UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN

MANUAL TÉCNICO

PRACTICA 1

Allan Josué Rafael Morales 201709196

FEBRERO/2021

REQUISITOS MINIMOS DEL SISTEMA

- SO: Windows Vista/7/8/8.1/10.
- Procesador: 2.0 GHz o mejor.
- Memoria: 1 GB de RAM.
- Gráficos: 1280×720 de resolución mínima, 512 MB video RAM recomendada.
- DirectX: Versión 11.
- Almacenamiento: 250 MB de espacio disponible.

APLICACIÓN PROGRAMADA EN PYTHON

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

IDE UTILIZADA

PyCharm proporciona una finalización del código inteligente, inspecciones del código, indicación de errores sobre la marcha y arreglos rápidos, así como refactorización de código automática y completas funcionalidades de navegación.

LIBRERIAS IMPORTADAS

```
from tkinter import *
from tkinter.filedialog import askopenfilename
```

LÓGICA DEL PROGRAMA

La creación del menú es muy simple, con la utilización de la función while, tomando un dato de entrada para la selección de la opción dentro de la aplicación, para luego llamar un método creado anteriormente que realice la operación.

```
if __name__ == '__main__':
    Opcion = None
    while Opcion != 6:
        print('--------------------------------')
        print('********PRACTICA N.1********')
        print('1. Cargar archivos de entrada.')
        print('2. Desplegar listas ordenadas.')
        print('3. Desplegar búsquedas.')
        print('4. Desplegar todas.')
        print('5. Desplegar todas a archivo.')
        print('6. Salir.')
        print('--------------------------------')
        Opcion = int(input())
        if Opcion == 1:
            print("--------------------------------")
            print("Cargar Archivo:")
            root = abrirDoc()
            leerDoc(root)
            print(root)
            print()
            Imprimir(cm)
            cm = QuitarLineas(cm)
        elif Opcion == 2:
            print("--------------------------------")
            Ordenados()
        elif Opcion == 3:
            print("--------------------------------")
            Buscados()
        elif Opcion == 4:
            print("--------------------------------")
            Imprimirlo()
        elif Opcion == 5:
            print("--------------------------------")
            HTML()
        elif Opcion == 6:
            print("----------FINALIZADO-----------")
        else:
            print("Escoga una opción valida")
```

Al presentarse el menú, lo que sigue es abrir un filechoose, en la que el usuario seleccionará el archivo con los datos que harán funcionar las demás opciones. Aquí es cuando utilizamos las librerías importadas:

```
def abrirDoc():
    root = Tk()
    name = askopenfilename()
    root.withdraw()
    return name
```

Al instante de abrir dicho archivo, se procederá a leer el contenido con el método LeerDoc():

```
def leerDoc(doc):
    with open(doc) as fichero:
        for lineas in fichero.readlines():
            cm.append(lineas)
        pass
```

Se crea con el parámetro doc para señalar que archivo se leerá.

DESPUES DE LEER EL ARCHIVO…

Se toman los dígitos por medio de la función QuitarComa.

```
def QuitarComa(lista):
    palabra = []
    for a in lista.split(","):
        palabra.append(a)
    return palabra
```

Se toma cada fila individualmente para buscar la operación deseada que se quiere realizar sobre la lista, esto se realiza con la función QuitarLineas.

```
def QuitarLineas(vec):
    b = []
    for fil in vec:
        b.append(fil.strip("\n").split("="))
    for fila in b:
        elemento = fila[1]
        orden = fila[1].find("ORDENAR")
        busca = fila[1].find("BUSCAR")
        if busca != -1 & orden == -1:
            fin = fila[1].find("BUSCAR")
            fila.append("BUSCAR")
            numero = ''.join(elemento[:fin - 1].strip().split())
            buscado = ''.join(elemento[fin + 6:].strip().split())
            fila[1] = numero
            fila.append(buscado)
```

Para ordenarlos:

```
def Ordenados():
    print("******Listas Ordenadas******")
    for linea in cm:
        for elemento in linea:
            if elemento == "ORDENAR":
                nombreLinea = linea[0]
                numero = QuitarComa(linea[1])
                num = ','.join(sorted(numero))
                print(nombreLinea + ": ORDENADOS = " + num, end="\n")
```

Se utiliza la función Ordenados, esta se mandará a llamar luego de haber utilizado la función de QuitarLineas e identificar qué línea tiene la lista que tiene que ordenar sus números.

Para buscarlos:

```
def Buscados():
    print("*******Listas Buscadas******")
    for linea in cm:
        for elemento in linea:
            if elemento == "BUSCAR":
                nombre = linea[0]
                numero = QuitarComa(linea[1])
                buscado = linea[len(linea) - 1]
                num = ','.join(numero)
                posiciones = Busqueda(numero, buscado)
                if not Vacio(posiciones):
                    pos = ','.join(map(str, posiciones))
                    print(nombre + ": = " + num + " EL NUMERO " + buscado + " SE ENCUENTRA EN LA POSICION = " + pos,end="\n")
                else:
                    print(nombre + ": = " + num + " EL NUMERO " + buscado + " NO SE HA ENCONTRADO ", end="\n")
```

Se utiliza la función Buscados, esta se encarga de tomar las listas en las que se buscaran los números seleccionados.

ARCHIVO HTML

```
def HTML():
    lol = open('Datos en HTML.html', 'w')
    start = """<html>
```

Para abrir un archivo HTML se utiliza la función “open”, esto creará un archivo que designaremos esta vez como ‘Datos en HTML.html’, seguido de ‘w’ para definir que escribiremos dentro de este. A continuación, se escribirá el código HTML para presentar los datos operados en un formato distinto.

PARADIGMAS

De los paradigmas utilizados para realizar la aplicación esta principalmente el paradigma orientado a objetos, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones. La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.