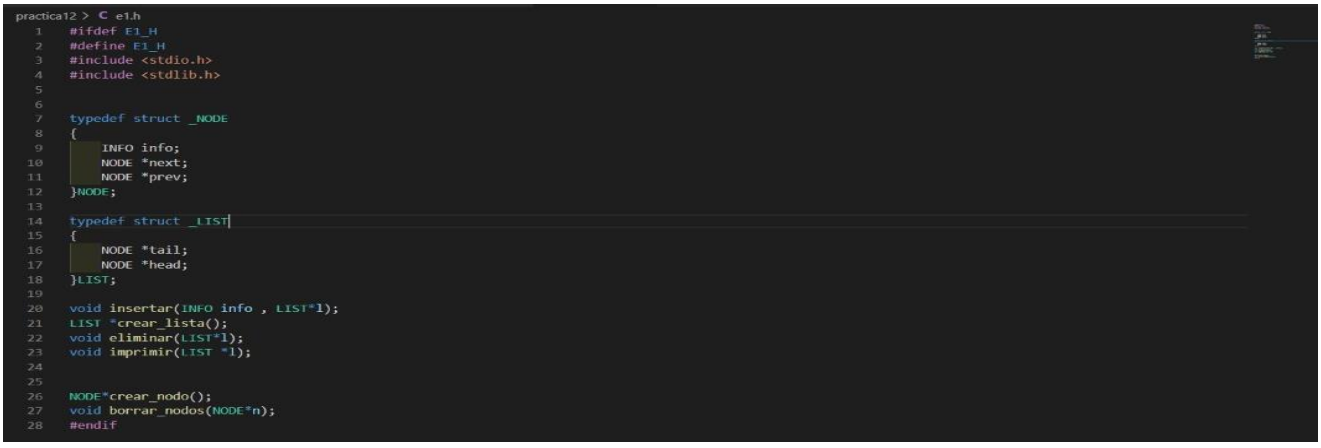
En esta practica veremos los algoritmos recursivos La recursividad es una técnica de programación que se utiliza para realizar una llamada a una función desde ella misma, de allí su nombre y de esta forma darle solucion a cierto problema, conocer esta tecnica de programacion es de gran utilidad ya que nos ayuda que ciertos algoritmos sean mas simples

Objetivo:

El objetivo de esta guía es aplicar el concepto de recursividad para la solución de problemas.

DESARROLLO

Para comprender de major manera como es el uso de la recursividad partimos de un algoritmo que ya habiamos programado anteriormente, este era un lista con ciertas funciones, esto nos ayudo ya que de esta forma podriamos ver de manera mas clara cuales eran las diferencias entre el algoritmo programado de forma “estandar” y el programado con recursividad



practica12 > C main.c

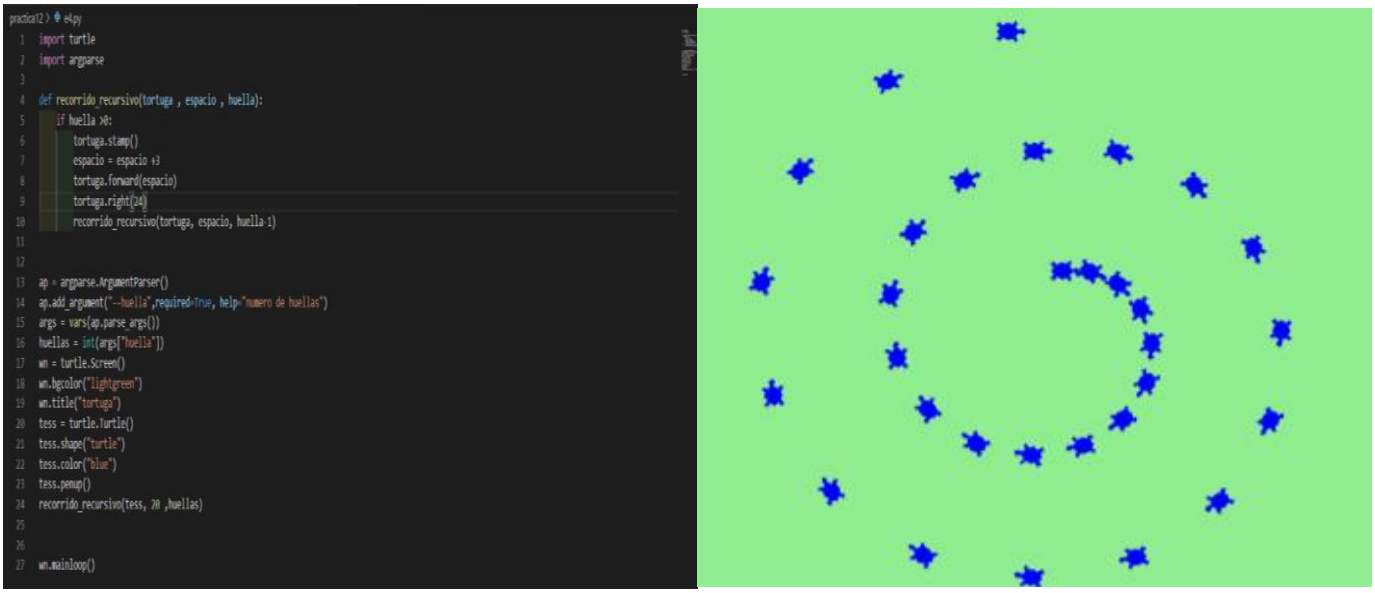
```c
#include <stdio.h>
#include <string.h>
#include "e1.h"

int  main() {
    LIST *lista;
    INFO info;
    strcpy(info.nombre, "nombre1");
    strcpy(info.apellido, "apellido 11 apellido 12");

    lista = crear_lista();
    insertar(info,lista);
    strcpy(info.nombre,"nombre2");
    strcpy(info.apellido, "apellido21 apellido 22");
    insertar(lista);
    imprimir(lista);
    strcpy(info.nombre, "nombre3");
    strcpy(info.apellido,"apellido 31 apellido 32");
    insertar(info , lista);
    imprimir(lista);
    eliminar(lista);

    return 0;
}
```

Posteriormente parte de este algoritmo lo transcribimos a lenguaje python de igual forma utilizando la recursividad, para poner en practica lo visto anteriormente en las practicas, sin embargo result un poco mas complejo el tartar de comprender por complete el codigo por eso solo hicimos una parte de las funciones

practica12 > e2.py > Lista > agregar

```python
class Nodo:
    def __init__(self):
        self.nombre = None
        self.apellidos = None
        self.next = None

    def eliminar(self):
        self = None


class Lista:
    def __init__(self):
        self.head = None
        self.tail = None

    def agregar(self, nombre, apellidos ):
        if self.head == None:
            self.head = Nodo()
            self.head.nombre = nombre
            self.head.apellidos = apellidos
            return
        temporal1 = Nodo()
        temporal1.nombre = nombre
        temporal1.apellidos = apellidos
        temporal1.next = self.head
        self.head = temporal1

    def eliminar(self):
        temp = self.head
        while self.head != None:
            while temp.next != None:
                temp = temp.next
        temp.eliminar()

```

Realizamos otro ejmplo de algoritmo de recursividad, este en especial me gusto bastante ya que mostraba las huellas de una Tortuga, mientras caminaba me parecio muy buen ejemplo pues el resultado queda muy agradable visualmente ademas de que el algoritmo con el que se program no es tan complejo como podria parecerlo



```
practica12 > e4.py
import turtle
import argparse

def recorrido_recursivo(tortuga , espacio , huella):
    if huella >0:
        tortuga.stamp()
        espacio = espacio +3
        tortuga.forward(espacio)
        tortuga.right(24)
        recorrido_recursivo(tortuga, espacio, huella-1)


ap = argparse.ArgumentParser()
ap.add_argument("--huella",required=True, help="numero de huellas")
args = vars(ap.parse_args())
huellas = int(args["huella"])
wn = turtle.Screen()
wn.bgcolor("lightgreen")
wn.title("tortuga")
tess = turtle.Turtle()
tess.shape("turtle")
tess.color("blue")
tess.penup()
recorrido_recursivo(tess, 20 ,huellas)


wn.mainloop()
```

Conclusion:

Los algoritmos recursivos son de gran utilidad para darle solucion a ciertos problemas de forma mas eficaz, igual es importante resaltar que aunque algunas veces estos algoritmos ocupan mayor cantidad de recursos, valen la pena ya que es mayor el beneficio al costo en ciertos casos, el hacer uso de la recursividad con un algoritmo bien diseñado puede lograr un algoritmo mas que eficaz