	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): Práctica 12

Integrante(s): Martinez Jacques Ricardo

No. de Equipo de cómputo empleado: N/A

No. de Lista o Brigada: Brigada 5

Semestre: 2020-2

Fecha de entrega: 7-06.2020

Observaciones:

CALIFICACIÓN: _____

INTRODUCCIÓN

Veremos como utilizar la recursividad con algunos programas. La recursividad es recursión, en ciencias de la computación, una forma de atajar y solventar problemas. De hecho, recursión es una de las ideas centrales de ciencia de computación. Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema.

DESARROLLO

Recursividad desarrollada en lenguaje C

Se desarrolla una lista doblemente ligada, donde tenemos los tipicos archivos de list.h list.c y otros. Usaremos estructuras con caracteres y nodos que sirven para adaptarlo a una lista doble ligada.

list.h

```c
#ifndef E1_H
#define E1_h

typedef struct _info{
    char nombre[32];
    char apellido[64];
}INFO;

typedef struct _node{
    INFO info;
    struct _node *next;
    struct _node *prev;
}NODE;

typedef struct list{
    NODE *tail;
    NODE *head;
} LIST;

LIST *crear_list();
void insertar(INFO info, LIST *l);
void eliminar(LIST *l);
NODE *crear_nodo();
void borrar_nodos(NODE *n);
void imprimir (LIST *l);
#endif
```

La fucion objetivo que se manipulará sera borrar_nodo ya que al momento de hacer l iteracion del if podemos volver a llamar a borrar_nodo haciendo asi la recursividad y resolviendo por partes el programa

List.c

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

LIST *crear_list(){
    LIST* l = (LIST*)malloc(sizeof(LIST));
    l->head = NULL;
    l->tail = NULL;
    return l;
}

void insertar(INFO info, LIST *l){
    if(l != NULL){
        if (l->head == NULL){
            l->head = crear_nodo();
            l->head->info = info;
            return ;
        }
        NODE *nuevo = crear_nodo();
        nuevo->info = info;
        nuevo->next = l->head;
        l->head->prev = nuevo;
        l->head = nuevo;
    }
}

void eliminar(LIST *l){
    if(l->head != NULL){
       borrar_nodos(l->head);
    }
    free(l);
}

NODE *crear_nodo(){
    NODE *n = (NODE*) malloc(sizeof(NODE));
    n->prev = NULL;
    n->next = NULL;
    return n;
}

void borrar_nodos(NODE *n){
    if(n->next != NULL){
        borrar_nodos(n->next);
    }
    n->prev = NULL;//Caso base
    free(n);
}

void imprimir (LIST *l){
    for (NODE *i = l->head; i != NULL; i = i->next ){
        printf("%s, %s\n", i->info.nombre, i->info.apellido);
    }
```

```
}
```

Para concluir haremos el programa principal main donde se solicitara datos como los nombres y apellidos y se usaran funciones como lo es strcpy para copiar caracteres a una variable o un array

Main.c

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


int main(){
    LIST *lista;
    INFO info[3];


    strcpy(info[0].nombre, "Darrion");
    strcpy(info[0].apellido, "Rohan Hagenes");
    strcpy(info[1].nombre, "Elian");
    strcpy(info[1].apellido, "Brenna Langworth");
    strcpy(info[2].nombre, "Shania");
    strcpy(info[2].apellido, "Carmella Gaylord");
    lista = crear_list();
    insertar(info[0], lista);
    insertar(info[1], lista);
    insertar(info[2], lista);
    imprimir(lista);
    eliminar(lista);
    return 0;
}
```

Recursividad en python

Usaremos dos bibliotecas de python, la primera turtle para hacer el camino de la tortuga y la otra argparse para mandar datos por banderas.

Usaremos las funcoines def y desarrollamos las partes del programa de la tortuga usando forward, right, stamo y otras más que sirven para hacer las huellas de la tortuga, la direccion izq, o derecha y otros movimientos.

Como vamos a ingresar el número de huellas que realice la tortuga por líneas de comando, se utiliza la biblioteca argparse, la cual se va a encargar de transformar el string ”huellas” en un entero.

Para ello, se inicializa la variable ap con la función argparse.ArgumentParser().

Después se agrega el dato de entrada con la bandera --huellas, también establecemos que se debe ingresar un número, con requiered y un aviso que escriba el “número de huellas“ por si no se sabe que se debe escribir ap.add_argument(“--huellas”, requiered = true, help = “número de huellas”)

Una vez que se tiene el dato de entrada, se convierte en un valor, ya que ap.argparse sigue siendo un string.

args = vars(ap.argparse( ))

Finalmente se iguala el número de huellas que va a realizar la tortuga con el valor de entrada:

huellas = int(args[“huellas”].

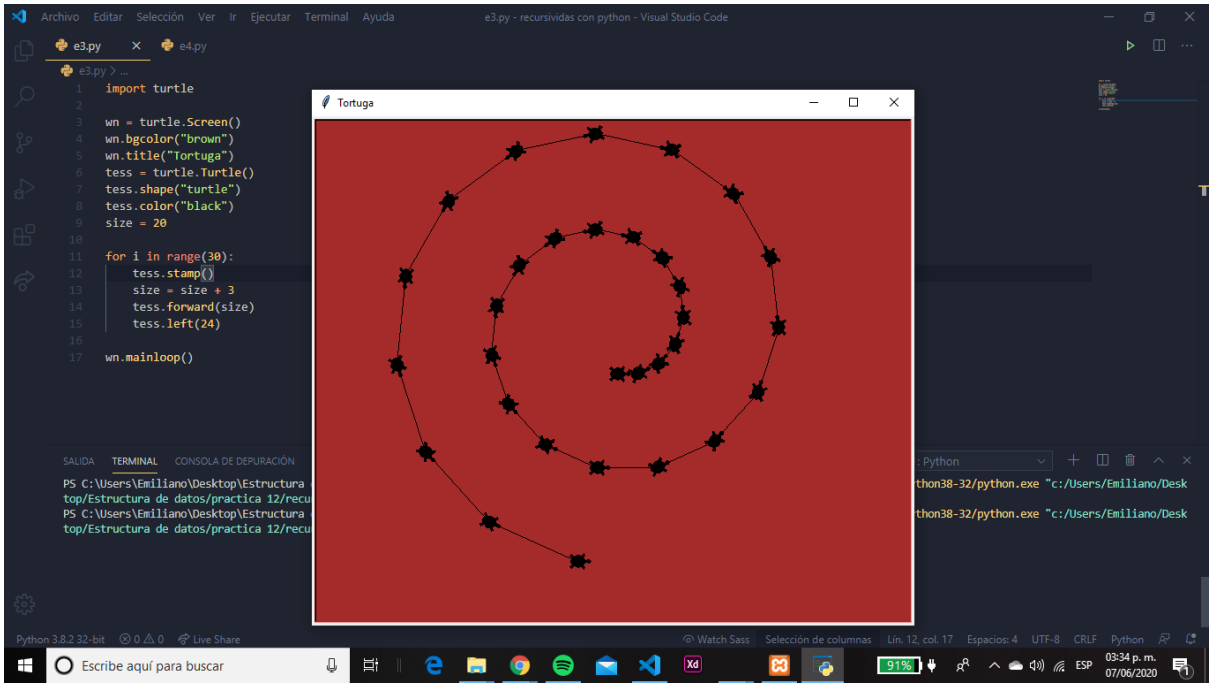
```
import turtle
import argparse

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


ap = argparse.ArgumentParser()
ap.add_argument("--huellas", required=True, help = "numero de huellas")
args = vars(ap.parse_args())
huellas = int(args["huellas"])

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()
```

EJECUCIÓN

Ejecución en c

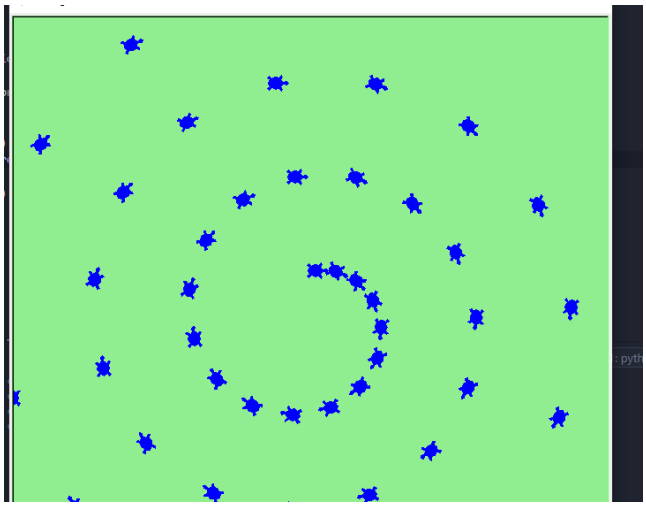
```
Shania, Carmella Gaylord
Elian, Brenna Langworth
Darrion, Rohan Hagenes
```

Ejecución en python no recursivo- **E3**



***E4**-SI ES RECURSIVO*

Para ejecutar el ejercicio recursivo, es necesario que en la terminal se escriba el nombre del archivo, seguido de --huellas y el número de pasos que quieres que realice la tortuga.
En este caso la tortuga realizó 45 pasos



CONCLUSIÓN

Es muy importante conocer las funciones recursivas y sus utilizaciones, para así ahorrarnos código, pero requerimos de un caso base para llevarlo a cabo.

COMENTARIOS

La bilbioteca de turtle me parece muy interactiva y me gusto el hecho de ver movimiento y no solo código a secas.

BIBLIOGRAFÍAS

Tutorial oficial de Python: https://docs.python.org/3/tutorial/
Galería de notebooks: https://wakari.io/gallery
Matplotlib: http://matplotlib.org/