ENUNCIADO DEL PROBLEMA:

Realizar un programa que trabaje con CONJUNTOS. Se seguirán estos tres pasos:

PASO 1: Realizar la carga de datos de dos conjuntos A y B.

PASO 2: Las operaciones que se deben poder realizar sobre los conjuntos son

- PERTENECE un elemento a un conjunto.
- A UNION B
- A INTERSECCIÓN B
- A RESTA B

PASO 3: Rehacer el programa con funciones y punteros.

OBJETIVOS:

(Para determinar porqué hacemos el trabajo, con qué intereses y poder medir finalmente si hemos alcanzado los objetivos)

A) Personales:
   i) Aprender a programar con prestanza para superar el exámen.
   ii) ...

B) De grupo:
   i) Explotar el trabajo en equipo para tener mejor rendimiento en esta y en otras asignaturas.
   ii) ...

PLANIFICACIÓN Y SEGUIMIENTO:

(Para repartir trabajos entre los miembros del grupo, controlar los tiempos y gestionar las desviaciones)

PLAZO ENTREGA	TAREA	RESPONSABLE	HORAS PREVISTAS	HORAS REALES	HORAS DESVIO
6 DE ABRIL	OBJETIVOS	GLORIA, IKER	1	1	0
6 DE ABRIL	PLANIFICACIÓN	IKER, GLORIA	1	0’5	-0’5
10 DE ABRIL	PENSAR CONTENIDO	IKER, SIDONIO	6	5	-1
11 DE ABRIL	ESCRIBIR DOCUMENTO CONTENIDO	IKER	2	4	+2
18 DE ABRIL	IMPLEMENTAR PRIMER PASO	SIDONIO, IKER	6	3	-3
9 DE MAYO	IMPLEMENTAR SEGUNDO PASO	SIDONIO, IKER	12	10	-2
16 DE MAYO	IMPLEMENTAR TERCER PASO	SIDONIO, IKER	6	7	+1
23 DE MAYO	DOCUMENTO FINAL	IKER	4	5	+1
25 DE MAYO	PREPARAR PRESENTACIÓN	IKER, GLORIA	2	3	+1
		TOTAL	40	38’5	-1’5

Metodología: nos comunicaremos por correo electrónico y todos los viernes ……

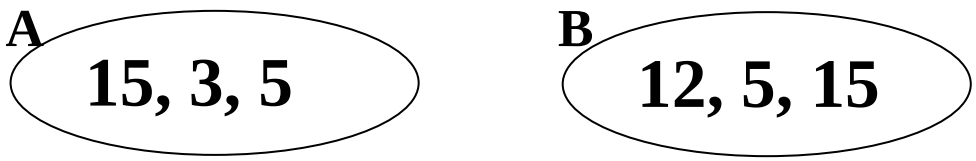
CONTENIDO:

(Responder a la pregunta ¿qué hay que hacer? Para entenderlo y diseñar una solución coherente al problema antes de empezar a implementar.)

COMPRENDER EL PROBLEMA:

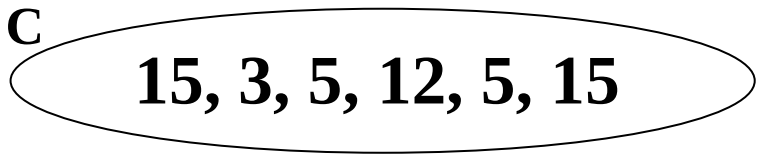
Un conjunto engloba elementos. Vamos a implementar conjuntos de números enteros. Tenemos dos conjuntos A y B y una serie de operaciones sobre ellos. La solución de algunas operaciones producen la creación de un nuevo conjunto solución. A continuación se muestra una representación teórica del problema que se plantea:

CONJUNTOS INICIALES:

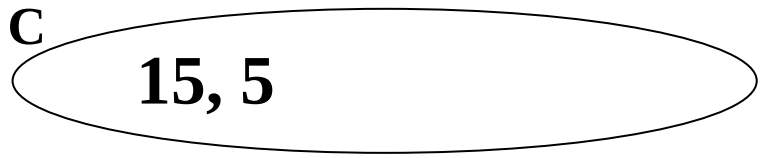


PERTENECE:
El elemento 3 PERTENECE al conjunto A. El elemento 8 no pertenece al conjunto A.

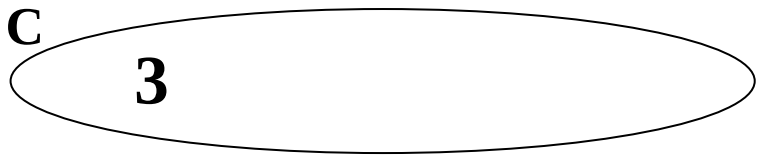
C = A UNION B:
Serán elementos de C aquellos que lo son de A o de B. Se admiten repeticiones.



C = A INTERSECCIÓN B:
Serán elementos de C aquellos elementos que pertenezcan al mismo tiempo al conjunto A y al conjunto B.



C = A RESTA B:
Serán elementos de C aquellos elementos que pertenecen al conjunto A y no al conjunto B.



SOLUCIONES PLANTEADAS:

Se debe seleccionar una estructura de datos que nos permita reflejar este comportamiento en un programa informático.

ESTRUCTURA DE DATOS 1:

La primera estructura de datos que creíamos mejor se ajustaba a este comportamiento era la de una tabla de dimensiones: 2 x 100. la primera fila guardaría el valor del elemento y la segunda el conjunto al que pertenece dicho elemento. La entrada de datos y las operaciones a implementar se realizarían así:

ENTRADA DE DATOS:
El usuario introduciría los datos como pares Elemento-Conjunto. Por ejemplo:
15 A
12 B
3 A
5 A
5 B
15 B
12 B
La repetición de un elemento (12 B) en la entrada de datos marcaría el final de la entrada de datos. Así también se evita que un conjunto tenga elementos repetidos. Los elementos podrán ser positivos o negativos pero entre el 100 y el –100. Esto lo hacemos así para evitar números muy grandes que se salgan del rango de los enteros (int). El segundo miembro del par indica el conjunto al que pertenece el elemento, que sólo puede ser el valor A o B.

CONJUNTOS INICIALES:
La tabla después de cargar esos datos quedaría de la forma:

15	12	3	5	5	15									
A	B	A	A	B	B									

Necesitaríamos un contador (CONT_ELEMENTOS, en este ejemplo tendría el valor 5) que nos indique cuantos datos de la tabla son elementos de los dos conjuntos.

PERTENECE:
Se permitirá preguntar si un elemento pertenece a los conjuntos A o B. Si el usuario introduce el número 3 se recorrerá la fila 0 de la tabla (elementos de los conjuntos A o B, es decir, desde la columna 0 hasta donde indique el contador CONT_ELEMENTOS). Si el programa encuentra el 3 comprobará en la misma columna de la fila 1 a qué conjunto pertenece. Si es elemento del conjunto A asignará el valor 1 a la variable PERTENECE_A; si es elemento del conjunto B asignará el valor 1 a la variable PERTENECE_B. Al final comprobando si estas dos variables están a 0 o a uno el programa indicará a que conjuntos pertenece el elemento 3.

C = A UNION B:
La tabla después de realizar la operación será:

15	12	3	5	5	15	15	12	3	5	5	15			
A	B	A	A	B	B	C	C	C	C	C	C			

Para obtener esta solución el programa sólo tendrá que copiar los elementos de la fila 0 que está desde la columna 0 hasta CONT_ELEMENTOS en las posiciones libres posteriores. Además, se indicará en la fila 1 que pertenecen al conjunto C. Los elementos de C son los que están en las columnas que van de CONT_ELEMENTOS + 1 a 2*CONT_ELEMENTOS + 1.

C = A INTERSECCIÓN B:
La tabla después de realizar la operación será:

15	12	3	5	5	15	15	5							
A	B	A	A	B	B	C	C							

Por cada elemento que se encuentra en la tabla habrá que comprobar si en el resto de la tabla está y además pertenece al conjunto contrario. Por ejemplo, comenzamos con el elemento de la columna 0, el elemento 15, que pertenece al conjunto A. Lo comparamos con los elementos desde la columna 1 hasta la columna CONT_ELEMENTOS. Lo encontramos en la columna 5 y vemos que es del conjunto B. Por lo tanto el 15 tiene que estar en el conjunto C. Lo copiamos al final. Seguimos con la columna 1, el elemento 12 del conjunto B, y lo comparamos con el resto de los elementos. No lo encontramos con lo que no debe estar en C. Y así sucesivamente. Para saber los elementos de C necesitaremos un contador CONT_RESULTADO. Los

elementos de C son los que están en las columnas desde la CONT_ELEMENTOS + 1 hasta la CONT_RESULTADO.

C = A RESTA B:
La tabla después de realizar la operación será:

15	12	3	5	5	15	3								
A	B	A	A	B	B	C								

Habrá que recorrer la tabla buscando los elementos del conjunto A. Cada vez que se encuentre uno habrá que recorrer desde el comienzo la tabla para ver si está también en el conjunto B. Por ejemplo, el elemento de la columna 0, el elemento 15 del conjunto A, lo encontramos en la columna 5 perteneciendo a B. Este no está en el conjunto C. El elemento de la columna 2, el elemento 3 del conjunto A, lo comparamos con todos los elementos de la tabla desde la columna 0 hasta la CONT_RESULTADO, y no lo encontramos, por lo que estará en el conjunto C. Los elementos de C son los que están en las columnas desde la CONT_ELEMENTOS + 1 hasta la CONT_RESULTADO.

PROBLEMAS DE LA SOLUCIÓN:
No parece una forma natural de gestionar los datos. Los elementos de los conjuntos A y B están mezclados. Para las operaciones INTERSECCIÓN y RESTA sería mejor tenerlos separados para saber cuales son de A y cuales de B. Así habría que hacer menos trabajo, por ejemplo, se recorrería menos la tabla.
Además, por definición, los conjuntos pueden tener elementos repetidos por lo que no parece correcto que tal y como se ha pensado la entrada de datos ésta tenga que impedir elementos repetidos.

ESTRUCTURA DE DATOS 2:

Para mejorar la solución proponemos una tabla de dimensiones: 3 x 100. La fila 0 guardaría los elementos del conjunto A, la fila 1 los del conjunto B y la fila 2 los del conjunto C. La entrada de datos y las operaciones a implementar se realizarían así:

ENTRADA DE DATOS:
Los elementos podrán ser positivos o negativos pero entre el 100 y el –100. Esto lo hacemos así para evitar números muy grandes que se salgan del rango de los enteros (int). El usuario podría seleccionar de un submenú el conjunto en el que quiere cargar los elementos:

1. Conjunto A
2. Conjunto B

Seguidamente introduciría los datos del conjunto seleccionado como una ristra de datos que terminan con un número no permitido. Esta solución permite que un conjunto tenga elementos repetidos.
Por ejemplo, la entrada de datos se haría así:
Datos del conjunto A: 15 3 5 101
Datos del conjunto B: 12 5 15 345

CONJUNTOS INICIALES:
La tabla después de cargar esos datos quedaría de la forma:

15	3	5	101											
12	5	15	345											

De alguna forma hay que indicar cuantos elementos tiene cada conjunto. Se puede hacer mediante un contador para cada conjunto (como se hacia en la primera solución) o con una marca especial . Optamos por la marca especial ya que la entrada de datos se controla de esta forma.

PERTENECE:
Se permitirá preguntar si un elemento pertenece a los conjuntos A o B. Si el usuario introduce el número 3 se recorrerá la fila 0 de la tabla (elementos del conjunto A) desde la columna 0 hasta un número que no este entre 100 y –100. Si el programa encuentra el 3 indicará directamente que si pertenece al conjunto A. No es necesario seguir comparando con el resto de elementos de A. Se haría lo mismo con el conjunto B (fila 1).

C = A UNION B:
La tabla después de realizar la operación será:

15	3	5	101											
12	5	15	345											
15	3	5	12	5	15	999								

El programa copiará en la fila 2 los elementos validos de la fila 0 y luego los de la fila 1. Al final ponemos una marca especial de final de conjunto. Por ejemplo el número 999

C = A INTERSECCIÓN B:
La tabla después de realizar la operación será:

15	3	5	101											
12	5	15	345											
15	5	999												

Cada elemento de la fila 0 lo comparamos con todos los de la fila 1. Si está en la fila 1 lo copiamos en la fila 2. Así con todos. Finalmente ponemos la marca 999.

C = A RESTA B:
La tabla después de realizar la operación será:

15	3	5	101											
12	5	15	345											
3	999													

Cada elemento de la fila 0 lo comparamos con todos los de la fila 1. Si NO está en la fila 1 lo copiamos en la fila 2. Así con todos. Finalmente ponemos la marca 999.

PROBLEMAS DE LA SOLUCIÓN:
Si comparamos esta solución con la anterior vemos que las operaciones se han simplificado. No tiene pegas especiales a no ser la limitación del universo al rango 100, -100. No es sencillo justificar esta limitación. Se puede optar por una solución más abierta: solicitarle al usuario que indique los límites inferior y superior del universo con el que desea trabajar.

OTRAS CUESTIONES:

Aunque en el enunciado no se plantean habrá otras operaciones que deba implementar el programa. Además de la validación de los datos de entrada, mensajes adecuados, menus y submenus, se precisa de una operación de visualización de los conjuntos. Esta operación será útil para el usuario y también para los programadores que podrán comprobar que las otras operaciones se realizan correctamente.

RESULTADOS:

(Solución a los problemas)
Los programas documentados se pueden encontrar en:

PRIMER PASO: paso1.c
SEGUNDO PASO: paso2.c
TERCER PASO: paso3.c

Esta memoria de proyecto se encuentra en el archivo memoria.pdf y la presentación realizada en clase en presentación.ppt.

El primer y segundo paso siguen el diseño planteado en el apartado CONTENIDO. En el tercer paso, se debía reescribir el programa usando funciones y punteros. Los punteros permiten acceder a las tablas y pasarlas como parámetros a las funciones. Las funciones agrupan el código que se utiliza varias veces o que tienen una función clara. Las funciones que hemos creado han sido:

```
//PROTOTIPOS
void visualizar (int *puntero, int LimInf, int LimSup);
int pertenece (int *puntero, int Elem, int LimInf, int LimSup);
void cargaElementos (int *puntero, int LimInf, int LimSup);
void interseccion (int *punteroA, int *punteroB, int *punteroC, int LimInf, int LimSup);
void resta (int *punteroA, int *punteroB, int *punteroC, int LimInf, int LimSup);
void unir (int *punteroA, int *punteroB, int *punteroC, int LimInf, int LimSup);
```

VISUALIZAR: Presenta los elementos de un conjunto por pantalla.
PERTENECE: Indica si ELEM pertenece al conjunto apuntado por el parámetro puntero.
CARGAELEMENTOS: Introduce los elementos suministrados por el usuario en el conjunto.
INTERSECCIÓN: realiza la intersección de los conjuntos A y B y el resultado lo guarda en el conjunto apuntado por PUNTEROC.
RESTA: realiza la operación resta. Para ello se utiliza la operación PERTENECE.
UNIR: realiza la operación unión. Para ello se utiliza la operación PERTENECE.

CONCLUSIONES:

(Comprobar si se han cumplido los objetivos, realizar recomendaciones a los alumnos de próximos cursos, etc)

Respecto a los objetivos planteados inicialmente creo ……
El trabajo en equipo ha sido ……

En general creo que la forma en que se ha planteado la práctica este año es …….

A los alumnos de próximos años les recomendaría que……

Mi valoración final es …….

LICENCIA:

(Quién ha realizado el trabajo, cómo permite su distribución, etc.)

Este trabajo (memoria, código fuente que se cita en la memoria y presentación que se cita en la memoria) ha sido realizado por Gloria Vázquez, profesora en la Escuela de Magisterio de Bilbao, e Iker Azpeitia y Sidonio Perez, profesores de Fundamentos de Informática de la EUITI de Eibar.

Los autores permiten la distribución de estos contenidos según la siguiente licencia Creative Common:

http://creativecommons.org/licenses/by-nc-sa/2.5/es/

creative commons

COMMONS DEED

Reconocimiento-NoComercial-CompartirIgual 2.5 España

Usted es libre de:

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

Bajo las condiciones siguientes:

Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador.

No comercial. No puede utilizar esta obra para fines comerciales.

Compartir bajo la misma licencia. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor

Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.

Esto es un resumen legible por humanos del texto legal (la licencia completa) disponible en los idiomas siguientes:

Catalán Castellano Gallego

Advertencia