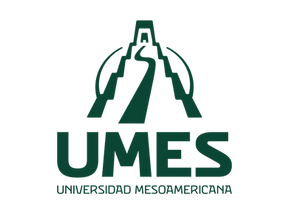
UNIVERSIDAD MESOAMERICANA

QUETZALTENANGO

FACULTAD DE INGENIERIA



PROYECTO

**Programa de Creación, Operación y Graficación**

**De conjuntos.**

Wendy Yamileth López Quijivix

Carnet No. 202508081

Emily Dayana Del Carmen Cabrera

Carnet No. 202508083

Miguel Abraham Mendoza López

Carnet No. 202508086

QUETZALTENANGO, MAYO 2025

**ÍNDICE**

1. Introducción 3

1. Objetivos del Proyecto 3
2. Descripción General del Sistema 3
3. Tecnologías Utilizadas 3

1. Estructura del Proyecto 4
2. Comunicación entre Python y C++ 4
3. Flujo de Ejecución 4
4. Manual de Usuario 4/5
5. Casos de Prueba 5
6. Resultados y Capturas de Pantalla 5

1. Conclusiones y Recomendaciones 10
2. Glosario de Términos 10
3. Código Fuente (Anexo) 11
4. **Introducción**

Este documento describe el desarrollo de un software educativo interactivo que permite la creación, operación y visualización de conjuntos utilizando diagramas de Venn. El sistema combina Python (para la interfaz gráfica y operaciones matemáticas) con C++ (para validación y carga eficiente de conjuntos). Esta combinación ofrece modularidad, rendimiento y una experiencia de usuario intuitiva.

1. **Objetivos del Proyecto**

Objetivo General:

* Desarrollar una aplicación gráfica que permita manipular conjuntos mediante operaciones matemáticas básicas, con soporte visual y validación robusta.

Objetivos Específicos:

* Integrar Python y C++ para lograr una arquitectura modular.
* Permitir operaciones como unión, intersección, diferencia y complemento.
* Visualizar resultados con diagramas de Venn.
* Validar datos de entrada y manejar errores comunes.

1. **Descripción General del Sistema**

El sistema consta de dos módulos los cuales son:

* Módulo C++: Se ejecuta mediante consola y permite ingresar, validar y guardar conjuntos en un archivo conjuntos.txt.
* Módulo Python: Proporciona una interfaz gráfica intuitiva con botones, etiquetas, listas y campos de texto. Lee el archivo generado por el módulo C++ y permite realizar operaciones visuales.

1. **Tecnologías Utilizadas**

* Lenguajes: Python 3.10, C++
* Librerías Python: Tkinter, Matplotlib, matplotlib-venn, Pillow
* Herramientas: Editor de texto (VSCode), compilador g++ / MinGW

1. **Estructura del Proyecto**

* codigofinal.py: Es todo el código ya terminando el cual contiene la interfaz, operaciones y lógica del sistema en Python.
* conjuntos.cpp: Programa en C++ para capturar y procesar los conjuntos.
* conjuntos.exe: Es el programa en C++ ya compilado.
* compilación y ejecucion: muestra los comandos base para poder compilar y ejecutar el programa en C++
* conjuntos.txt: Archivo temporal con los conjuntos generados.
* README: Se indican instrucciones importantes para conseguir que el programa funcione adecuadamente.
* requirements.txt: Contiene todas las librerías que fueron utilizadas para la elaboración del proyecto así mismo permite su instalación de forma automática a través del comando “pip install -r requirements.txt”

1. **Comunicación entre Python y C++**

Python llama al ejecutable C++ (conjuntos.exe) mediante el módulo subprocess. C++ captura los conjuntos desde la consola, los valida y escribe en conjuntos.txt. Luego, Python lee este archivo y extrae los datos para procesarlos o visualizarlos.

1. **Flujo de Ejecución**
2. El usuario ejecuta codigofinal.py.
3. Se muestra una pantalla de presentación.
4. El usuario presiona "Crear nuevo conjunto".
5. Se abre la consola del programa en C++.
6. Tras ingresar los conjuntos, se guardan en conjuntos.txt.
7. Python los carga y se muestran en pantalla.
8. El usuario puede aplicar operaciones y visualizar resultados a través del diagrama de venn.
9. **Manual de Usuario**

Requisitos Previos:

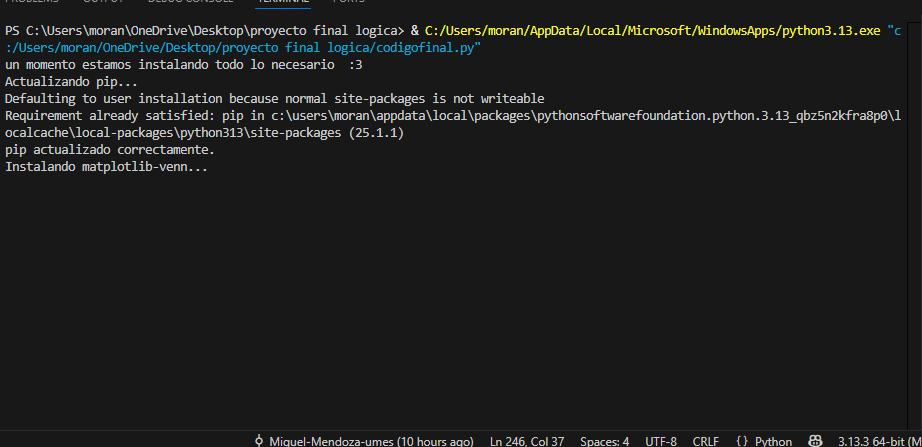
* Python 3.10+
* pip install -r requirements.txt

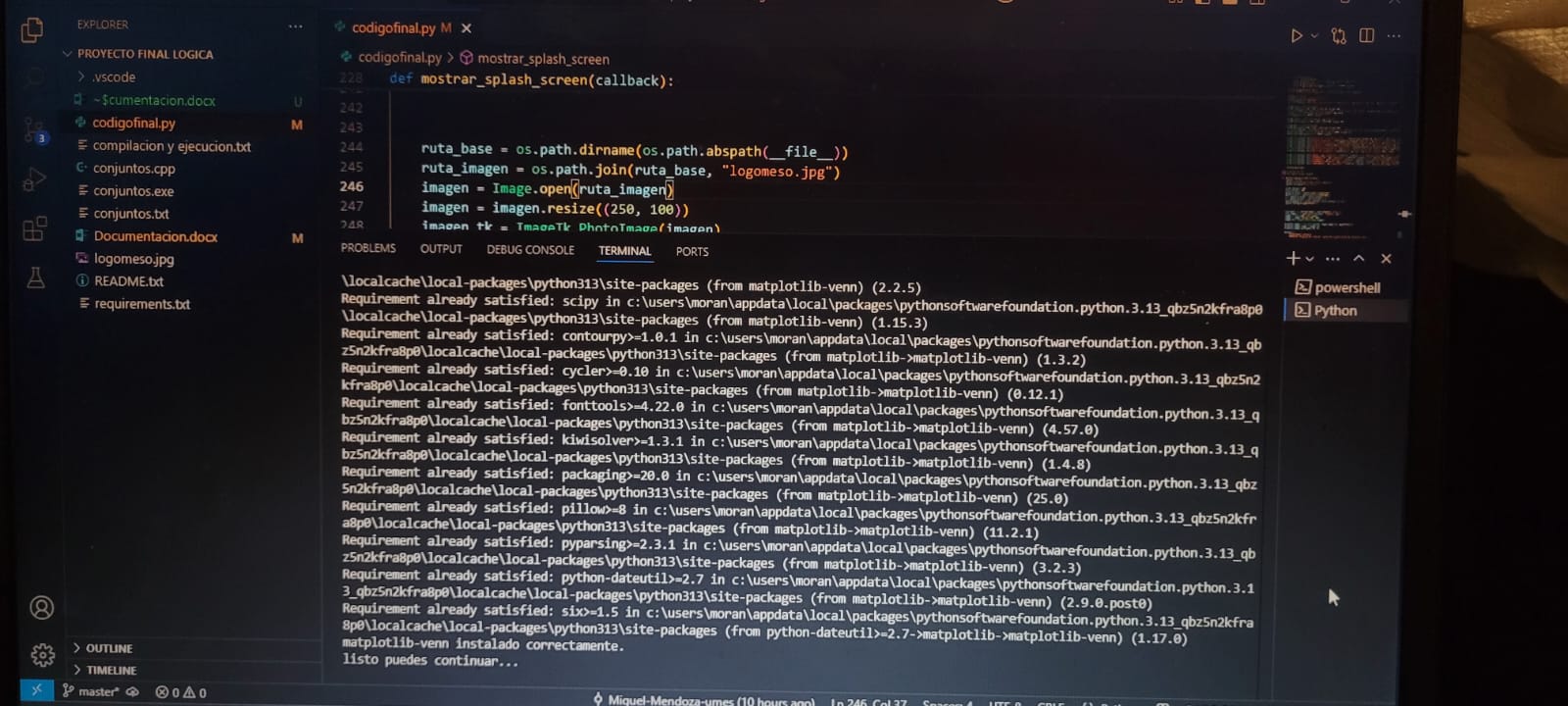
Pasos para uso:

1. Ejecutar codigofinal.py
2. Crear conjuntos usando la opción correspondiente.
3. Cargar conjuntos en la interfaz.
4. Seleccionar operaciones.
5. Visualizar el diagrama de Venn
6. **Casos de Prueba**

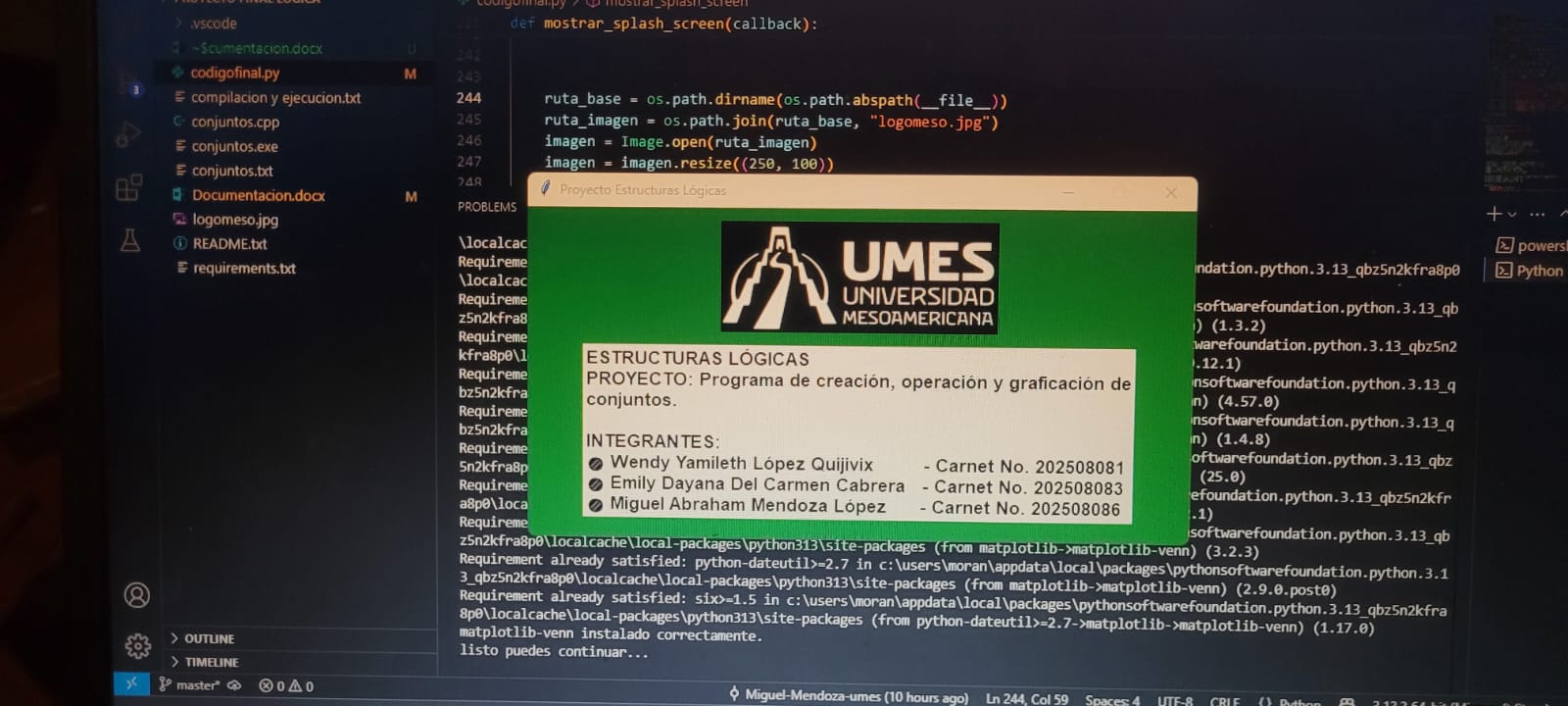
|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 1 | A={1,2}, B={2,3} | Unión | {1,2,3} |
| 2 | A={1,2,3}, B={2} | Diferencia | {1,3} |
| 3 | A={5,6}, B={7,8} | Intersección | {} |
| 4 | A={a,b}, B={b,c} | Diferencia Simétrica | {a,c} |
| 5 | A={1,2}, U={1,2,3,4} | Complemento | {3,4} |
|  |  |  |  |

1. **Resultados y Capturas de Pantalla**
   1. Instalación, Actualización de librerías y PIP y Abriendo el programa





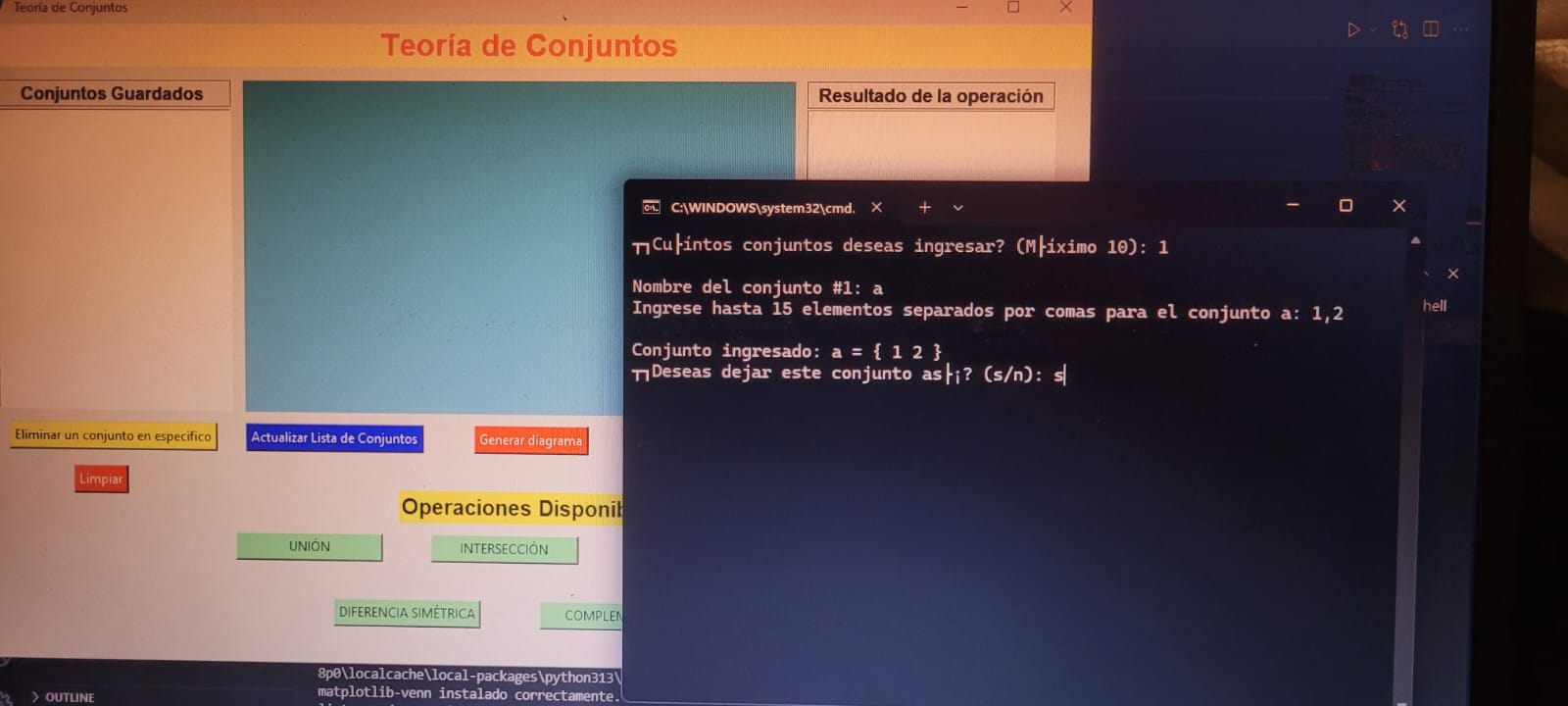
* 1. Ventana de presentación del grupo



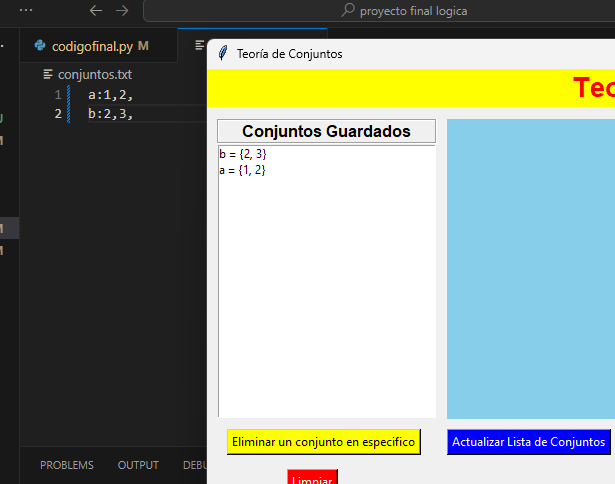
* 1. Programa principal



* 1. Programa en C++ creación de conjuntos

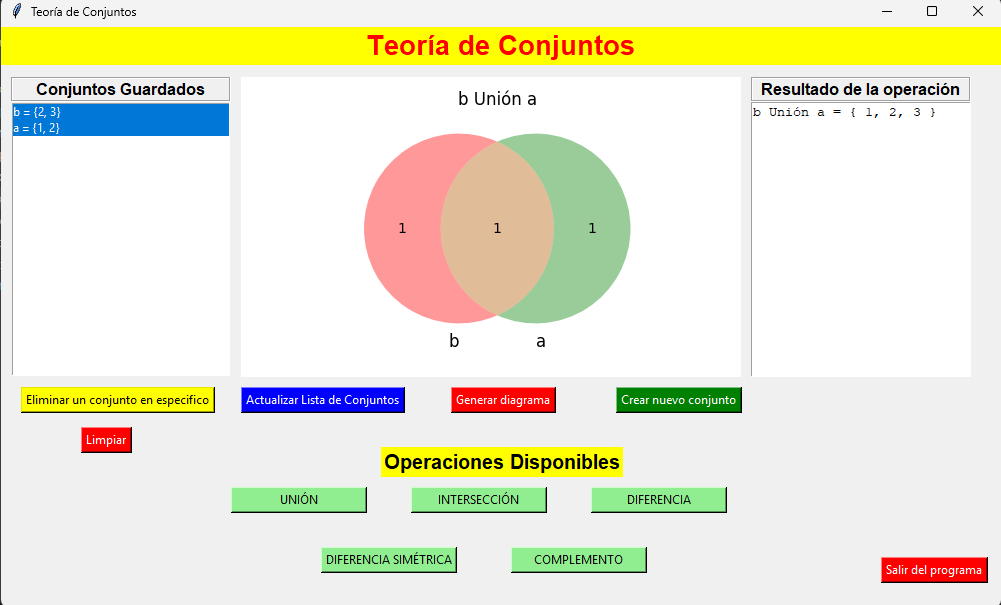


* 1. Conjuntos añadidos al programa principal



* 1. Primer caso de prueba:

|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 1 | A={1,2}, B={2,3} | Unión | {1,2,3} |

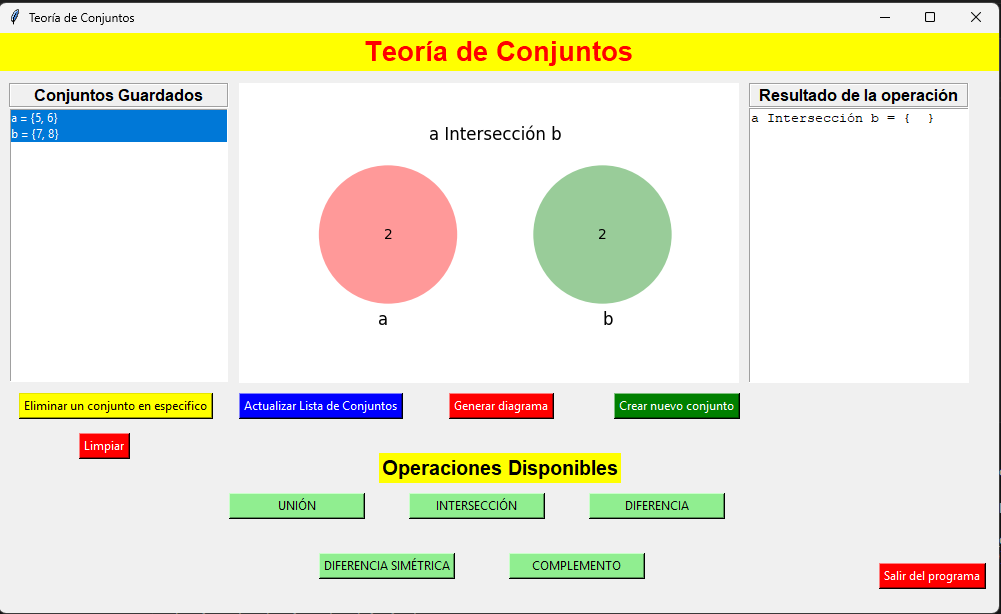


* 1. Segundo caso de prueba:

|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 2 | A={1,2,3}, B={2} | Diferencia | {1,3} |
|  |  |  |  |
|  |  |  |  |

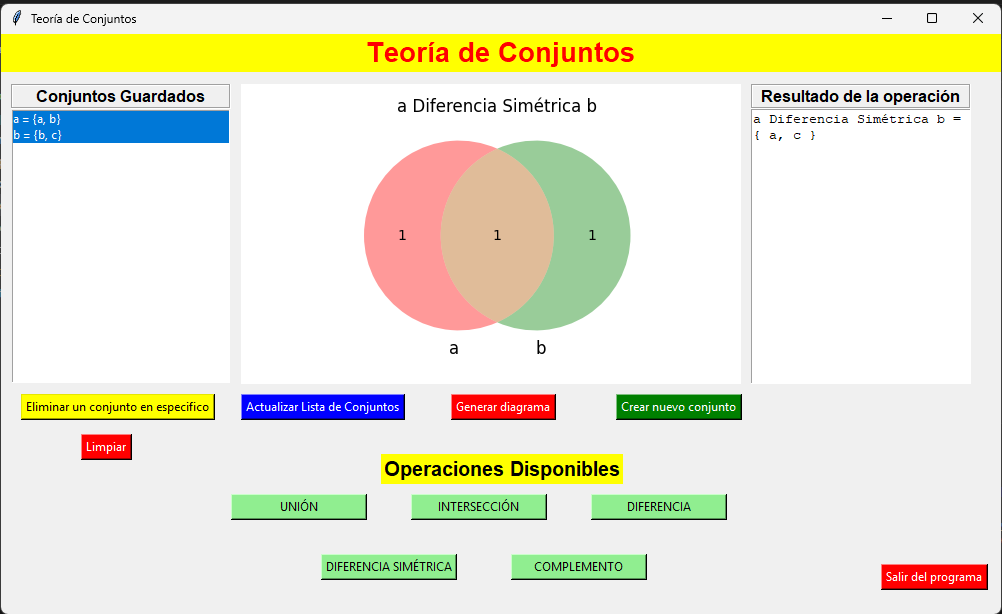
* 1. Tercer caso de prueba:

|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 3 | A={5,6}, B={7,8} | Intersección | {} |



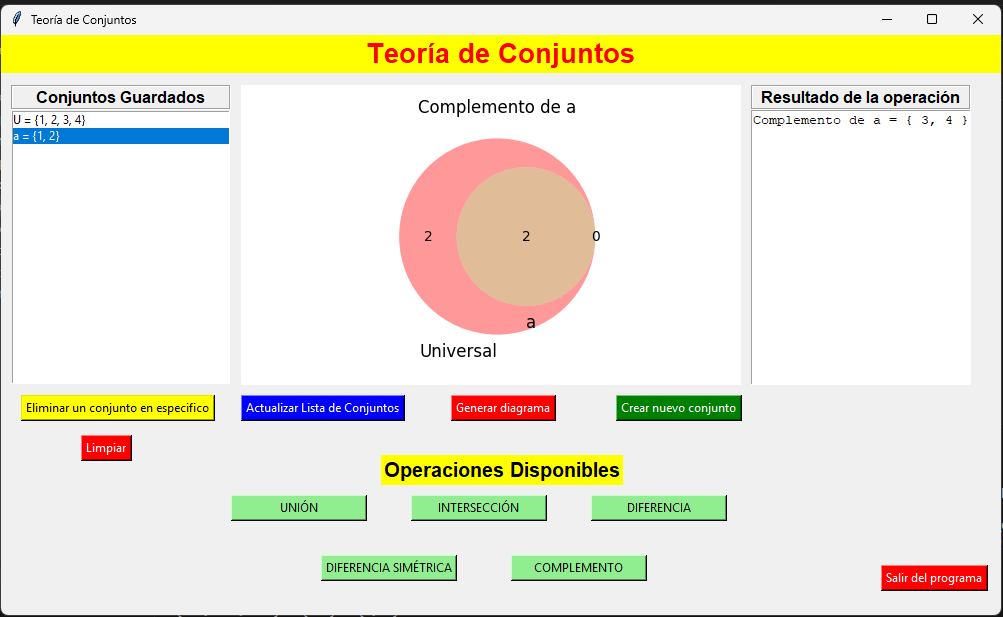
|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 4 | A={a,b}, B={b,c} | Diferencia Simétrica | {a,c} |

* 1. Cuarto caso de prueba:

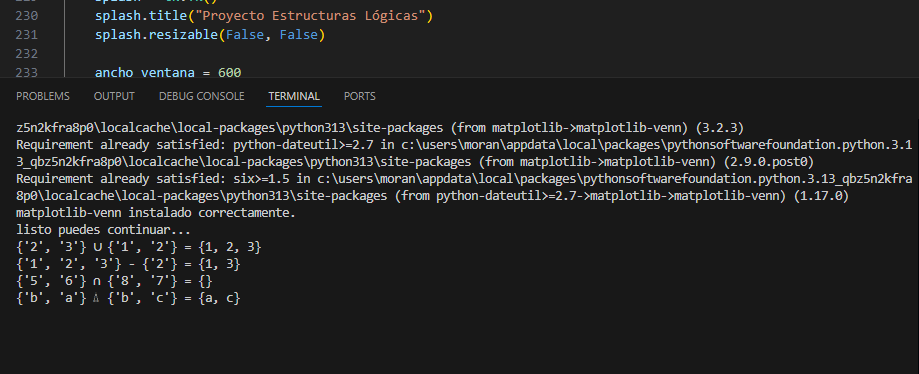


* 1. Quinto caso de prueba:

|  |  |  |  |
| --- | --- | --- | --- |
| Caso | Entrada | Acción | Resultado Esperado |
| 5 | A={1,2}, U={1,2,3,4} | Complemento | {3,4} |



* 1. Resumen de resultados obtenidos



1. **Conclusiones y Recomendaciones**

* El proyecto evidencia cómo integrar lenguajes de manera eficiente.
* La separación de responsabilidades mejora la mantenibilidad.
* Se recomienda no sobre cargar el programa puesto que podría llegar a congelarse.

1. **Glosario de Términos**

* Tkinter: Librería de Python para crear interfaces gráficas de usuario.
* Set (C++): Contenedor que almacena elementos únicos sin orden específico.
* Subprocess: Módulo en Python que permite ejecutar procesos externos (como

ejecutables).

* Diagrama de Venn: Representación visual de relaciones entre conjuntos.
* - Pack(): Método de Tkinter que posiciona elementos en la interfaz.
* matplotlib\_venn: Módulo de Python para generar diagramas de Venn.

1. **Código Fuente (Anexo)**

* **Código de c++:**

#include <iostream>     // Para entrada/salida estándar

#include <set>          // Para usar conjuntos (set)

#include <string>       // Para manipular cadenas de texto

#include <fstream>      // Para manejar archivos

#include <map>          // Para almacenar múltiples conjuntos con nombres

#include <algorithm>    // Para funciones como remove\_if

using namespace std;

// Función que verifica si un nombre de conjunto ya ha sido usado

bool nombre\_usado(const map<string, set<string>>& conjuntos, const string& nombre) {

    return conjuntos.find(nombre) != conjuntos.end();

}

int main() {

    const int MAX\_CONJUNTOS = 10;     // Número máximo de conjuntos permitidos

    const int MAX\_ELEMENTOS = 15;    // Número máximo de elementos por conjunto

    map<string, set<string>> conjuntos; // Mapa para almacenar conjuntos con su nombre

    int numConjuntos;

    // Solicita al usuario cuántos conjuntos desea ingresar

    while (true) {

        cout << "¿Cuántos conjuntos deseas ingresar? (Máximo " << MAX\_CONJUNTOS << "): ";

        cin >> numConjuntos;

        // Verifica que la entrada sea válida

        if (cin.fail() || numConjuntos < 1 || numConjuntos > MAX\_CONJUNTOS) {

            cin.clear();             // Limpia el estado de error

            cin.ignore(1000, '\n');  // Descarta entrada incorrecta

            cout << "Entrada inválida. Ingresa un número entre 1 y " << MAX\_CONJUNTOS << "." << endl;

        } else {

            cin.ignore();            // Limpia el buffer de entrada después del número

            break;

        }

    }

    // Funcion para ingresar cada conjunto

    for (int i = 0; i < numConjuntos; ++i) {

        string nombre;

        while (true) {

            cout << "\nNombre del conjunto #" << (i + 1) << ": ";

            getline(cin, nombre);

            // Verifica si el nombre ya fue usado

            if (nombre\_usado(conjuntos, nombre)) {

                cout << "Este nombre ya fue utilizado. Usa un nombre único." << endl;

                continue;

            }

            cout << "Ingrese hasta " << MAX\_ELEMENTOS << " elementos separados por comas para el conjunto " << nombre << ": ";

            string linea;

            getline(cin, linea);

            set<string> elementos;

            size\_t start = 0, end;

            // Procesa la línea separando los elementos por comas

            while ((end = linea.find(',', start)) != string::npos) {

                string token = linea.substr(start, end - start);

                token.erase(remove\_if(token.begin(), token.end(), ::isspace), token.end());

                if (!token.empty()) elementos.insert(token);

                start = end + 1;

            }

            // Agrega el último elemento

            string ultimo = linea.substr(start);

            ultimo.erase(remove\_if(ultimo.begin(), ultimo.end(), ::isspace), ultimo.end());

            if (!ultimo.empty()) elementos.insert(ultimo);

            // Verifica que no exceda el límite de elementos

            if (elementos.size() > MAX\_ELEMENTOS) {

                cout << "Has superado el límite de " << MAX\_ELEMENTOS << " elementos. Intenta de nuevo." << endl;

                continue;

            }

            // Muestra el conjunto ingresado

            cout << "\nConjunto ingresado: " << nombre << " = { ";

            for (const auto& e : elementos) cout << e << " ";

            cout << "}" << endl;

            // Pide confirmación al usuario

            cout << "¿Deseas dejar este conjunto así? (s/n): ";

            string confirmar;

            getline(cin, confirmar);

            if (!confirmar.empty() && (confirmar[0] == 's' || confirmar[0] == 'S')) {

                conjuntos[nombre] = elementos;     // Guarda el conjunto

                break;

            } else {

                cout << "Reingresando los elementos del conjunto..." << endl;

            }

        }

    }

    // Muestra todos los conjuntos ingresados

    cout << "\nConjuntos ingresados:" << endl;

    for (const auto& par : conjuntos) {

        cout << par.first << " = { ";

        for (const auto& e : par.second) cout << e << " ";

        cout << "}" << endl;

    }

    // Guarda los conjuntos en un archivo de texto

    ofstream archivo("conjuntos.txt");

    for (const auto& par : conjuntos) {

        archivo << par.first << ":";          // Nombre del conjunto

        for (const auto& e : par.second) {

            archivo << e << ",";              // Elementos separados por coma

        }

        archivo << "\n";

    }

    cout << "\nConjuntos guardados en conjuntos.txt" << endl;

    return 0;

}

* **Código de Python:**

# === Instalación y verificación de dependencias necesarias ===

import subprocess

import sys

print("un momento estamos instalando todo lo necesario  :3")

def actualizar\_pip():

    """

    Intenta actualizar pip a la última versión.

    """

    try:

        print("Actualizando pip...")

        subprocess.check\_call([sys.executable, "-m", "pip", "install", "--upgrade", "pip"])

        print("pip actualizado correctamente.")

    except subprocess.CalledProcessError:

        print("Error al actualizar pip.")

actualizar\_pip()

# Lista de paquetes externos y módulos estándar requeridos

paquetes = ["Pillow", "matplotlib", "matplotlib-venn"]

modulos\_estandar = ["tkinter", "os", "subprocess", "sys"]

# Verifica e instala los paquetes necesarios

for paquete in paquetes:

    try:

        nombre\_import = "PIL" if paquete == "Pillow" else paquete

        \_\_import\_\_(nombre\_import)

    except ImportError:

        print(f"Instalando {paquete}...")

        try:

            subprocess.check\_call([sys.executable, "-m", "pip", "install", paquete])

            print(f"{paquete} instalado correctamente.")

        except subprocess.CalledProcessError:

            print(f"Error al instalar {paquete}.")

print("listo puedes continuar...")

# === Librerias necesarias ===

import tkinter as tk

import os

from tkinter import messagebox

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from matplotlib\_venn import venn2, venn3

import subprocess

from tkinter import PhotoImage

from PIL import Image

# === Funciones de operaciones entre conjuntos ===

def union(conj1, conj2):

    """Retorna la unión entre dos conjuntos."""

    return conj1.union(conj2)

def interseccion(conj1, conj2):

    """Retorna la intersección entre dos conjuntos."""

    return conj1.intersection(conj2)

def diferencia(conj1, conj2):

    """Retorna la diferencia entre dos conjuntos (conj1 - conj2)."""

    return conj1.difference(conj2)

def diferencia\_simetrica(conj1, conj2):

    """Retorna la diferencia simétrica entre dos conjuntos."""

    return conj1.symmetric\_difference(conj2)

def complemento(conj, universo):

    """Retorna el complemento del conjunto respecto al universo."""

    return universo.difference(conj)

# === Gestión de conjuntos cargados ===

conjuntos = {}

canvas\_diagrama = None

def cargar\_conjuntos\_desde\_archivo(nombre\_archivo="conjuntos.txt"):

    """

    Carga conjuntos desde un archivo de texto con formato: nombre:elem1,elem2,...

    """

    try:

        with open(nombre\_archivo, "r") as f:

            for linea in f:

                if ":" in linea:

                    nombre, elementos = linea.strip().split(":")

                    conjunto = set(e.strip() for e in elementos.split(",") if e.strip())

                    conjuntos[nombre] = conjunto

    except FileNotFoundError:

        messagebox.showerror("Error", "Archivo de conjuntos no encontrado.")

    actualizar\_lista\_conjuntos()

def actualizar\_lista\_conjuntos():

    """Actualiza la lista visual de conjuntos en la interfaz."""

    lista.delete(0, tk.END)

    for nombre, elementos in conjuntos.items():

        lista.insert(tk.END, f"{nombre} = {{{', '.join(sorted(elementos))}}}")

# === Funciones relacionadas con el programa en C++ ===

def ejecutar\_cpp\_en\_ventana():

    """Ejecuta el programa C++ en una nueva ventana CMD."""

    try:

        subprocess.Popen(["start", "cmd", "/c", "conjuntos.exe"], shell=True)

        messagebox.showinfo("Ejecutando", "Se ha abierto una ventana para crear conjuntos.\nCompleta el proceso y luego vuelve aquí.")

    except Exception as e:

        messagebox.showerror("Error", f"No se pudo ejecutar el código C++: {e}")

def cargar\_desde\_cpp():

    """Carga los conjuntos generados desde el programa en C++."""

    try:

        cargar\_conjuntos\_desde\_archivo()

        messagebox.showinfo("Conjuntos Cargados", "Los conjuntos fueron cargados exitosamente desde conjuntos.txt.")

    except Exception as e:

        messagebox.showerror("Error", f"No se pudo cargar: {e}")

def crear\_nuevo\_conjunto():

 """Llama al programa en C++ para crear un nuevo conjunto."""

 ejecutar\_cpp\_en\_ventana()

def limpiar\_lista\_de\_conjuntos():

    """Limpia todos los conjuntos cargados."""

    conjuntos.clear()

    lista.delete(0, tk.END)

def eliminar\_un\_conjunto\_especifico\_de\_la\_lista():

    seleccion = lista.curselection()

    """Elimina un conjunto seleccionado de la lista."""

    if seleccion:

        item\_texto = lista.get(seleccion[0])

        nombre\_conjunto = item\_texto.split("=")[0].strip()

        if nombre\_conjunto in conjuntos:

            del conjuntos[nombre\_conjunto]

            actualizar\_lista\_conjuntos()

    else:

        messagebox.showwarning("Advertencia", "Selecciona un conjunto para eliminar.")

def salir():

    """Cierra la aplicación."""

    ventana.destroy()

# === Visualización en Diagramas de Venn ===

def generar\_diagrama():

    """

    Genera un diagrama de Venn (2 o 3 conjuntos seleccionados).

    """

    sel = lista.curselection()

    nombres\_sel = [lista.get(i).split('=')[0].strip() for i in sel]

    if len(nombres\_sel) not in [2, 3]:

        messagebox.showwarning("Atención", "Selecciona 2 o 3 conjuntos para generar un diagrama de Venn.")

        return

    conj\_list = [conjuntos[nombre] for nombre in nombres\_sel]

    global canvas\_diagrama

    for widget in frame\_centro.winfo\_children():

        widget.destroy()

    fig, ax = plt.subplots(figsize=(5.5, 4))

    if len(conj\_list) == 2:

        venn2(conj\_list, set\_labels=nombres\_sel)

    else:

        venn3(conj\_list, set\_labels=nombres\_sel)

    plt.title("Diagrama de Venn")

    canvas\_diagrama = FigureCanvasTkAgg(fig, master=frame\_centro)

    canvas\_diagrama.draw()

    canvas\_diagrama.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

# === Mostrar resultados de operaciones ===

def mostrar\_resultado(op):

    """

    Muestra el resultado de la operación de conjuntos seleccionada.

    """

    sel = lista.curselection()

    nombres\_sel = [lista.get(i).split('=')[0].strip() for i in sel]

    resultado = set()

    operacion = ""

    valido = True

    if op in ["Unión", "Intersección", "Diferencia", "Diferencia Simétrica"]:

        if len(nombres\_sel) < 2:

            messagebox.showwarning("Atención", "Selecciona dos conjuntos para esta operación.")

            return

        A, B = conjuntos[nombres\_sel[0]], conjuntos[nombres\_sel[1]]

        if op == "Unión":

            resultado = union(A, B)

            print(f"{A} ∪ {B} = {{{', '.join(sorted(resultado))}}}")

        elif op == "Intersección":

            resultado = interseccion(A, B)

            print(f"{A} ∩ {B} = {{{', '.join(sorted(resultado))}}}")

        elif op == "Diferencia":

            resultado = diferencia(A, B)

            print(f"{A} - {B} = {{{', '.join(sorted(resultado))}}}")

        elif op == "Diferencia Simétrica":

            resultado = diferencia\_simetrica(A, B)

            print(f"{A} △ {B} = {{{', '.join(sorted(resultado))}}}")

        operacion = f"{nombres\_sel[0]} {op} {nombres\_sel[1]}"

    elif op == "Complemento":

        if len(nombres\_sel) != 1:

            messagebox.showwarning("Atención", "Selecciona solo un conjunto para el complemento.")

            return

        c = nombres\_sel[0]

        universo = set().union(\*conjuntos.values())

        resultado = complemento(conjuntos[c], universo)

        operacion = f"Complemento de {c}"

    else:

        valido = False

    if valido:

        texto\_resultado.delete("1.0", tk.END)

        texto\_resultado.insert(tk.END, f"{operacion} = {{ {', '.join(sorted(resultado))} }}")

        for widget in frame\_centro.winfo\_children():

            widget.destroy()

        fig, ax = plt.subplots(figsize=(5.5, 4))

        if op == "Complemento":

            venn2([universo, conjuntos[c]], set\_labels=("Universal", c))

        else:

            venn2([A, B], set\_labels=(nombres\_sel[0], nombres\_sel[1]))

        plt.title(operacion)

        global canvas\_diagrama

        canvas\_diagrama = FigureCanvasTkAgg(fig, master=frame\_centro)

        canvas\_diagrama.draw()

        canvas\_diagrama.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

# === Interfaz gráfica principal ===

def iniciar\_programa\_principal():

    """

    Inicializa y ejecuta la interfaz gráfica del programa.

    """

    global ventana, lista, frame\_centro, texto\_resultado, canvas\_diagrama

    ventana = tk.Tk()

    ventana.title("Teoría de Conjuntos")

    ventana.geometry("1000x580")

    tk.Label(ventana, text="Teoría de Conjuntos", font=("Arial", 20, "bold"), fg="red", bg="yellow").pack(fill=tk.X)

    # Panel Izquierdo - Lista de conjuntos

    frame\_izq = tk.Frame(ventana)

    frame\_izq.place(x=10, y=50, width=220, height=300)

    tk.Label(frame\_izq, text="Conjuntos Guardados", font=("Arial", 12, "bold"), borderwidth=2, relief="groove").pack(fill=tk.X)

    lista = tk.Listbox(frame\_izq, selectmode=tk.MULTIPLE)

    lista.pack(fill=tk.BOTH, expand=True)

    # Panel Central - Diagrama

    frame\_centro = tk.Frame(ventana, bg="skyblue")

    frame\_centro.place(x=240, y=50, width=500, height=300)

    # Panel Derecho - Resultado de operaciones

    frame\_der = tk.Frame(ventana)

    frame\_der.place(x=750, y=50, width=220, height=300)

    tk.Label(frame\_der, text="Resultado de la operación", font=("Arial", 12, "bold"), borderwidth=2, relief="groove").pack(fill=tk.X)

    texto\_resultado = tk.Text(frame\_der, wrap=tk.WORD, height=15)

    texto\_resultado.pack(fill=tk.BOTH, expand=True)

    # Botones principales

    tk.Button(ventana, text="Actualizar Lista de Conjuntos", command=cargar\_desde\_cpp, bg="blue", fg="white").place(x=240, y=360)

    tk.Button(ventana, text="Generar diagrama", command=generar\_diagrama, bg="red", fg="white").place(x=450, y=360)

    tk.Button(ventana, text="Crear nuevo conjunto", command=crear\_nuevo\_conjunto, bg="green", fg="white").place(x=615, y=360)

    tk.Button(ventana, text="Limpiar", command=limpiar\_lista\_de\_conjuntos, bg="red", fg="white").place(x=80, y=400)

    tk.Button(ventana, text="Eliminar un conjunto en especifico", command=eliminar\_un\_conjunto\_especifico\_de\_la\_lista, bg="yellow", fg="black").place(x=20, y=360)

    tk.Label(ventana, text="Operaciones Disponibles", bg="yellow", font=("Arial", 15, "bold")).place(x=380, y=420)

    tk.Button(ventana, text="Salir del programa", command=salir, bg="red", fg="white").place(x=880, y=530)

    # Botones de operaciones de conjuntos

    tk.Button(ventana, text="UNIÓN", bg="lightgreen", width=18, command=lambda: mostrar\_resultado("Unión")).place(x=230, y=460)

    tk.Button(ventana, text="INTERSECCIÓN", bg="lightgreen", width=18, command=lambda: mostrar\_resultado("Intersección")).place(x=410, y=460)

    tk.Button(ventana, text="DIFERENCIA", bg="lightgreen", width=18, command=lambda: mostrar\_resultado("Diferencia")).place(x=590, y=460)

    tk.Button(ventana, text="DIFERENCIA SIMÉTRICA", bg="lightgreen", width=18, command=lambda: mostrar\_resultado("Diferencia Simétrica")).place(x=320, y=520)

    tk.Button(ventana, text="COMPLEMENTO", bg="lightgreen", width=18, command=lambda: mostrar\_resultado("Complemento")).place(x=510, y=520)

    ventana.mainloop()

# === Cuadro de presentacion del grupo ===

import tkinter as tk

from PIL import Image, ImageTk

def mostrar\_splash\_screen(callback):

    """

    Muestra una pantalla de presentación antes de abrir la interfaz principal.

    """

    splash = tk.Tk()

    splash.title("Proyecto Estructuras Lógicas")

    splash.resizable(False, False)

 # Centrado y dimencones de la ventana de presentacion

    ancho\_ventana = 600

    alto\_ventana = 300

    pantalla\_ancho = splash.winfo\_screenwidth()

    pantalla\_alto = splash.winfo\_screenheight()

    x = (pantalla\_ancho - ancho\_ventana) // 2

    y = (pantalla\_alto - alto\_ventana) // 2

    splash.geometry(f"{ancho\_ventana}x{alto\_ventana}+{x}+{y}")

    splash.configure(bg="#0C6913")

   # Imagen y mensaje del proyecto

    ruta\_base = os.path.dirname(os.path.abspath(\_\_file\_\_))

    ruta\_imagen = os.path.join(ruta\_base, "logomeso.jpg")

    imagen = Image.open(ruta\_imagen)

    imagen = imagen.resize((250, 100))

    imagen\_tk = ImageTk.PhotoImage(imagen)

    etiqueta\_imagen = tk.Label(splash, image=imagen\_tk, bg="#0C6913")

    etiqueta\_imagen.image = imagen\_tk

    etiqueta\_imagen.pack(pady=(10, 0))

    mensaje = (

        "ESTRUCTURAS LÓGICAS\n"

        "PROYECTO: Programa de creación, operación y graficación de conjuntos.\n\n"

        "INTEGRANTES:\n"

        "🟢 Wendy Yamileth López Quijivix         - Carnet No. 202508081\n"

        "🟢 Emily Dayana Del Carmen Cabrera   - Carnet No. 202508083\n"

        "🟢 Miguel Abraham Mendoza López      - Carnet No. 202508086"

    )

    etiqueta\_texto = tk.Label(

        splash,

        text=mensaje,

        font=("Helvetica", 13, "bold"),

        fg="black",

        bg="white",

        justify="left",

        wraplength=550

    )

    etiqueta\_texto.pack(pady=(10, 0))

    # Cierra el Cuadro de presentacion del grupo y lanza el programa principal

    def cerrar\_y\_abrir():

        splash.destroy()

        callback()

    splash.after(10000, cerrar\_y\_abrir)

    splash.bind("<Key>", lambda event: cerrar\_y\_abrir())

    splash.mainloop()

# === Lanzamiento del programa ===

mostrar\_splash\_screen(iniciar\_programa\_principal)