2do examen parcial

Axel Daniel, Ramirez Urbina, 202010083
Escuela de Mecánica Eléctrica, Facultad de Ingeniería,
Universidad de San Carlos de Guatemala

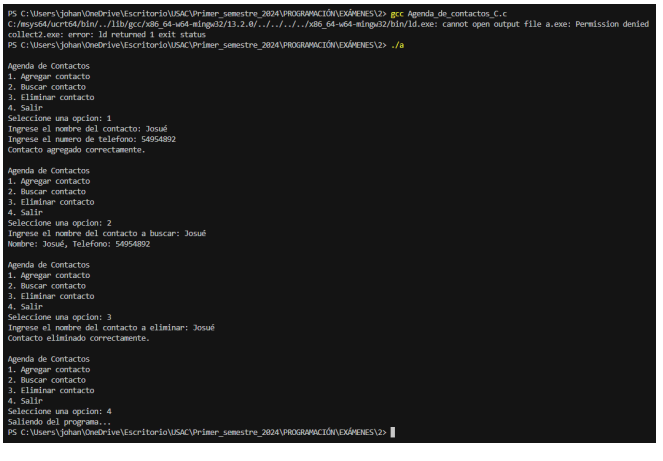
A. *Resumen*

El programa "Agenda de contactos" en lenguaje C es una aplicación diseñada para permitir a los usuarios gestionar una lista de contactos. Ofrece una interfaz gráfica de usuario utilizando la biblioteca Tkinter, donde los usuarios pueden realizar diversas acciones, como agregar nuevos contactos, editar los existentes, eliminar contactos y buscarlos por nombre o número de teléfono. La información de los contactos se guarda en un archivo de texto, lo que proporciona una persistencia de los datos incluso después de cerrar el programa. El programa gestiona las operaciones de lectura y escritura en el archivo de texto para garantizar que los usuarios puedan acceder y mantener su lista de contactos de manera eficiente.

B. *Código*

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CONTACTOS 100
#define MAX_NOMBRE 50
#define MAX_TELEFONO 20

// Definición de la estructura de un contacto
struct Contacto {
    char nombre[MAX_NOMBRE];
    char telefono[MAX_TELEFONO];
};

// Función para agregar un contacto
void agregarContacto(struct Contacto agenda[], int *numContactos) {
    if (*numContactos >= MAX_CONTACTOS) {
        printf("La agenda esta llena. No se pueden agregar mas contactos.\n");
        return;
    }

    printf("Ingrese el nombre del contacto: ");
    scanf("%s", agenda[*numContactos].nombre);
    printf("Ingrese el número de telefono: ");
    scanf("%s", agenda[*numContactos].telefono);

    (*numContactos)++;
    printf("Contacto agregado correctamente.\n");
}

// Función para buscar un contacto por nombre
void buscarContacto(struct Contacto agenda[], int numContactos, char nombre[]) {
    int encontrado = 0;
    for (int i = 0; i < numContactos; i++) {
        if (strcmp(agenda[i].nombre, nombre) == 0) {
            printf("Nombre: %s, Telefono: %s\n", agenda[i].nombre, agenda[i].telefono);
            encontrado = 1;
        }
    }
    if (!encontrado) {
        printf("El contacto no se encuentra en la agenda.\n");
    }
}

// Función para eliminar un contacto por nombre
void eliminarContacto(struct Contacto agenda[], int *numContactos, char nombre[]) {
    int encontrado = 0;
    for (int i = 0; i < *numContactos; i++) {
        if (strcmp(agenda[i].nombre, nombre) == 0) {
            for (int j = i; j < *numContactos - 1; j++) {
                strcpy(agenda[j].nombre, agenda[j + 1].nombre);
                strcpy(agenda[j].telefono, agenda[j + 1].telefono);
            }
            (*numContactos)--;
            encontrado = 1;
            printf("Contacto eliminado correctamente.\n");
        }
    }
    if (!encontrado) {
        printf("El contacto no se encuentra en la agenda.\n");
    }
}

// Función para editar un contacto por nombre
void editarContacto(struct Contacto agenda[], int numContactos, char nombre[]) {
    int encontrado = 0;
    for (int i = 0; i < numContactos; i++) {
        if (strcmp(agenda[i].nombre, nombre) == 0) {
            printf("Ingrese el nuevo nombre del contacto: ");
            scanf("%s", agenda[i].nombre);
            printf("Ingrese el nuevo número de telefono: ");
            scanf("%s", agenda[i].telefono);
            printf("Contacto editado correctamente.\n");
            encontrado = 1;
        }
    }
    if (!encontrado) {
        printf("El contacto no se encuentra en la agenda.\n");
    }
}

// Función principal
int main() {
    struct Contacto agenda[MAX_CONTACTOS];
    int numContactos = 0;
    int opcion;
    char nombre[MAX_NOMBRE];

    while (1) {
        printf("\nAgenda de Contactos\n");
        printf("1. Agregar contacto\n");
        printf("2. Buscar contacto\n");
        printf("3. Eliminar contacto\n");
        printf("4. Editar contacto\n");
        printf("5. Salir\n");
        printf("Seleccione una opcion: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                agregarContacto(agenda, &numContactos);
                break;
            case 2:
                printf("Ingrese el nombre del contacto a buscar: ");
                scanf("%s", nombre);
                buscarContacto(agenda, numContactos, nombre);
                break;
            case 3:
                printf("Ingrese el nombre del contacto a eliminar: ");
                scanf("%s", nombre);
                eliminarContacto(agenda, &numContactos, nombre);
                break;
            case 4:
                printf("Ingrese el nombre del contacto a editar: ");
                scanf("%s", nombre);
                editarContacto(agenda, numContactos, nombre);
                break;
            case 5:
                printf("Saliendo del programa...\n");
                exit(0);
            default:
                printf("Opción no valida. Por favor, seleccione una opción valida.\n");
        }
    }

    return 0;
}
```

C. *Resultados*



Documento en GitHub
https://github.com/AxelRamirez12/parcial2.git

D. *Resumen*

El programa "Agenda de contactos" en Python es una herramienta diseñada para ayudar a los usuarios a administrar sus contactos de manera eficiente. Este programa ofrece una interfaz gráfica de usuario utilizando la biblioteca Tkinter, que permite a los usuarios realizar diversas acciones, como agregar nuevos contactos, editar los existentes, eliminar contactos y buscarlos por nombre o número de teléfono. La información de los contactos se guarda en un archivo de texto, lo que asegura que los datos persistan incluso después de cerrar el programa. Además, el programa gestiona las operaciones de lectura y escritura en el archivo de texto para garantizar un acceso rápido y una actualización efectiva de la lista de contactos.

E. *Código*

```python
from tkinter import *

ventana = Tk()
ventana.title("Calculadora")

i = 0

#Entrada
e_texto = Entry(ventana, font= ("Calibri 20"))
e_texto.grid(row = 0, column = 0, columnspan = 4, padx = 5, pady = 5)

#Funciones
def click_boton(valor):
    global i
    e_texto.inser

def borrar():
    e_texto.delete(0, END)
    i = 0

def hacer_operacion():
    ecuacion = e_texto.get()
    resultado = eval(ecuacion)
    e_texto.delete(0, END)
    e_texto.insert(0,resultado)
    i = 0


#Botones
boton1 = Button(ventana, text = "1", width = 5, height = 2, command = lambda: click_boton(1))
boton2 = Button(ventana, text = "2", width = 5, height = 2, command = lambda: click_boton(2))
boton3 = Button(ventana, text = "3", width = 5, height = 2, command = lambda: click_boton(3))
boton4 = Button(ventana, text = "4", width = 5, height = 2, command = lambda: click_boton(4))
boton5 = Button(ventana, text = "5", width = 5, height = 2, command = lambda: click_boton(5))
boton6 = Button(ventana, text = "6", width = 5, height = 2, command = lambda: click_boton(6))
boton7 = Button(ventana, text = "7", width = 5, height = 2, command = lambda: click_boton(7))
boton8 = Button(ventana, text = "8", width = 5, height = 2, command = lambda: click_boton(8))
boton9 = Button(ventana, text = "9", width = 5, height = 2, command = lambda: click_boton(9))
boton0 = Button(ventana, text = "0", width = 13, height = 2, command = lambda: click_boton(0))

boton_borrar = Button(ventana, text = "AC", width = 5, height = 2,command = lambda: borrar())
boton_parentesis1 = Button(ventana, text = "(", width = 5, height = 2,command = lambda: click_boton("("))
boton_parentesis2 = Button(ventana, text = ")", width = 5, height = 2,command = lambda: click_boton(")"))
boton_punto = Button(ventana, text = ".", width = 5, height = 2,command = lambda: click_boton("."))

boton_div = Button(ventana, text = "/", width = 5, height = 2,command = lambda: click_boton("/"))
boton_mult = Button(ventana, text = "x", width = 5, height = 2,command = lambda: click_boton("*"))
boton_sum = Button(ventana, text = "+", width = 5, height = 2,command = lambda: click_boton("+"))
boton_rest = Button(ventana, text = "-", width = 5, height = 2,command = lambda: click_boton("-"))
boton_igual = Button(ventana, text = "=", width = 5, height = 2,command = lambda: hacer_operacion("="))

#Agregar botones en pantalla.
boton_borrar.grid(row = 1, column = 0, padx = 5, pady = 5)
boton_parentesis1.grid(row = 1, column = 1, padx = 5, pady = 5)
```

F. *Resultados*

```
PS C:\Users\Johan\OneDrive\Escritorio\USAC\Primer_semestre_2024\PROGRAMACIÓN\EXÁMENES\2> & C:/Users/Johan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Johan/OneDrive/Escritorio/USAC/Primer_semestre_2024/PROGRAMACIÓN/EXÁMENES/2/Agenda_de_contactos_Python.py
PS C:\Users\Johan\OneDrive\Escritorio\USAC\Primer_semestre_2024\PROGRAMACIÓN\EXÁMENES\2> & C:/Users/Johan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Johan/OneDrive/Escritorio/USAC/Primer_semestre_2024/PROGRAMACIÓN/EXÁMENES/2/Agenda_de_contactos_Python.py
PS C:\Users\Johan\OneDrive\Escritorio\USAC\Primer_semestre_2024\PROGRAMACIÓN\EXÁMENES\2>
```

Documento en GitHub
https://github.com/AxelRamirez12/parcial2.git