	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorio de computación Salas A y B

Profesor: Dr. Ismael Everardo Bárcenas Patiño

Asignatura: Estructuras de datos y algoritmos I

Grupo: 3

No. de práctica: 2

Integrantes:
10 Gurrión Aquino Carlos Ángel
14 León Ruiz Eduardo
16 Macías Niño Carmen Violeta
17 Marroquín García Ricardo
21 Montaño Torres Rodolfo Santiago

No. de equipo de cómputo empleado:

No. de brigada: 7

Semestre: 2022-1

Fecha de entrega: 14 de septiembre de 2021

Observaciones:

Calificación:

Rúbrica de evaluación:

100	El programa cumple con todos los requerimientos
80-99	El programa cumple con la mayoría de los requerimientos
60-79	El programa cumple con algunos de los requerimientos
50-59	La lógica del programa es correcta, pero no corre o se cuelga

Práctica 2 | Aplicaciones de apuntadores

Problema a resolver:

I. Escribir un programa en lenguaje C que cumpla con las siguientes características:

- calcule la unión de conjuntos. Dados los conjuntos A y B, la unión se define como

$$A \cup B = \{x | x \in A \vee x \in B\};$$

- calcule la intersección de conjuntos. Dados los conjuntos A y B, la intersección se define como

$$A \cap B = \{x | x \in A \wedge x \in B\}.$$

El cálculo de la unión e intersección de conjuntos se debe realizar en funciones con 3 parámetros pasados por referencia A, B y C, donde A y B son los conjuntos de entrada y C el conjunto unión o intersección, según sea el caso.

Además, A, B y C deben ser:

- implementados en arreglos de tipo carácter y
- deben ser accedidos a través la aritmética de direcciones.

Código solución:

Enlace al código en compilador en línea: https://onlinegdb.com/NnbQoXLZD

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

//Declaración de variables globales
      //Dos espacios en memoria que actuán como iteradores en varias funciones
int i;
int j;

      //Dos espacios en memoria para almacenar el tamaño de los dos conjuntos
int m;
int n;

      //Varios arreglos de caracteres para almacenar los nombres predefinidos de los diferentes conjuntos
char a[] = "mA0";
char b[] = "nB";
char aa[] = " A";
char bb[] = " B";
char un[] = " unión";
char in[] = " intersección";
char aub[] = " A u B";
char anb[] = " A n B";


//Declaración de funciones
void progDescription();  //Imprime la descripción del programa
char* readSet(char* x, int* a);  //Lee el tamaño y los elementos de los diferentes conjuntos
void printSet(char* x, char* y, int* a, char* X);  //Imprime los elementos de los conjuntos
int unionSet(char* X, char* Y, char* U);  //Calcula la unión entre dos conjuntos dados
int intersectionSet(char* X, char* Y, char* I);  //Calcula la intersección entre dos conjuntos dados
void validRep(char* X);  //Valida si hay o no elementos repetidos dentro de un conjunto


//Función principal
float main()
{
              //Dos apuntadores que guardarán la dirección de los dos arreglos
      char* A;
      char* B;

      progDescription();  //Se imprime la descripción

      A = readSet(a, &m);  //Se lee el primer conjunto y se guarda

      while (m > 0)  //while que valida si se ingresó un dato válido
      {

              B = readSet(b, &n);  //Se lee el segundo conjunto y se guarda

              while (n > 0)  //while que valida si se ingresó un dato válido
              {
                      char U[m + n];  //Se crea el conjunto unión con un tamaño de máximo m + n elementos
                      char I[1];  //Se crea el conjunto intersección con un tamaño provisional

                      if (m < n)  //if que nos sirve para saber cuál conjunto es más pequeño
                      {
                              char I[m];  //Hacer que la intersección tenga el tamaño más pequeño posible
                      }
```

```
                else
                {
                        char I[n];
                }

                printSet(aa, aa, &m, A);  //Se imprime el primer conjunto
                printSet(bb, bb, &n, B);  //Se imprime el segundo conjunto

                int u = unionSet(A, B, U);  //Se ejecuta la unión de conjuntos
                printSet(un, aub, &u, U);  //Se imprime el conjunto unión


                int v = intersectionSet(A, B, I);  //Se ejecuta la intersección de conjuntos
                printSet(in, anb, &v, I);  //Se imprime el conjunto intersección

                return 2.718281;
        }

        printf("\nHa ingresado un valor no permitido. No se puede ejecutar el programa.");
        return 2.718281;
    }

    printf("\nHa ingresado un valor no permitido. No se puede ejecutar el programa.");
    return 2.718281;
}


void progDescription()
{
    printf("PROGRAMA PARA CALCULAR LA UNIÓN E INTERSECCIÓN DE DOS CONJUNTOS\n");
    printf("\nEste programa recibe como entrada dos conjuntos de caracteres %c y %c ", *(a + 1), *(b + 1));
    printf("de tamaño %c y %c respectivamente (donde %c y %c son enteros positivos). ", *a, *b, *a, *b);
    printf("Ambos conjuntos no pueden contener en ellos elementos repetidos. ");
    printf("El programa calcula la unión y la intersección de ambos conjuntos. Y como dato de salida se
imprimen ");
    printf("los conjuntos %c y %c, el conjunto unión y el conjunto intersección.\n", *(a + 1), *(b + 1));
}


char* readSet(char* x, int* a)
{
    printf("\n\nIngrese la cardinalidad (%c) del conjunto %c:\n\t%c = ", *x, *(x + 1), *x);
    scanf("%d", a);

    while(*a > 0)  //while que valida que se ingrese un número y no otro tipo de dato
    {
        char* X = (char*)calloc(*a, sizeof(char));

        printf("\tIngresa los %d elementos (caracteres no repetidos) del conjunto %c:\n", *a, *(x + 1));

        for (i = 0; i < *a; i ++)  //for que recorre la longitud del arreglo
        {
            printf("\tElemento no. %d = ", i + 1);
            scanf(" %c", X + i);  //Se almacenan los elementos del arreglo

            if (i > 0)
            {
                validRep(X);  //Función que revisa si no hay elementos repetidos
            }
        }
        return X;
    }
```

```
}


void validRep(char* X)
{
	for (j = 0; j < i; j++)  //Se recorren los elementos del arreglo que ya hemos ingresado
	{
		if (*(X + i) == *(X + j))  //Si el elemento en cuestión es igual a uno que ya ingresamos
		{
			printf("\n\tNo puedes ingresar elementos repetidos.\n\tIngresa otra vez el elemento no. %d
= ", i + 1);
			scanf(" %c", X + i);
			validRep(X);  //La función se llama a sí misma para revisar de nuevo
		}
	}
}


void printSet(char* x, char* y, int* a, char* X)
{
	printf("\n\nConjunto%s:\n\t\t%s = (", x, y);

	for (i = 0; i < *a; i ++)  //Se recorre el arreglo
	{
		if (i == *a - 1)  //Si el índice es el último...
		{
			printf("%c", *(X + i));  //Imprime el último elemento (sin coma delante)
			break;
		}
		printf("%c, ", *(X + i));  //Se imprime cada elemento (seguido de una coma)
	}
	printf(")\n");
}


int unionSet(char* X, char* Y, char* U)
{
	int i2 = 0;  //Variable que va a ir contando el número de elementos del conjunto unión

	for (i = 0; i < m; i ++)  //Se recorre el primer arreglo
	{
		for (j = 0; j < n; j ++)  //Se recorre el segundo arreglo
		{
			if (*(X + i) != *(Y + j))  //Si algún elemento no se repite
			{
				if (j == n - 1)
				{
					*(U + i2) = *(X + i);  //Se guarda el elemento no repetido
					i2 ++;
				}
			}
			else
			{
				break;
			}
		}
	}

	for (i = 0; i < n; i ++)  //Se recorre el segundo arreglo
	{
		*(U + i2) = *(Y + i);  //Se guardan todos los elementos del segundo arreglo
		i2 ++;
	}
```

```
	return i2;  //La función retorna el número de elementos que posee el conjunto unión
}


int intersectionSet(char* X, char* Y, char* I)
{
	int i2 = 0;  //Variable que contará el número de elementos del conjunto intersección

	for (i = 0; i < m; i ++)  //Se recorre el primer arreglo
	{
		for (j = 0; j < n; j ++)  //Se recorre el segundo arreglo
		{
			if (*(X + i) == *(Y + j))  //Si algún elemento se repite en ambos arreglos...
			{
				*(I + i2) = *(X + i);  //Se guarda ese elemento en común
				i2 ++;
			}
		}
	}

	free(X);  //Se libera el espacio en memoria que ocupaban los arreglos
	free(Y);

	return i2;  //Se retorna el números de elementos que posee el conjunto intersección
}
```