Especificación matemática ejercicios 31-50

José Simón Ramos Sandoval

Universidad Nacional de Colombia, Programación de computadores 2021-1s

1 Arreglos

Ejercicio 31

Suponga que un arreglo de enteros esta lleno de unos y ceros y que el arreglo representa un núumero binario al revés. Hacer un algoritmo que calcule los números en decimal que representa dicho arreglo de unos y ceros.

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conoce el tamaño del arreglo (n), que para el caso será la cantidad de bits ($n \in \mathbb{N}$)
- Se conocen los elementos del arreglo, estos elementos solo pueden ser o el 0 o el 1

1.2 Objetos desconocidos:

- Se desconoce el número decimal (X) que representa el arreglo binario (X $\in \mathbb{N}$)

1.3 Relaciones entre objetos:

- Para el caso, se garantiza que el número resultante es un entero positivo, o lo que es lo mismo, un número natural
- Para convertir un número de binario a decimal, se ubican de derecha a izquierda las potencias de dos empezando desde 2^0hasta 2^{n-1} (siendo n la cantidad de digitos del binario) debajo de los digitos del número binario, se multiplica el resultado de la potencia por el digito del número binario que se encuentra arriba, por último se suman todos los resultados de estas multiplicaciones

- Con la explicación anterior, observamos que al recibir el arreglo, el exponente de la potencia de dos se corresponde con la posición del digito binario en el arreglo. Siguiendo estas relaciones podemos establecer fácilmente el número decimal correspondiente mediante una función recursiva

- El caso base será cuando n = 1, en ese caso retornará el valor de la posicion A_0. mientras $n \neq 0$, hará una suma recursiva con el producto de la potencia de $2^i * A_i$

2. Diseño y prueba conceptual

Función que convierte de binario a decimal:

Entradas: 2 variables: 1 variable n que representa la cantidad de elementos del arreglo A; 1 arreglo $A \in \mathbb{N}^n$ tal que $\forall_{i=0}^{n-1} A_i = 0, 1$

S**alidas:** 1 variable: Número decimal al que equivale el arreglo binario

Relación:

binario: $(\mathbb{N}^{[0,1]*} \times \mathbb{N} \to \mathbb{N})$

$$\text{binario}(A, n) \to \begin{cases} A_0 & si \quad n = 1 \\ A_{n-1} * 2^{n-1} + binario(A, n-1) & \text{En otro caso} \end{cases}$$

3. Codificación

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_31_SimonRamos.py»

Ejercicio 32

Hacer un algoritmo que dado un número entero no negativo, cree un arreglo de unos y ceros que representa el número en binario al revés.

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conoce un número decimal entero no negativo ($num \in \mathbb{Z}^{0,+}$)

- Se conoce que para convertir un número de decimal a binario se hace una serie consecutiva de divisiones entre 2

1.2 Objetos desconocidos:

- Se desconoce el arreglo binario al que corresponde el número entero positivo dado inicialmente

1.3 Relación entre objetos:

- dado un número, para convertirlo a binario este número se divide por 2, el resultado se divide por 2 y así sucesivamente hasta que no se puedan efectuar más divisiones. Los módulos de las divisiones (que siempre serán o 0 o 1) formarán el código binario

- Al formar una lista con los módulos de las divisiones, se tiene el número binario al revés
- Los anteriores pasos pueden realizarse con una función recursiva. Ya que a un número x le puedo realizar divisiones consecutivas de 2, si también puedo realizarle divisiones consecutivas de 2 al número $\frac{x}{2}$. Se supone que la función que hace la divisón de x entre dos está bien, por lo que puedo dividir entre 2 a $\frac{x}{2}$. Puedo seguir haciendo divisiones mientras que $x \geq 2$

2. Diseño y prueba conceptual
Función convierte de decimal a binario:

Entradas: 2 variables: un número entero no negativo ($num \in \mathbb{Z}^{0,+}$) que será convertido en binario; 1 arreglo vacío donde se acumularán los binarios

Salidas: Un arreglo A, tal que $\forall_{i=0}^{|A|-1} A_i = 0, 1$

Relación:

$decimal : \mathbb{Z}^{0,+} \to \mathbb{N}^{[0,1]*}$

$$decimal(A, num) \to \begin{cases} [num] & si \quad num < 2 \\ A\text{tal que } A = [num \bmod 2] + \ decimal(\llbracket \frac{num}{2} \rrbracket) & \text{En otro caso} \end{cases}$$

3. Codificación

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_32_SimonRamos.py»

Ejercicio 33
Hacer un algoritmo que calcule el Máximo Común Divisor (MCD) para un arreglo de enteros positivos.
1. Análisis y especificación:
1.1 Objetos conocidos:

- Se conoce la cantidad de elementos del arreglo de enteros ($n \in \mathbb{N} : n \neq 0$)
- Se conocen los elementos del arreglo de enteros (A) $\begin{array}{ccc} \mathbb{Z}^* & \to & \mathbb{Z}^* \\ A & \to & (A_0, A_1, A_2, \cdots, A_{n-1}) \end{array}$

1.2 Objetos desconocidos:

- Se desconoce el máximo común divisor del arreglo de enteros positivos

1.3 Relación entre los onjetos:

- El máximo común divisor es el número más grande por el cuál todos los números del arreglo son divisibles.
- Según el algoritmo de Euclides, para determinar el mcd primero se debe garantizar que los números deben estar ordenados de forma descendente de mayor a menor. Es decir, $(A_0, A_1, A_2, \cdots, A_{n-1})$, tal que $A_0 \geq A_1 \geq A_2 \geq \cdots \geq A_{n-1}$

- Posteriormente, los nuevos términos del arreglo saldrán del módulo entre el primer número y el número de su derecha, hasta el número de la posición A_{n-1}, al cuál no se le hará ninunga operación. El arreglo resultante será $(A_0\%A_1, A_1\%A_2, \cdots, A_{n-2}\%A_{n-1}, A_{n-1})$. El procedimiento se repetirá ordenando de nuevo los elementos del arreglo. Cuando el módulo de algún termino sea igual a cero, este no se tendrá en cuenta y se repetirá el proceso hasta que todos los números del arreglo sean 0 menos el primero. El mcd será A_0 tal que $A_0 \neq 0 \wedge \forall_{i=1}^{|A|-1} A_i = 0$

- El algoritmo puede representarse como una función recursiva, ya que el mcd de un arreglo puede hallarse si puedo hallar el mcd de otro arreglo formado por los módulos del primer arreglo. Se supone que la función que halla el mcd del primer arreglo está bien, por lo que puedo usarla para hallar el mcd del segundo arreglo. Hallar el mcd de un arreglo con todos los términos menos el primero iguales a cero es fácil, ya que el mcd será el término A_0. Sin embargo, para hallar el mcd de los elementos de un arreglo se necesitan de varias funciones que se integren entre ellas

2. Diseño y prueba conceptual

ïndice del mínimo elemento de un arreglo:

Entradas: 2 variables: Un arreglo de enteros positivos (A^*) tal que $\forall_{i=0}^{|A|-1} A_i > 0$ y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento mínimo del arreglo ($m \in \mathbb{N}$)

Relación:

$min : \mathbb{Z}^{+*} \times \mathbb{N} \rightarrow \mathbb{N}$
$\min(A^*, n) \rightarrow m$ talque$m = n-1 \wedge A_m = A_{n-1}$pero si$\forall_{i=n-1, i=i-1}^{0} A_i <$
$A_m \rightarrow A_m = A_i \wedge m = i$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo de enteros positivos (A^*) tal que $\forall_{i=0}^{|A|-1} A_i > 0$ y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo B^*

Relación:

$ordena : \mathbb{Z}^{+*} \times \mathbb{N} \rightarrow \mathbb{R}^*$
$\text{ordena}(A^*, n) \rightarrow B^*$ tal que $\forall_{i=n-1, i=i-1}^{0} B_i = A_{min(A^*, i)} \rightarrow$
$B_0 \geq B_1 \geq B_2 \geq ... \geq B_{n-1}$

Función que busca el número de ceros del arreglo:

Entradas: 1 variables: un arreglo de enteros no negativos $\mathbb{Z}^{0,+*}$

Salidas: 1 variable: Un número natural x con el número de ceros que aparecen en el arreglo, ($x \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} ceros : \mathbb{Z}^{o,+*} & \rightarrow & \mathbb{N} \\ ceros(C) & \rightarrow & x \text{ tal que si } \exists_{i=0}^{|A|-1} C_i = 0 \rightarrow x+ = 1 \end{array}$$

Funcion que saca los modulos en el arreglo:

Entradas: 2 variable: 1 arreglo de enteros positivos (B^*) tal que $\forall_{i=0}^{|B|-1} B_i > 0$ y tal que $B_0 \geq B_1 \geq \cdots \geq B_{|B|-1}$(Se garantiza que el arreglo que esté ordenado de mayor a menor); 1 variable x que indica el número de ceros del arreglo ($x \in \mathbb{N}$)

Salidas: 1 arreglo de enteros no negativos $\mathbb{Z}^{0.+}$

Relación:

$$\begin{array}{rcl} modulos : \mathbb{Z}^{+*} \times \mathbb{N} & \rightarrow & \mathbb{Z}^* \\ modulos(B, x) & \rightarrow & \forall_{i=0}^{|B|-2-x} B_i = B_i \% B_{i+1} \end{array}$$

máximo común divisor:

Entradas: 1 arreglo de enteros positivos (B^*) tal que $\forall_{i=0}^{|B|-1} B_i > 0$ y tal que $B_0 \geq B_1 \geq \cdots \geq B_{|B|-1}$(Se garantiza que el arreglo que esté ordenado de mayor a menor)

Salidas: 1 variable: un número natural que es el máximo común divisor del arreglo ($maximo \in \mathbb{N}$)

$$mcd : \mathbb{Z}^{+*} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$mcd(A) \rightarrow \begin{cases} A_0 & si \quad A_1 = 0 \\ \text{mcd (modulos(A, ceros(A)))} & \text{En otro caso} \end{cases}$$

3. Codificación

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_33_SimonRamos.py»

Ejercicio 34

Hacer un algoritmo que calcule el Mínimo Común Múltiplo (MCM) para un arreglo de enteros positivos.

1. Análisis y especificación:

1.1 Objetos conocidos:

- Se conoce la cantidad de elementos del arreglo de enteros ($n \in \mathbb{N} : n \neq 0$)
- Se conocen los elementos del arreglo de enteros (A) $\begin{array}{rcl} \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ A & \rightarrow & (A_0, A_1, A_2, \cdots, A_{n-1}) \end{array}$

1.2 Objetos desconocidos:

- Se desconoce el mínimo común múltiplo del arreglo de enteros positivos

1.3 Relación entre los onjetos:

- El mínimo común múltiplo es el número más pequeño múltiplo de todos los números del arreglo.
- Para determinar el mínimo común múltiplo de dos números A y B se sigue la relación: $A \times B = mcm \times mcd \rightarrow mcm = \frac{A \times B}{mcd}$ .Así que para hallar el mcm se halla en primer lugar el mcd según el algoritmo de Euclides.
- Para hallar el mínimo común múltiplo de varios números, primero se halla el mínimo común múltiplo de los últimos dos números siguiendo el algoritmo de Euclídes $mcm = \frac{A \times B}{mcd}$. Posteriormente, se halla el minimo común múltiplo de la pareja formada por el siguiente número del arreglo y el minimo común múltiplo encontrado antes. Se sigue esta relación hasta el último número del arreglo.
- El algoritmo puede representarse como una función recursiva, ya que el mcm de un arreglo puede hallarse si puedo hallar el mcm_1 entre A_{n-1}, A_{n-2}, y posteriormente el mcm de A_{n-3}, mcm_1. Se supone que la función que halla el mcm de la primera pereja está bien, por lo que puedo usarla para hallar el mcm de la segunda pareja. Hallar el mcm de la última pareja es fácil, pues es solo utilizar la función mínimo común múltiplo cuando solo se tienen dos términos. Sin embargo, para hallar el mcd de los elementos de un arreglo se necesitan de varias funciones que se integren entre ellas

2. Diseño y prueba conceptual:
Máximo común divisor entre dos números:

Entradas: 2 variables: Dos números $x, y \in \mathbb{Z}^+$ tal que $x > y$

Salidas: 1 variable: El máximo común divisor de los dos números ($z \in \mathbb{Z}^+$)

Relación:

$$mcd : \mathbb{Z}^+ \times \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$$

$$mcd(x, y) \rightarrow \begin{cases} x & si \quad y = 0 \\ mcd(y, x) & si \quad y > x \\ mcd(y, x \ \% \ y) & \text{En otro caso} \end{cases}$$

Mínimo común múltiplo:

Entradas: 2 variables: Dos números $x, y \in \mathbb{Z}^+$

Salidas: 1 variable: El mínimo común múltiplo de los dos números ($m \in \mathbb{Z}^+$)

Relación:

$$mcm : \mathbb{Z}^+ \times \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$$

$$mcm(x, y) \rightarrow \frac{x * y}{mcd(x, y)}$$

Mínimo común múltiplo de un arreglo:

Entradas: 3 variables: 1 arrreglo A de enteros positivos ($A \in \mathbb{Z}^{+*}$); 1 variable que indica la cantidad de elementos del arreglo ($n \in \mathbb{N}$); mínimo común múltiplo de A_{n-1}y$A_{n-2}(m \in \mathbb{N})$

Salidas: 1 variable: El mínimo común múltiplo de todos los números del arreglo

Relación:

$$mincomun : \mathbb{Z}^{+*} \quad \times\mathbb{N} \times \mathbb{N} \rightarrow \quad \mathbb{Z}^{+}$$

$$mincomun(A, n, m) \quad \rightarrow \quad \begin{cases} m & si \quad n < 3 \\ mincomun(A,\ n-1,\ mcm(\ A_{n-3}, m\ )) & \text{En otro caso} \end{cases}$$

3. Codificación

> El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_34_SimonRamos.py»

2 Conjuntos como arreglos

Un arreglo de elementos de tipo T se puede utilizar para representar un conjunto finito de elementos del tipo T. Esta representación es como sigue:
El conjunto A = $\{x_0, x_1, x_2, \ldots, x_{n-1}\}$ se representa como el arreglo (x_0, x_1, x_2, …, x_{n-1}).
Usando esta representación hacer un programa que le permita al usuario leer dos conjuntos de enteros y escoger mediante un menú, una de las siguientes operaciones sobre ellos:
Después de realizada la operación, el menú se debe presentar de nuevo hasta que el usuario desee salir. Se debe verificar que el arreglo no tenga elementos repetidos.

Ejercicio 35

UNIÓN: Calcula en un arreglo la unión de los conjuntos y la imprime.

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior
- Al aplicar la operación de unión sobre estos conjuntos, los elementos del conjunto resultante no pueden estar repetidos y deben estar ordenados.

1.2 Objetos desconocidos:

- Se desconocen los elementos que conforman al conjunto resultante al aplicar la operación unión sobre los conjuntos ya dados

1.3 Relación entre los objetos:

- En primer lugar, podemos obtener un conjunto resultante de la concatenación de los dos conjuntos dados previamente. A este conjunto resultante

podemos ordenarlo de menor a mayor, y finalmente solo eliminar los elementos repetidos, para que todos los elementos del conjunto estén solo una vez. Después de realizar estas sencillas operaciones, obtendremos la unión de los conjuntos dados inicialmente

2. Diseño y prueba conceptual

función que concatena los dos conjuntos:

Entradas: 2 arreglos: Dos arreglos de enteros ($\mathbb{Z}^* \times \mathbb{Z}^*$)

Salidas: 1 arreglo de enteros resultante de la concatenación de dos arreglos

Relación:

$$\begin{array}{rcl} concatena : \mathbb{Z}^* \times \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ concatena(A, B) & \rightarrow & A + B \end{array}$$

ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} \mathbb{Z}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(C^*, n) & \rightarrow & m \end{array} \text{ talque} m = n-1 \wedge C_m = C_{n-1} \text{pero si} \forall^0_{i=n-1, i=i-1} C_i > C_m \rightarrow C_m = C_i \wedge m = i$$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo D^*

Relación:

$$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ ordena(C^*, n) & \rightarrow & D^* \end{array} \text{ tal que } \forall^0_{i=n-1, i=i-1} D_i = C_{max(C^*, i)} \rightarrow D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$$

Elmina los iguales de un arreglo:

Entradas: 1 arreglo (D^*) de enteros tal que $D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$(Se garantiza que el arreglo está ordenado)

Salidas: 1 arreglo (E^*) de enteros tal que $E_0 \neq E_1 \neq E_2 \neq \ldots \neq E_{n-1} \wedge E_0 \leq E_1 \leq E_2 \leq ... \leq E_{n-1}$

Relación:

$$\begin{array}{rcll} Eliminaiguales : \mathbb{Z}^* & \rightarrow & & \mathbb{Z}^* \\ Eliminaiguales(D) & \rightarrow E & \text{tal que } E + [D_i] \, si \, \forall^{|D|-2}_{i=0} D_i \neq D_{i+1} \vee i = |D| - 1 & \end{array}$$

3. Codificación:

> El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las funciones concatena, max, ordena y elimina_iguales se encuentran de la línea de código 23 a la 52. La parte del programa principal donde se llama a las funciones se encuentra de la línea 117 a la 120

Ejercicio 36
INTERSECCIÓN: Calcula en un arreglo la intersección de los conjuntos y la imprime.
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior
- En la operación de intersección, se imprimen en un mismo arreglo (conjunto) los elementos que ambos conjunto comparten, los elementos de este conjunto resultante no pueden estar repetidos y deben estar ordenados

1.2 Objetos desconocidos:

- Se desconocen los elementos que los dos arreglos dados comparten, es decir, se desconocen los elementos del conjunto resultante de la operación intersección

1.3 Relación entre los objetos:

- Ya que se garantiza que los dos conjuntos dados inicialmente no contienen elementos repetidos, para hacer la intersección podemos, en primer lugar, concatenar los dos conjuntos dados inicialmente y al conjunto resultante ordenarlo de menor a mayor. Posteriormente, podemos almacenar en otro arreglo los elementos que se repiten al menos una vez en el conjunto concatenado. Y finalmente, almacenar en un último arreglo solo una única vez los elementos que se han repetido en el arreglo concatenado
- Ya que se ha garantizado que los conjuntos dados inicialmente no contienen elementos repetidos, es correcto afirmar que los elementos de la intersección serán los elementos repetidos al menos una vez en el conjunto resultante de la concatenación
- El procedimiento descrito es más eficiente que comparar elemento por elemento si cada número del primer conjunto se repite en el segundo conjunto, pues en los pasos descritos se deben hacer menos operaciones

2. Diseño y prueba conceptual
función que concatena los dos conjuntos:

Entradas: 2 arreglos: Dos arreglos de enteros ($\mathbb{Z}^* \times \mathbb{Z}^*$)

Salidas: 1 arreglo de enteros resultante de la concatenación de dos arreglos

Relación:

$$\begin{array}{rcl} concatena : \mathbb{Z}^* \times \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ concatena(A, B) & \rightarrow & A + B \end{array}$$

ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} \mathbb{Z}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(C^*, n) & \rightarrow & m \end{array} \text{ talque} m = n-1 \wedge C_m = C_{n-1} \text{pero si} \forall^0_{i=n-1, i=i-1} C_i > C_m \rightarrow C_m = C_i \wedge m = i$$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo D^*

Relación:

$$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ ordena(C^*, n) & \rightarrow & D^* \end{array} \text{ tal que } \forall^0_{i=n-1, i=i-1} D_i = C_{max(C^*, i)} \rightarrow D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$$

Función que almacena los repetidos:

Entradas: 1 arreglo de enteros (D^*) tal que $D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$(Se garantiza que el arreglo está ordenado)

Salidas: 1 arreglo con los elementos que se han repetido al menos una vez

Relación:

$$\begin{array}{rcl} repetidos : \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ repetidos(D) & \rightarrow & E \text{ tal que } E + [D_i] \; si \; \forall^{|D|-2}_{i=0} D_i = D_{i+1} \end{array}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las

funciones concatena, max, ordena y elimina_iguales se encuentran de la línea de código 23 a la 43, la función repetidos se encuentra de la línea 53 a la 61. La parte del programa principal donde se llama a las funciones se encuentra de la línea 121 a la 124

Ejercicio 37
DIFERENCIA Calcula en un arreglo la diferencia del primero con el segundo y la imprime.
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior
- En la operación de diferencia, se imprimen los elementos que pertenecen a A y que no pertencen a B, es decir, $A - B = x : x \in A \wedge x \notin B$. Si tenemos la intersección entre A y B, la diferencia entre A y B sería igual a imprimir los elementos de A excepto los de la intersección entre A y B

1.2 Objetos desconocidos:

- Se desconocen los elementos del conjunto resultante de la diferencia entre el primer arreglo y el segundo arreglo

1.3 Relación entre los objetos:

- La diferencia entre un conjunto A y un conjunto B, será igual a imprimir los elementos de A excepto los que pertenecen a la intersección entre A y B. Si hallamos la intersección entre A y B con el algoritmo desarrollado en el punto anterior, esta intersección vendrá ordenada. Así que si ordenamos también el conjunto A, se puede hacer una serie de comparaciones para determinar cuáles son los elementos que pertencen a A y no pertencen a B.

2. Diseño y prueba conceptual
ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (A^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:

$$\begin{array}{ccc} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(A^*, n) & \rightarrow & m \end{array} \text{ talque} m = n-1 \wedge A_m = A_{n-1} \text{pero si} \forall^0_{i=n-1, i=i-1} A_i > A_m \rightarrow A_m = A_i \wedge m = i$$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (A^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo B^*

Relación:

$\mathbb{R}^* \times \mathbb{N} \quad \rightarrow \quad \mathbb{R}^*$
$ordena(A^*, n) \quad \rightarrow \quad B^*$ tal que $\forall_{i=n-1, i=i-1}^{0} B_i = A_{max(A^*,i)} \rightarrow$
$B_0 \leq B_1 \leq B_2 \leq ... \leq B_{n-1}$

Intersección de los conjuntos:

Haciendo uso de las funciones especificadas en el punto anterior (ejercicio 36), se obtiene la intersección de los conjuntos

Función que determina los elementos de la diferencia:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al conjunto A que ha sido ordenado, y el otro (el conjunto INT) al conjunto de la intersección entre A y B (los conjuntos dados inicialmente); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del conjunto de la intersección ($k \in \mathbb{N} : k \neq 0$)

Salidas: 1 arreglo

Relación:

Garantizar que A está ordenado, y hayar la intersección primero

$diferencia : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$

$$diferencia(A, INT, n, k) \rightarrow \begin{cases} [] & si \quad n = 0 \\ \text{C Tal que} \forall_{i=0}^{n-1} C_i = A_i & si \quad k = 0 \\ diferencia(A, INT, n-1, k-1) & si \quad A_{n-1} = INT_{k-1} \\ diferencia(A, INT, n, k-1) & \quad A_{n-1} < INT_{k-1} \\ C\text{tal que } C = diferencia(A, INT, n-1, k) + [A_{n-1}] & si \quad A_{n-1} > INT_{k-1} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las funciones max y ordena se encuentran de la línea de código 27 a la 43, la función que calcula la intersección se encuentra de la línea 53 a la 61. La función diferencia se encuentra de la línea de código 64 a la 78. La parte del programa principal donde se llama a las funciones se encuentra de la línea 125 a la 131

Ejercicio 38

DIFERENCIA SIMÉTRICA : Calcula en un arreglo la diferencia simétrica de los conjuntos y la imprime.

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior
- Se conoce que la diferencia simétrica entre dos conjuntos puede versse como $A \bigcup B - A \bigcap B$

1.2 Objetos desconocidos:

- Se desconocen los elementos del conjunto resultante al aplicar la diferencia simétrica entre los dos conjuntos dados inicialmente

1.3 Relación entre los objetos:

- La diferencia simétrica resulta de aplicar la operación de diferencia a la unión de los conjuntos con la intersección de estos. Para esto, se pueden utilizar las funciones descritas en el ejercicio 35 y 36

2. Diseño y prueba conceptual

Unión de los conjuntos:

Para hallar la unión entre los conjuntos se utilizan las funciones especificadas en el ejercicio 35

función que concatena los dos conjuntos:

Entradas: 2 arreglos: Dos arreglos de enteros ($\mathbb{Z}^* \times \mathbb{Z}^*$)

Salidas: 1 arreglo de enteros resultante de la concatenación de dos arreglos

Relación:

$$\begin{array}{rcl} concatena : \mathbb{Z}^* \times \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ concatena(A, B) & \rightarrow & A + B \end{array}$$

ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} \mathbb{Z}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(C^*, n) & \rightarrow & m \end{array} \text{ talque} m = n-1 \wedge C_m = C_{n-1} \text{pero si} \forall^0_{i=n-1, i=i-1} C_i > C_m \rightarrow C_m = C_i \wedge m = i$$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo D^*

Relación:

$$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ ordena(C^*, n) & \rightarrow & D^* \end{array} \text{ tal que } \forall^0_{i=n-1, i=i-1} D_i = C_{max(C^*, i)} \rightarrow D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$$

Elmina los iguales de un arreglo:

Entradas: 1 arreglo (D^*) de enteros tal que $D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$(Se garantiza que el arreglo está ordenado)

Salidas: 1 arreglo (E^*) de enteros tal que $E_0 \neq E_1 \neq E_2 \neq \ldots \neq E_{n-1} \wedge E_0 \leq E_1 \leq E_2 \leq ... \leq E_{n-1}$

Relación:

$$\begin{array}{rcl} Eliminaiguales : \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ Eliminaiguales(D) & \rightarrow E & \text{tal que } E + [D_i]\, si\ \forall_{i=0}^{|D|-2} D_i \neq D_{i+1} \vee i = |D| - 1 \end{array}$$

Intersección de los conjuntos

Para obtener la intersección entre los conjuntos se utilizan las funciones especificadas en el ejercicio 36

función que concatena los dos conjuntos:

Entradas: 2 arreglos: Dos arreglos de enteros ($\mathbb{Z}^* \times \mathbb{Z}^*$)

Salidas: 1 arreglo de enteros resultante de la concatenación de dos arreglos

Relación:

$$\begin{array}{rcl} concatena : \mathbb{Z}^* \times \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ concatena(A, B) & \rightarrow & A + B \end{array}$$

ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} \mathbb{Z}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(C^*, n) & \rightarrow & m \end{array} \text{ talque} m = n-1 \wedge C_m = C_{n-1} \text{pero si} \forall_{i=n-1, i=i-1}^{0} C_i > C_m \rightarrow C_m = C_i \wedge m = i$$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (C^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo D^*

Relación:

$$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ ordena(C^*, n) & \rightarrow & D^* \end{array} \text{ tal que } \forall_{i=n-1, i=i-1}^{0} D_i = C_{max(C^*, i)} \rightarrow D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$$

Función que almacena los repetidos:

Entradas: 1 arreglo de enteros (D^*) tal que $D_0 \leq D_1 \leq D_2 \leq ... \leq D_{n-1}$(Se garantiza que el arreglo está ordenado)

Salidas: 1 arreglo con los elementos que se han repetido al menos una vez

Relación:

$$\begin{array}{lcl} repetidos : \mathbb{Z}^* & \rightarrow & \mathbb{Z}^* \\ repetidos(D) & \rightarrow & E \text{ tal que } E + [D_i] \; si \; \forall_{i=0}^{|D|-2} D_i = D_{i+1} \end{array}$$

Función que calcula la diferencia:
Para hacer la fierencia entre la unión y la intersección se utilizan las funciones definidas en el punto 37

función que determina los elementos de la diferencia:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al conjunto U entre los conjuntos A y B, y el otro (el conjunto INT) al conjunto de la intersección entre A y B (los conjuntos dados inicialmente); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del conjunto de la intersección ($k \in \mathbb{N} : k \neq 0$)

Salidas: 1 arreglo

Relación:

Garantizar que A está ordenado, y hallar la intersección primero

$$diferencia : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Z}^*$$

$$diferencia(U, INT, n, k) \rightarrow \begin{cases} [] & si \quad n = 0 \\ \text{C Tal que} \forall_{i=0}^{n-1} C_i = U_i & si \quad k = 0 \\ diferencia(U, INT, n-1, k-1) & si \quad U_{n-1} = INT_{k-1} \\ diferencia(U, INT, n, k-1) & \quad U_{n-1} < INT_{k-1} \\ C \text{tal que } C = diferencia(U, INT, n-1, k) + [U_{n-1}] & si \quad U_{n-1} > INT_{k-1} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las funciones se encuentran de la línea de código 24 a la línea de código 78. La parte del programa principal donde se llama a las funciones se encuentra de la línea 132 a la 139

Ejercicio 39
PERTENECE: Lee un entero y determina si el elemento pertenece o no a cada uno de los conjuntos y lo imprime
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior
- Se conoce un elemento el cual se determinará si pertenece o no a cada conjunto

1.2 Objetos desconocidos:

- Se desconoce si el elemento dado pertenece a cada uno de los conjuntos

1.3 Relación entre los objetos:

- Si al comparar en cada uno de los elementos de un conjunto alguno de ellos es igual al elemento que se busca determinar como perteneciente, este último sí pertenecerá al conjunto
- Si el elemento pertenece a cada conjunto por separado, pertenecerá a los dos al mismo tiempo

2. Diseño y prueba conceptual
Pertenece a un conjunto:

Entradas: Dos variables: 1 arreglo (conjunto) A de enteros $\mathbb{Z}^*$; 1 variable ($x \in \mathbb{Z}$)

Salidas: 1 variable: Un booleano, True si $x \in A$ False si $x \notin A$

Relación:

$$conjuntos : \mathbb{Z}^* \times \mathbb{Z} \rightarrow \mathbb{B}$$

$$conjuntos(A, x) \rightarrow \begin{cases} True & si \quad \exists_{i=0}^{|A|-1} A_i = x \\ Fasle & \text{En otro caso} \end{cases}$$

Pertenece a los dos conjuntos;

Entradas: 2 variables de booleanos: $x, y \in \mathbb{B}$

Salidas: 1 cadena de caracteres

Relación:

$$pertenece : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{ASCII}^*$$

$$pertenece(x, y) \rightarrow \begin{cases} \text{"Pertenece a ambos conjuntos"} & si \quad x \wedge y = true \\ \text{"Pertenece solo al primer conjunto"} & si \quad x = True \\ \text{"Pertenece solo al segundo conjunto"} & si \quad y = True \\ \text{"No pertenece a ninguno"} & \text{En otro caso} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las funciones se encuentran de la línea de código 81 a la línea de código 97. La parte del programa principal donde se llama a las funciones se encuentra de la línea 140 a la 146

Ejercicio 40
CONTENIDO: Determina sí el primer conjunto esta contenido en el segundo y lo imprime.
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior
- Si un conjunto A está contenido en el conjunto B, todos los elementos de A pertenecen también a B

1.2 Objetos desconocidos:

- Se desconoce si el primer conjunto está contenido o no en el segundo

1.3 Relación entre los objetos:

- Si el conjunto A está contenido en el conjunto B, entonces la diferencia entre A y B será el conjunto vacío $A - B = \emptyset$. En caso contrario no estará contenido

2. Diseño y prueba conceptual
Diferencia entre los dos conjuntos:
Para hallar la diferencia entre los dos conjuntos se recurre a las funciones planteadas en el punto 37

ïndice del máximo elemento de un arreglo:

Entradas: 2 variables: Un arreglo (A^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 variable: Indice del elemento maximo del arreglo ($m \in \mathbb{N}$)

Relación:
$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{N} \\ \max(A^*, n) & \rightarrow & m \end{array}$ talque$m = n-1 \wedge A_m = A_{n-1}$pero si$\forall^0_{i=n-1, i=i-1} A_i > A_m \rightarrow A_m = A_i \wedge m = i$

orden de los elementos de un arreglo:

Entradas: 2 variables: Un arreglo (A^*)y la cantidad de elementos del arreglo ($n \in \mathbb{N} : n \neq 0$)

Salidas: 1 arreglo B^*

Relación:
$\begin{array}{rcl} \mathbb{R}^* \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ ordena(A^*, n) & \rightarrow & B^* \end{array}$ tal que $\forall^0_{i=n-1, i=i-1} B_i = A_{max(A^*, i)} \rightarrow B_0 \leq B_1 \leq B_2 \leq ... \leq B_{n-1}$

Función que determina los elementos de la diferencia:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al conjunto A que ha sido ordenado, y el otro al conjunto B también ordenado inicialmente; cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($k \in \mathbb{N} : k \neq 0$)

Salidas: 1 arreglo

Relación:

$$diferencia : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$$

$$diferencia(A, B, n, k) \rightarrow \begin{cases} [] & si \quad n = 0 \\ \text{C Tal que} \forall_{i=0}^{n-1} C_i = A_i & si \quad k = 0 \\ diferencia(A, B, n-1, k-1) & si \quad A_{n-1} = B_{k-1} \\ diferencia(A, B, n, k-1) & \quad A_{n-1} < B_{k-1} \\ C \text{tal que } C = diferencia(A, B, n-1, k) + [A_{n-1}] & si \quad A_{n-1} > B_{k-1} \end{cases}$$

Determina si el conjunto está contenido o no:

Entradas: 1 arreglo de enteros $\mathbb{Z}^*$

Salidas: 1 variable: Un valor booleano que indica si el primer conjunto está contenido en el segundo

Relación:

$$contenido : \mathbb{Z}^* \quad \rightarrow \quad \mathbb{B}$$

$$contenido(C) \quad \rightarrow \quad \begin{cases} True & si \quad |C| = 0 \\ False & \text{En otro caso} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». Las funciones max y ordena se encuentran de la línea de código 27 a la 43. La función diferencia se encuentra de la línea de código 64 a la 78., la función contenido se encuentra de la línea de código 99 a la 105.La parte del programa principal donde se llama a las funciones se encuentra de la línea 147 a la 154

Ejercicio 41

SALIR: Permite al usuario salir de la aplicación.

1. Análisis y especificación

1.1 Objetos conocidos:

- Ya que mediante un menú se deben escoger las operaciones especificadas en los puntos 35-40, para salir se debe pedir esta función especifica que terminará con las operaciones antes pedidas.

1.2 Objetos desconocidos:

- Se desconoce si el usuario quiere salir del menú presentado

1.3 Relación entre los objetos:

- Si el usuario desea salir, debe escoger esta función en el menú presentado presionando un número específico

2. Diseño y prueba conceptual
En el programa principal se presentarán como opciones las operaciones descritas en los puntos 35-40. Además tendrá la opción de salir

Entradas: 1 variable: $x \in \mathbb{Z}$la cual indica la operación que se requiere realizar

Salidas: El procedimiento que se ha pedido

Relación:

$$menú : \mathbb{Z} \rightarrow funciones$$

$$menu(x) \rightarrow \begin{cases} UNION & si \quad x = 1 \\ INTERSECCION & si \quad x = 2 \\ DIFERENCIA & si \quad x = 3 \\ DIFERENCIA\ SIMËTRICA & si \quad x = 4 \\ PERTENECE & si \quad x = 5 \\ CONTENIDO & si \quad x = 6 \\ SALIR & \text{En otro caso} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_CONJUNTOS_SimonRamos.py». El programa principal donde se llama a todas las funciones y se desarrolla el menú con las opciones se encuentra de la línea de código 106 a la 158.

Ejercicio 42
Suponga ahora que los elementos de tipo T se encuentran ordenados totalmente. La representación anterior se puede modificar de tal manera que las operaciones anteriores sean implementadas de manera más eficiente. La idea es mantener el conjunto de manera ordenada.
Desarrollar el programa anterior usando la representación modificada con las operaciones entre conjuntos optimizadas.

UNIÓN:

1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior y están completamente ordenados

- Al aplicar la operación de unión sobre estos conjuntos, los elementos del conjunto resultante no pueden estar repetidos y deben mantener el orden.

1.2 Objetos desconocidos:

- Se desconocen los elementos que conforman al conjunto resultante al aplicar la operación unión sobre los conjuntos ya dados

1.3 Relación entre los objetos:

- Ya que los conjuntos vienen ordenados podemos establecer el siguiente procedimiento para determinar la unión entre los conjuntos y mantener el orden: Mediante una función recursiva, comparar el último elemento del primer conjunto con el último elemento del segundo conjunto. Posteriormente pegar en otro arreglo el mayor elemento y volver a comparar con el elemento de la posición anterior del arreglo que contenía el elemento mayor y repetir el proceso. Si ambos elementos son iguales, pegar solo una vez dicho elemento en otro arreglo. Continuar hasta que la longitud de almenos uno termine, si al otro aún le quedan elementos, añadirlos al arreglo nuevo.

2. Diseño y prueba conceptual
función unión:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($k \in \mathbb{N} :\neq 0$)

Salidas: 1 arreglo

Relación:

$$union : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$$

$$union(A, B, n, k) \rightarrow \begin{cases} C \text{ Tal que} \forall_{i=0}^{k-1} C_i = B_i & si & n = 0 \\ C \text{Tal que} \forall_{i=0}^{n-1} C_i = A_i & si & k = 0 \\ C \text{tal que } C = union(A, B, n-1, k-1) + [A_{n-1}] & si & A_{n-1} = B_{k-1} \\ C \text{tal que } C = unio(A, B, n-1, k) + [A_{n-1}] & si & A_{n-1} > B_{k-1} \\ C \text{ tal que } C = union(A, B, n, k-1) + [B_{k-1}] & & \text{En otro caso} \end{cases}$$

INTERSECCION
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior y están completamente ordenados

- En la operación de intersección, se imprimen en un mismo arreglo (conjunto) los elementos que ambos conjunto comparten, los elementos de este conjunto resultante no pueden estar repetidos y deben estar ordenados

1.2 Objetos desconocidos:

- Se desconocen los elementos que los dos arreglos dados comparten, es decir, se desconocen los elementos del conjunto resultante de la operación intersección

1.3 Relación entre los objetos:

- Ya que se garantiza que los dos conjuntos dados inicialmente no contienen elementos repetidos y están ordenados, se pueden hacer una serie de comparaciones mediante una función recursiva para determinar los elementos de la intersección: Comparando el último elemento de los dos conjntos, si estos son iguales el elemento se añadirá a otro arreglo. En otro caso, se comparará el penúltimo elemento del arreglo con el elemento mayor con el último elemento con el arreglo del elemento menor. Se repetirá este proceso hasta que alguno de los dos conjuntos se quede sin elementos

2. Diseño y prueba conceptual
función intersección:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($m \in \mathbb{N} : m \neq 0$)

Salidas: 1 arreglo

Relación:

$$interseccion : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$$

$$interseccion(A, B, n, m) \rightarrow \begin{cases} [] & si \quad n = 0 \vee m = 0 \\ C\text{tal que } C = interseccion(A, B, C, n-1, m-1) + [A_{n-1}] & si \quad A_{n-1} = B_{m-1} \\ interseccion(A, B, C, n-1, m) & si \quad A_{n-1} > B_{m-1} \\ interseccion(A, B, C, n, m-1) & \text{En otro caso} \end{cases}$$

DIFERENCIA
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) de enteros, los cuales no tienen elementos repetidos en su interior y están completamente ordenados
- En la operación de diferencia, se imprimen los elementos que pertenecen a A y que no pertencen a B, es decir, $A - B = x : x \in A \wedge x \notin B$.

1.2 Objetos desconocidos:

- Se desconocen los elementos del conjunto resultante de la diferencia entre el primer arreglo y el segundo arreglo

1.3 Relación entre los objetos:

- La diferencia entre un conjunto A y un conjunto B, será igual a imprimir los elementos de A excepto los que pertenecen también a B. Para obtener la diferencia comparamos un elemento del conjunto A con un elemento del conjunto B, si elemento de A es mayor pegamos el elemento en un nuevo arreglo y ahora comparamos con el elemento anterior en A; si son iguales comparamos el anterior elemento de A con el elemento aneterior de B; si es menor comparamos con el elemento anterior de B. Repetimos el proceso hasta que alguno de los dos conjuntos no tenga elementos.

2. Diseño y prueba conceptual
Función que determina los elementos de la diferencia:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($m \in \mathbb{N} : m \neq 0$)

Salidas: 1 arreglo

Relación:

$$diferencia : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$$

$$diferencia(A, B, n, m) \rightarrow \begin{cases} [] & si \quad n = 0 \\ \text{C Tal que} \forall_{i=0}^{n-1} C_i = A_i & si \quad m = 0 \\ diferencia(A, B, n-1, m-1) & si \quad A_{n-1} = B_{m-1} \\ diferencia(A, B, n, m-1) & si \quad A_{n-1} < B_{m-1} \\ C \text{tal que } C = diferencia(A, B, n-1, m) + [A_{n-1}] & si \quad A_{n-1} > B_{m-1} \end{cases}$$

DIFERENCIA SIMÉTRICA
1. Análisis y especificación
1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior y están completamente ordenados
- Se conoce que la diferencia simétrica entre dos conjuntos puede versse como $A \bigcup B - A \bigcap B$

1.2 Objetos desconocidos:

- Se desconocen los elementos del conjunto resultante al aplicar la diferencia simétrica entre los dos conjuntos dados inicialmente

1.3 Relación entre los objetos:

- La diferencia simétrica resulta de aplicar la operación de diferencia a la unión de los conjuntos con la intersección de estos. Para esto, se pueden utilizar las funciones descritas en el ejercicio anteriores

2. Diseño y prueba conceptual

función unión:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($m \in \mathbb{N} : m \neq 0$)

Salidas: 1 arreglo

Relación:

$union : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$

$$union(A,B,n,k) \rightarrow \begin{cases} C \text{ Tal que} \forall_{i=0}^{k-1} C_i = B_i & si & n = 0 \\ C \text{Tal que} \forall_{i=0}^{n-1} C_i = A_i & si & k = 0 \\ C \text{tal que } C = union(A,B,n-1,k-1) + [A_{n-1}] & si & A_{n-1} = B_{k-1} \\ C \text{tal que } C = unio(A,B,n-1,k) + [A_{n-1}] & si & A_{n-1} > B_{k-1} \\ C \text{ tal que } C = union(A,B,n,k-1) + [B_{k-1}] & & \text{En otro caso} \end{cases}$$

función intersección:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($m \in \mathbb{N} : m \neq 0$)

Salidas: 1 arreglo

Relación:

$interseccion : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$

$$interseccion(A,B,n,m) \rightarrow \begin{cases} [] & si & n = 0 \vee m = 0 \\ C \text{tal que } C = interseccion(A,B,C,n-1,m-1) + [A_{n-1}] & si & A_{n-1} = B_{m-1} \\ interseccion(A,B,C,n-1,m) & si & A_{n-1} > B_{m-1} \\ interseccion(A,B,C,n,m-1) & & \text{En otro caso} \end{cases}$$

función que determina los elementos de la diferencia:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al conjunto U entre los conjuntos A y B, y el otro (el conjunto INT) al conjunto de la intersección entre A y B (los conjuntos dados inicialmente); cantidad de elementos de la unión ($l \in \mathbb{N} : l \neq 0$), cantidad de elementos del conjunto de la intersección ($k \in \mathbb{N} : k \neq 0$)

Salidas: 1 arreglo

Relación:

$diferencia : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{Z}^*$

$$diferencia(U, INT, l, k) \rightarrow \begin{cases} [] & si & l = 0 \\ \text{C Tal que} \forall_{i=0}^{l-1} C_i = INT_i & si & k = 0 \\ diferencia(U, INT, l-1, k-1) & si & U_{l-1} = INT_{k-1} \\ diferencia(U, INT, l, k-1) & & U_{l-1} < INT_{k-1} \\ C \text{tal que } C = diferencia(U, INT, l-1, k) + [U_{l-1}] & si & U_{l-1} > INT_{k-1} \end{cases}$$

PERTENECE

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior y están completamente ordenados
- Se conoce un elemento el cual se determinará si pertenece o no a cada conjunto

1.2 Objetos desconocidos:

- Se desconoce si el elemento dado pertenece a cada uno de los conjuntos

1.3 Relación entre los objetos:

- Si al comparar en cada uno de los elementos de un conjunto alguno de ellos es igual al elemento que se busca determinar como perteneciente, este último sí pertenecerá al conjunto
- Si el elemento pertenece a cada conjunto por separado, pertenecerá a los dos al mismo tiempo

2. Diseño y prueba conceptual

Pertenece a un conjunto:

Entradas: Dos variables: 1 arreglo (conjunto) A de enteros $\mathbb{Z}^*$; 1 variable ($x \in \mathbb{Z}$)

Salidas: 1 variable: Un booleano, True si $x \in A$ False si $x \notin A$

Relación:

$$conjuntos : \mathbb{Z}^* \times \mathbb{Z} \rightarrow \mathbb{B}$$

$$conjuntos(A, x) \rightarrow \begin{cases} True & si & \exists_{i=0}^{|A|-1} A_i = x \\ Fasle & & \text{En otro caso} \end{cases}$$

Pertenece a los dos conjuntos:

Entradas: 2 variables de booleanos: $x, y \in \mathbb{B}$

Salidas: 1 cadena de caracteres

Relación:

$$pertenece : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{ASCII}^*$$

$$pertenece(x, y) \rightarrow \begin{cases} \text{"Pertenece a ambos conjuntos"} & si & x \wedge y = true \\ \text{"Pertenece solo al primer conjunto"} & si & x = True \\ \text{"Pertenece solo al segundo conjunto"} & si & y = True \\ \text{"No pertenece a ninguno"} & & \text{En otro caso} \end{cases}$$

CONTENIDO

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conocen dos arreglos (conjuntos) los cuales no tienen elementos repetidos en su interior y están completamente ordenados
- Si un conjunto A está contenido en el conjunto B, todos los elementos de A pertenecen también a B

1.2 Objetos desconocidos:

- Se desconoce si el primer conjunto está contenido o no en el segundo

1.3 Relación entre los objetos:

- Si el conjunto A está contenido en el conjunto B, entonces la intersección entre A y B será igual al conjunto A

2. Diseño y prueba conceptual

función intersección:

Entradas: 4 variables: 2 arreglos de enteros ordenados, uno de ellos corresponde al primer conjunto dado A ($\mathbb{Z}^*$), y el otro al segundo conjunto dado B ($\mathbb{Z}^*$); cantidad de elementos del primer conjunto n ($n \in \mathbb{N} : n \neq 0$), cantidad de elementos del segundo conjunto ($m \in \mathbb{N} : m \neq 0$)

Salidas: 1 arreglo

Relación:

$$interseccion : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Z}^*$$

$$interseccion(A, B, n, m) \rightarrow \begin{cases} [] & si & n = 0 \vee m = 0 \\ C\text{tal que } C = interseccion(A, B, C, n-1, m-1) + [A_{n-1}] & si & A_{n-1} = B_{m-1} \\ interseccion(A, B, C, n-1, m) & si & A_{n-1} > B_{m-1} \\ interseccion(A, B, C, n, m-1) & & \text{En otro caso} \end{cases}$$

Determina si el conjunto está contenido o no:

Entradas: 2 arreglos de enteros $\mathbb{Z}^*$

Salidas: 1 variable: Un valor booleano que indica si el primer conjunto está contenido en el segundo

Relación:

$$contenido : \mathbb{Z}^* \times \mathbb{Z}^* \rightarrow \mathbb{B}$$

$$contenido(A, C) \rightarrow \begin{cases} True & si & A = C \\ False & & \text{En otro caso} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_42_SimonRamos.py».

3 Polinomios como arreglos

Un polinomio de grado n, como P(x) = $a_n x^x + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0 x^0$ se puede representar mediante un arreglo de reales de la siguiente manera: $(a_0, a_1, ..., a_{n-1}, a_n)$.

Usando esta representación hacer un programa que le permita al usuario leer dos polinomios y escoger mediante un menú, una de las siguientes operaciones sobre dichos polinomios:

Funciones que hacen las lecturas de los polinomios:

Función que parte una cadena cuando encuentre un + o -:

Entrada: Una cadena de tamaño n ($p \in \mathbb{ASCII}^n$)

Salidas: Una lista de cadenas ($s \in (\mathbb{ASCII}^*)^*$)

Relación:

$$\begin{array}{lcl} partir : \mathbb{ASCII}^n & \rightarrow & (\mathbb{ASCII}^*)^* \\ \quad partir(p) & \rightarrow & s \text{ tal que } \forall_{i=0}^{n-1} s_{kj} = p_i \end{array}$$

Donde k =número de $+, -$ desde la posición 0 hasta la posición i y $j = i$−la posición del último $+, -$ antes de la posición i o 0 si no había

Función que toma los coeficientes, es decir, cadenas antes de la x:

Entrada: Una cadena de tamaño n ($t \in \mathbb{ASCII}^n$)

Salidas: Un número que equivale al coeficiente del termino ($m \in \mathbb{R}$)

Relación:

$$\begin{array}{lcl} coeficiente : \mathbb{ASCII}^n & \rightarrow & \mathbb{R} \\ \quad coeficiente(t) & \rightarrow & float(m) \text{ tal que } \forall_{i=0}^{i<n \wedge t_i \neq x} m_i = t_i \end{array}$$

Función que crea una lista con todos los coeficientes:

Entrada: Una lista de cadenas de caracteres de tamño n ($ts \in (\mathbb{ASCII}^*)^n$)

Salida: Un arreglo s de reales con todos los coeficientes ($s \in \mathbb{R}^n$)

Relación:

$$\begin{array}{lcl} coeficientes : \mathbb{ASCII}^n & \rightarrow & \mathbb{R}^n \\ \quad coeficientes(ts) & \rightarrow & s \text{ tal que } \forall_{i=0}^{n-1} s_i = coeficiente(ts_i) \end{array}$$

Función que toma los exponentes, es decir, cadenas después de la x:

Entrada: Una cadena de caracteres de tamaño n ($t \in \mathbb{ASCII}^n$)

Salidas: Un natural, que equivale al exponente del término ($s \in \mathbb{N}$)

Relación:

$$\begin{array}{rcl} exponente : \mathbb{ASCII}^n & \rightarrow & \mathbb{N} \\ exponente(t) & \rightarrow & \begin{cases} 0 & si \quad \forall_{i=0}^{n-1} t_i \neq "x" \\ 1 & si \quad t_{n-1} = "x" \\ int(s) \text{ tal que } \forall_{i=k+2}^{n-1} s_{i-k-2} = t_i & \text{En otro caso} \end{cases} \end{array}$$

Donde k es la posición de "x"

Función que crea una lista con los exponentes

Entrada: Una lista de cadenas de caracteres de tamño n ($ts \in (\mathbb{ASCII}^*)^n$)

Salida: Un arreglo s de naturales con todos los coeficientes ($s \in \mathbb{N}^n$)

Relación:

$$\begin{array}{rcl} exponentes : \mathbb{ASCII}^n & \rightarrow & \mathbb{N}^n \\ exponentes(ts) & \rightarrow & s \text{ tal que } \forall_{i=0}^{n-1} s_i = exponente(ts_i) \end{array}$$

Función que detecta el grado del polinomio (máximo elemento del arreglo de exponentes):

Entradas: Un arreglo A de enteros de tamaño n ($A \in \mathbb{Z}^n$)

Salidas: 1 variable: el elemento más grande del arreglo A ($A_{max} \in \mathbb{Z}$)

Relación:

$$\begin{array}{rcl} \mathbb{Z}^* & \rightarrow & \mathbb{Z} \\ \text{maximo1(A)} & \rightarrow & A_m \text{ talque} A_m = A_{n-1} \text{pero si} \forall_{i=n-1, i=i-1}^{0} A_i > A_m \rightarrow A_m = A_i \end{array}$$

Función que crea el polinomio:

Entradas: Dos arreglos: Un arreglo de reales que se corresponde con los coeficientes del polinomio ($a \in \mathbb{R}^m$), un arreglo de naturales que se corresponde con el arreglo de exponentes ($e \in \mathbb{N}^m$). Ambos arreglos son de tamaño m

Salidas: Un arreglo de reales que se corresponde con la representación del polinomio ($p \in \mathbb{R}^{n+1}$)

Relación:

$$\begin{array}{rcl} creapolinomio : \mathbb{R}^m \times \mathbb{N}^m & \rightarrow & \mathbb{R}^{n+1} \\ creapolinomio(a, e) & \rightarrow & p \text{ tal que } \forall_{i=0}^{m} p_{e_i} = a_i \wedge p_i = 0 \text{En otro caso} \end{array}$$

Función que integra las anteriores funciones paea construir el polinomio:

Entradas: Una cadena de caracteres que representa un polinomio ($s \in \mathbb{ASCII}^*$). El polinomio debe escribirse tomando el caracter "^" como simbolo que antecede al exponente, escribir todo el polinomio sin espacios. Un ejemplo de como debe escribirse es: "3x^2+2x+1".

Salidas: Un arreglo de reales que representa al polinomio escrito

Relación:

$$\begin{array}{rcl} polinomio : \mathbb{ASCII}^* & \rightarrow & \mathbb{R}^* \\ polinomio(s) & \rightarrow & crearpolinomio(coeficientes(z), exponentes(z)) \end{array}$$

Donde $z = partir(s)$

Función que da la salida de las operaciones:

Entradas: Un arreglo que se corresponde con el polinomio resultante de las operaciones ($A \in \mathbb{R}^*$)

Salidas: Una cadena de caracteres que se corresponde con el polinomio

Relación:

$$mostrarpoli : \mathbb{R}^* \rightarrow \mathbb{ASCII}^*$$

$$mostrarpoli(A) \rightarrow \begin{cases} \text{"0"} & si \quad |A| = 0 \\ c \text{ tal que "}A_0\text{"} + \forall_{i=0}^{|A|-1} \begin{cases} \text{"+}A_i\text{x\^{}i"} & si\ A_i \geq 0 \\ \text{" }A_i\text{x\^{}i"} & \text{En otro caso} \end{cases} & \text{En otro caso} \end{cases}$$

3. Codificación:

Las funciones de lectura se encuentran en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». Desde la línea de código 1 hasta la 72. Y la de mostrarpoli se encuentra de la línea 165 a la 176

Ejercicio 43

EVALUAR: Lee un real e imprime la evaluación de los dos polinomios en dicho dato.

1. Análisis y especificación

1.1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Se sabe que el exponente de cada término es igual a la posición del término en el arreglo
- Al evaluar un polinomio en un valor dado, se reemplaza este valor por la "x" en el polinomio

1.2 Objetos desconocidos:

- Se desconoce el resultado de evaluar cada polinomio en el valor dado por el usuario

1.3 Relación entre los objetos:

- Al reemplazar, el valor dado se multiplica por cada elemento del arreglo, no sin antes este número dado ser elevado a una potencia (la potencia se corresponde con el indice del elemento por el cual se va a multiplicar). Finalmente, cada resultado de estas multiplicaciones es sumado, de esta forma se obtiene la observación

2. Diseño y prueba conceptual
Función que hace las multiplicaciones

> ***Nota:*** Esta función es la misma para cada polinomio, por lo cual debe ejecutarse dos veces
>
> **Entradas:** 2 variables: 1 arreglo de reales A ($\mathbb{R}^*$) y un número real x en el cual se evaluara ($x \in \mathbb{R}$)
>
> **Salidas:** 1 variable: Un real que corresponde con la evaluación del polinomio $y \in \mathbb{R}$

$$\begin{array}{rcl} evalua : \mathbb{R}^* \times \mathbb{R} & \rightarrow & \mathbb{R} \\ evalua(A, x) & \rightarrow & \sum_{i=0}^{|A|-1} (A[i] * x^i) \end{array}$$

3. Codificación:

> El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». La función se encuentra de la línea 75 a la 80 y el programa principal donde se hace la lectura se encuentra de la líena 187 a la 195

Ejercicio 44
SUMAR: Calcula el polinomio suma y lo imprime.
1. Anáñisis y especificación
1,1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Al sumar polinomios, solo se pueden sumar los términos con el mismo grado. Según la representación dada, solo se podrían sumar los elementos encontrados en la misma posición

1.2 Objetos desconocidos:

- Se desconocen los coeficientes del polinomio resultante de la suma

1.3 Relación entre los objetos:

- Ya que solo se sumarán los elementos que están en la misma posición (Porque compartirían el mismo exponente), podemos decir que al polinomio de mayor grado le sumaremos los elementos del polinomio de menor grado. Si garantizamos que el primer argumento de la función es el polinomio de mayor o giual grado, se sumarían todos los elementos del segundo polinomio sobre los del primero

2. Diseño y prueba conceptual
Suma de los polinomios

Entradas: 2 arreglos de reales A,B ($\mathbb{R}^*, \mathbb{R}^*$), tal que A es de mayor o igual grado que B ($A\partial \geq B\partial$)

Salidas: 1 arreglo con los coeficientes de la suma entre A y B

Relación:

$$\begin{array}{rcl} suma : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ suma(A,B) & \rightarrow & \begin{cases} suma(B,A) & si \quad |A| < |B| \\ \forall_{i=0}^{|B|-1} A_i = A_i + B_i & \text{en otro caso} \end{cases} \end{array}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». La función se encuentra de la línea 83 a la 90 y el programa principal donde se hace la lectura se encuentra de la líena 196 a la 202

Ejercicio 45
RESTA: Calcula el polinomio resta y lo imprime.
1,1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Al restar polinomios, solo se pueden restar los términos con el mismo grado. Según la representación dada, solo se podrían restar los elementos encontrados en la misma posición

1.2 Objetos desconocidos:

- Se desconocen los coeficientes del polinomio resultante de la resta

1.3 Relación entre los objetos:

- Podemos ver a la resta $P(x) - Q(x)$ como una suma en la que todos los términos del polinomio $Q(x)$ son multiplicados por -1. Es decir, tenemos $P(x) + (-Q(x))$. Por ende, podemos crear una función que cambie de signo a todos los elementos de Q y reutilizar el la función de suma

2. Diseño y prueba conceptual
Función cambio de signo:

Entradas: 1 arreglo de reales B ($\mathbb{R}^*$)

Salidas: 1 arreglo de reales ($\mathbb{R}^*$)

$$\begin{array}{rcl} cambiasigno : \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ cambiasigno(B) & \rightarrow & \forall_{i=0}^{|B|-1} B_i = -B_i \end{array}$$

Suma de los polinomios

Entradas: 2 arreglos de reales A,B $(\mathbb{R}^*, \mathbb{R}^*)$, tal que A es de mayor o igual grado que B $(A\partial \geq B\partial)$

Salidas: 1 arreglo con los coeficientes de la suma entre A y B

Relación:

$$\begin{array}{rcl} suma : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ suma(A,B) & \rightarrow & \begin{cases} suma(B,A) & si \quad |A| < |B| \\ \forall_{i=0}^{|B|-1} A_i = A_i + B_i & \text{en otro caso} \end{cases} \end{array}$$

resta de polinomios:

Entradas: 2 arreglos de reales A,B $(\mathbb{R}^*, \mathbb{R}^*)$, tal que A es de mayor o igual grado que B $(A\partial \geq B\partial)$

Salidas: 1 arreglo con los coeficientes de la resta entre A y B

Relación:

$$\begin{array}{rcl} resta : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ resta(A,B) & \rightarrow & suma(A, cambiasigno(B)) \end{array}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». La función suma, cambiasigno y resta se encuentran de la línea 83 a la 99 y el programa principal donde se hace la lectura se encuentra de la líena 203 a la 209

Ejecicio 46

MULTIPLICAR: Calcula el polinomio multiplicación y lo imprime.

1,1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Al multiplicar polinomios se aplica la propiedad distribibutiva

1.2 Objetos desconocidos:

- Se desconocen los coeficientes del polinomio resultante de la multiplicación

1.3 Relación entre los objetos:

- Al multiplicar dos polinomos se aplica la propiedad distributiva y porteriormente se simplifican los términos semejantes. Si tuvieramos dos polinomios $A = (x^0 + x^1 + x^2 + x^3)$ y $B = (x^0 + x^1 + x^2 + x^3 + x^4 + x^5)$ la multplicación $A * B$ se obtendría de multiplicar cada elemento de A por cada elemento de B y posteriormente simplificar terminos semejantes. También sabemos que al multiplicar los exponentes que acompalan a las x se suman

- Siguiendo lo anterior y teniendo en cuenta que la posicion del coeficiente en el arreglo refleja su exponente, tendríamos la siguiente representación matricial para la multiplicacion $A * B$

$$\begin{array}{cccccccccc} x^0*x^0 & x^0*x^1 & x^0*x^2 & x^0*x^3 & x^0*x^4 & x^0*x^5 & 0 & 0 & 0 \\ 0 & x^1*x^0 & x^1*x^1 & x^1*x^2 & x^1*x^3 & x^1*x^4 & x^1*x^5 & 0 & 0 \\ 0 & 0 & x^2*x^0 & x^2*x^1 & x^2*x^2 & x^2*x^3 & x^2*x^4 & x^2*x^5 & 0 \\ 0 & 0 & 0 & x^3*x^0 & x^3*x^1 & x^3*x^2 & x^3*x^3 & x^3*x^4 & x^3*x^5 \end{array}$$

- Si sumamos las columnas de la anterior matriz, tendríamos el resultado de la multiplicacion de los polinomios $A * B$

2. Diseño y prueba conceptual

Función que crea las filas de la matriz:

Entradas: 4 variables: 2 arreglos (polinomios) A y B; el grado de A ($n \in \mathbb{N}$) y el grado de B ($m \in \mathbb{N}$)

Salidas: Un arreglo $\mathbb{R}^*$

Relación:

$$filas : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{R}^*$$

$$filas(A, B, n, m) \rightarrow \begin{cases} C \text{ tal que } \forall_{i=0}^{n-1} C_i = 0 & si \quad m = -1 \\ C \text{ tal que } C = filas(A, B, n, m-1) + [A_n * B_m] & \text{En otro caso} \end{cases}$$

Función que crea a la matriz

Entradas: 4 variables: 2 arreglos (polinomios) A y B; el grado de A ($n \in \mathbb{N}$) y el grado de B ($m \in \mathbb{N}$) **Nota:** Se debe garantizar que el primer argumento (El polinomio A) sea el de menor o igual grado

Salidas: Un arreglo de listas $\mathbb{R}^*$

Relación:

$$matriz : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{R}^*$$

$$matriz(A, B, n, m) \quad \rightarrow \quad \begin{cases} matriz(B, A, m, n) & si \quad |A| > |B| \\ [] & si \quad n = -1 \\ matriz(A, B, n-1, m) + [filas \quad (A, B, n, m)] & \text{En otro caso} \end{cases}$$

Para sumar las filas, se necesita utilizar la función especificada anteriormente para la suma de polinomios

Suma de los polinomios

Entradas: 2 arreglos de reales A,B ($\mathbb{R}^*, \mathbb{R}^*$), tal que A es de mayor o igual grado que B ($A\partial \geq B\partial$)

Salidas: 1 arreglo con los coeficientes de la suma entre A y B

Relación:

$$suma : \mathbb{Z}^* \times \mathbb{Z}^* \quad \rightarrow \quad \mathbb{Z}^*$$

$$suma(A, B) \quad \rightarrow \quad \begin{cases} suma(B, A) & si \quad |A| < |B| \\ \forall_{i=0}^{|B|-1} A_i = A_i + B_i & \text{en otro caso} \end{cases}$$

Función que suma las filas para calcular el producto:

Entradas: 2 variable: Un arreglo de arreglo (matriz) de reales; Longitud de este arreglo ($k \in \mathbb{N}$)

Salidas: Un arreglo de reales:

Relación:

$$\begin{aligned} producto : \mathbb{R}^* \times \mathbb{N} &\rightarrow \mathbb{R}^* \\ producto(C,k) &\rightarrow \begin{cases} [] & si \quad k = 0 \\ suma(C_{k-1}, producto(C,k-1)) & \text{En otro caso} \end{cases} \end{aligned}$$

funcion producto:

Entradas: 4 variables: 2 arreglos (polinomios) A y B; el grado de A ($n \in \mathbb{N}$) y el grado de B ($m \in \mathbb{N}$)

Salidas: 1 arreglo correspondiente al producto

Relación

$$\begin{aligned} multi : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \times \mathbb{N} &\rightarrow \mathbb{R}^* \\ multi(A,B,n,m) &\rightarrow producto(matriz(A,B,n,m), len(matriz(A,B,n,m)) \end{aligned}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». La función suma se encuentran de la línea 83 a la 90, las demás funciones se encuentran de la línea 102 hasta la línea 127 y el programa principal donde se hace la lectura se encuentra de la líena 210 a la 218

Ejercicio 47

DIVIDIR: Calcula el polinomio división del primer polinomio por el segundo y lo imprime.

1,1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Para dividir polinomios, el dividendo debe tener mayor o igual grado que el divisor. en caso contrario, la división no se podrá realizar

1.2 Objetos desconocidos:

- Se desconocen los coeficientes del polinomio resultante de la división

1.3 Relación entre los objetos:

- Para dividir dos polinomos, primero se dividen el término de grado mayor del dividendo por el término de grado mayor del divisor. Una vez se tiene el resultado de la división, este se multiplica por todos los términos del divisor y el resultado de esta multiplicación se resta con el dividendo. Al resultado de esta resta se le aplican los pasos anteriores, todo se repite hasta que se tenga un polinomio de grado menor al divisor.

- Para determinar el término resultado de la división entre los términos de grado mayor del dividendo y del divisor, primero se dividen los coeficientes de estos y a la lista se le añaden tantos ceros como indica la diferencia entre los grados. Al final, el resultado de la división será la suma de todos estos términos

2. Diseño y prueba conceptual
Función que calcula el término:

Entradas: 4 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor; grado del polinomio A ($n \in \mathbb{N}$) y el grado del polinomio B ($m \in \mathbb{N}$))

Salidas: 1 arreglo de reales

Relación:

$$\begin{array}{rcl} termino : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ termino(A, B, n, m) & \rightarrow & \begin{cases} [] & si \quad n < m \\ C \text{ tal que } C = [\forall_{i=0}^{(n-m)-1}] + [\frac{A_n}{B_m}] & \text{En otro caso} \end{cases} \end{array}$$

Función que resta las multiplicaciones y crea residuos:

Entradas: 2 arreglos de reales A y B

Salidas: 1 arreglo de reales

Relación:

$$\begin{array}{rcl} residuos : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ residuos(A, B) & \rightarrow & C \text{ tal que } \forall_{i=0}^{|A|-2} C_i = A_i - B_i \end{array}$$

Función que hace la operación de multiplicar y crear los residuos:

Entradas: 2 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor **Nota:** En esta función se utiliza la función multiplicación especificada en el ejercicio 46

Salidas: 1 arreglo de reales

Relación:

$$nuevodividendo : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$$

$nuevodividendo(A, B) \rightarrow residuos(A, multiplica(B, termino(A, B, |A|-1, |B|-1), |B|-1, |A|-|B|))$

Función que hace la división:

Entradas: 2 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor **Nota:** En esta función se utiliza la función suma especificada en el ejercicio 44

Salidas: 1 arreglo de reales

Relación:

$$division : \mathbb{R}^* \times \mathbb{R}^* \quad \rightarrow \quad \mathbb{R}^*$$

$$division(A,B) \rightarrow \begin{cases} [] & si \quad |A| < |B| \\ suma(termino(A,B,|A|-1,|B|-1), division(nuevodividendo(A,B),B)) & \text{En otro caso} \end{cases}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». Las funciones se encuentran de la línea 130 hasta la línea 154 y el programa principal donde se hace la lectura se encuentra de la líena 219 a la 225

Ejercicio 48

RESIDUO: Calcula el polinomio residuo de la divisi´on del primero por el segundo y lo imprime.

1,1 Objetos conocidos:

- Se conoce el grado de cada polinomio y, por ende, el número de términos de cada polinomio. Además, se conocen los coeficientes de cada término
- Para hallar el residuo, primero hay que dividir los polinomios, para esto el dividendo debe tener mayor o igual grado que el divisor. en caso contrario, el residuo será el propio dividendo

1.2 Objetos desconocidos:

- Se desconocen los coeficientes del polinomio resultante del residuo de la división

1.3 Relación entre los objetos:

- Para dividir dos polinomos, primero se dividen el término de grado mayor del dividendo por el término de grado mayor del divisor. Una vez se tiene el resultado de la división, este se multiplica por todos los términos del divisor y el resultado de esta multiplicación se resta con el dividendo. Al resultado de esta resta se le aplican los pasos anteriores, todo se repite hasta que se tenga un polinomio de grado menor al divisor. Para determinar el término resultado de la división entre los términos de grado mayor del dividendo y del divisor, primero se dividen los coeficientes de estos y a la lista se le añaden tantos ceros como indica la diferencia entre los grados.
- Al final, el residuo será aquel nuevo dividendo que se genera que sea de menor grado al divisor, en caso de que la división sea exacta, el residuo será cero (o un polinomio vacío)

2. Diseño y prueba conceptual
Función que calcula el término:

Entradas: 4 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor; grado del polinomio A ($n \in \mathbb{N}$) y el grado del polinomio B ($m \in \mathbb{N}$))

Salidas: 1 arreglo de reales

Relación:

$$\begin{array}{rcl} termino : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \times \mathbb{N} & \rightarrow & \mathbb{R}^* \\ termino(A, B, n, m) & \rightarrow & \begin{cases} [] & si \quad n < m \\ C \text{ tal que } C = [\forall_{i=0}^{(n-m)-1}] + [\frac{A_n}{B_m}] & \text{En otro caso} \end{cases} \end{array}$$

Función que resta las multiplicaciones y crea residuos:

Entradas: 2 arreglos de reales A y B

Salidas: 1 arreglo de reales

Relación:

$$\begin{array}{rcl} residuos : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ residuos(A, B) & \rightarrow & C \text{ tal que } \forall_{i=0}^{|A|-2} C_i = A_i - B_i \end{array}$$

Función que hace la operación de multiplicar y crear los residuos:

Entradas: 2 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor **Nota:** En esta función se utiliza la función multiplicación especificada en el ejercicio 46

Salidas: 1 arreglo de reales

Relación:

$$nuevodividendo : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$$

$nuevodividendo(A, B) \rightarrow residuos(A, multiplica(B, termino(A, B, |A|-1, |B|-1), |B| - 1, |A| - |B|))$

Función que retorna el residuo de la división:

Entradas: 2 variables: 2 arreglos de reales los cuales serán los polinomios a dividir, el arreglo A es el dividendo y el arreglo B el divisor **Nota:** En esta función se utiliza la función suma especificada en el ejercicio 44

Salidas: 1 arreglo de reales

Relación:

$$\begin{array}{rcl} residuodiv : \mathbb{R}^* \times \mathbb{R}^* & \rightarrow & \mathbb{R}^* \\ residuodiv(A, B) & \rightarrow & \begin{cases} A & si \quad |A| < |B| \\ residuodiv(nuevodividendo(A, B), B)) & \text{En otro caso} \end{cases} \end{array}$$

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». Las funciones termino, residuos y nuevodividendo se encuentran de la línea 130 hasta la línea 148, la función residuodiv se encuentra de la línea 157 a la 162 y el programa principal donde se hace la lectura se encuentra de la líena 226 a la 232

Ejercicio 49

SALIR: Permite al usuario salir de la aplicación.

1. Análisis y especificación

1.1 Objetos conocidos:

- Ya que mediante un menú se deben escoger las operaciones especificadas en los puntos 43-48, para salir se debe pedir esta función especifica que terminará con las operaciones antes pedidas.

1.2 Objetos desconocidos:

- Se desconoce si el usuario quiere salir del menú presentado

1.3 Relación entre los objetos:

- Si el usuario desea salir, debe escoger esta función en el menú presentado presionando un número específico

2. Diseño y prueba conceptual

En el programa principal se presentarán como opciones las operaciones descritas en los puntos 43-48. Además tendrá la opción de salir

Entradas: 1 variable: $x \in \mathbb{Z}$la cual indica la operación que se requiere realizar

Salidas: El procedimiento que se ha pedido

Relación:

$$menú : \mathbb{Z} \rightarrow funciones$$

$$menu(x) \rightarrow \begin{cases} EVALUAR & si \quad x = 1 \\ SUMAR & si \quad x = 2 \\ RESTA & si \quad x = 3 \\ MULTIPLICAR & si \quad x = 4 \\ DIVIDIR & si \quad x = 5 \\ RESIDUO & si \quad x = 6 \\ SALIR & \text{En otro caso} \end{cases}$$

3. Codificación

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicios_POLINOMIOS_SimonRamos.py». El programa principal donde se llama a todas las funciones y se desarrolla el menú con las opciones se encuentra de la línea de código 177 a la 235.

4 MATRICES

Ejercicio 50
Desarrollar un algoritmo que permita sumar dos matrices de números reales (enteros).

1. Análisis y especificación:
1.1 Objetos conocidos:

- Se conocen dos matrices, las cuales deben tener las mismas dimensiones, es decir, deben tener el mismo número de columnas y de filas
- La suma de las matrices se hace sumando los elementos que están en la misma posicón
- Las matrices se representan como una lista de listas

1.2 Objetos desconocidos:

- Se desconoce la matriz resultante de la suma de las dos matrices dadas por el usuario

1.3 Relación entre los objetos:

- Para la lectura de las matrices, se hará mediante una cadena de texto dada por el usuario. Ya que las matrices son listas de listas, si tenemos la matriz (lista) A^*, entonces A_0será también una lista. Donde A_0 (y en general cualquier A_n) representaría las filas de la matriz, y los elementos con la misma posición en los A_n representarían las columnas. En este sentido, la cadena de texto correspondendiente a una matriz vendría de la forma "2 3 4, 5 6 7, 8 9 1". Donde la "," separa a las filas entre sí (las listas de la lista de listas) y los espacios entre los números separa a los números que conformarían a las columnas
- Para la suma de dos matrices, se suman los elementos que comparten la misma posición en ambas matrices. Para que esta operación sea posible, las dos matrices deben tener igual número de filas y de columnas

2. Diseño y prueba conceptual
Funciones que hacen la lectura de la matriz:

> ***Función que separa por una "," y crea las filas:***
>
> **Entradas:** Una cadena de texto de longitud n ($s \in \mathbb{ASCII}^n$) con la representación de la matriz, esta cadena de texto debe venir de la forma "2 3 4, 5 6 7, 8 9 1"
>
> **Salidas:** Un arreglo de cadenas de caracteres ($C \in (\mathbb{ASCII}^*)^*$)
>
> **Relación:**
>
> $$\begin{array}{rcl} filas : \mathbb{ASCII}^n & \rightarrow & (\mathbb{ASCII}^*)^* \\ filas(s) & \rightarrow & D \text{ tal que } \forall_{i=0}^{n-1} D_{kj} = s_i \end{array}$$

Donde k =número de "," desde la posición 0 hasta la posición i y $j = i-$la posición + 1 del último "," antes de la posición i o 0 si no había

Función que separa por columnas:

Entradas: Una cadena de caracteres de tamañon ($s \in \mathbb{ASCII}^n$)

Salidas: Un arreglo de cadenas de caracteres ($C \in (\mathbb{ASCII}^*)^*$)

Relación:

$$\begin{array}{lcl} columnas : \mathbb{ASCII}^n & \rightarrow & \mathbb{R}^* \\ \quad columnas(s) & \rightarrow & C \text{ tal que } C + [folat(s_i)] \text{si } \forall_{i=0}^{n-1} s_i \neq \text{" "} \\ columnas : \mathbb{ASCII}^n & \rightarrow & (\mathbb{ASCII}^*)^* \\ \quad columnas(s) & \rightarrow & C \text{ tal que } \forall_{i=0}^{n-1} C_{kj} = s_i \end{array}$$

Donde k =número de " "(espacios en blanco) desde la posición 0 hasta la posición i y $j = i-$la posición + 1 del último " " antes de la posición i o 0 si no había

Función que crea la matriz apartir de la cadena de caracteres:

Entradas: Una cadena de caracteres ($s \in \mathbb{ASCII}^*$)

Salidas: Un arreglo de tamaño m de arreglos de tamaño n de cadenas de caracteres ($D \in (\mathbb{ASCII}^*)^m$)

Relación:

$$\begin{array}{lcl} matriz : \mathbb{ASCII}^* & \rightarrow & (\mathbb{ASCII}^*)^m \\ \quad matriz(s) & \rightarrow & D \text{ tal que } \forall_{i=0}^{m-1} D_i = columnas(A_i) \end{array}$$

Donde $A = filas(s)$ (es decir, las filas que tendría la matriz) y m =cantidad de elementos del arreglo A

Funciones que convierte los elementos en reales:

Entradas: Un arreglo de tamaño n de arreglos de tamaño m de cadenas de caracteres ($A \in (\mathbb{ASCII}^m)^n$)

Salidas: Un arreglo de tamaño n de arreglos de reales de tamaño m ($A \in (\mathbb{R}^m)^n$)

Relación:

$$\begin{array}{lcl} conviertereal : \mathbb{ASCII}^{n \times m} & \rightarrow & \mathbb{R}^{n \times m} \\ \quad conviertereal(A) & \rightarrow & B\text{tal que } \forall_{i=0}^{n-1} (\forall_{j=0}^{m-1} B_{ij} = float(A_{ij})) \end{array}$$

Funciones que convierte los elementos en enteros:

Entradas: Un arreglo de tamaño n de arreglos de tamaño m de cadenas de caracteres ($A \in (\mathbb{ASCII}^m)^n$)

Salidas: Un arreglo de tamaño n de arreglos de reales de tamaño m ($A \in (\mathbb{Z}^m)^n$)

Relación:

$$\begin{array}{lcl} conviertereal : \mathbb{ASCII}^{n \times m} & \rightarrow & \mathbb{R}^{n \times m} \\ \quad conviertereal(A) & \rightarrow & B\text{tal que } \forall_{i=0}^{n-1} (\forall_{j=0}^{m-1} B_{ij} = int(A_{ij})) \end{array}$$

Función que hace la impresión del resultado:

Entradas: Un arreglo de arreglos (matriz) de enteros o reales $C \in \mathbb{R}^{n \times m}$

Salidas: Una cadena de caracteres $e \in \mathbb{ASCII}^*$

Relación: Se crea una cadena de caracteres que representa la matriz, en la cual se genera una nueva línea cada vez que se termina una fila de la matriz y todos los elementos de esta se separan por un espacio

$$\begin{array}{rcl} salida : \mathbb{R}^{n \times m} & \rightarrow & \mathbb{ASCII}^* \\ salida(C) & \rightarrow & e \text{ tal que } \forall_{i=0}^{n-1}(\forall_{j=0}^{m-1} e_{id} = str(C_{ij})) \end{array}$$

Donde $d = 0$ si $i = 0 \wedge (2i + 1)$ En otro caso

Función que hace la suma de la matriz:

Entradas: 2 matrices de reales (o enteros) de dimensiones $n \times m$ $(A, B \in \mathbb{R}^{n \times m})$ ***Nota:*** Se toma el conjunto de los reales dado que este contiene al conjunto de los reales, además las dos matrices deben tener las mismas dimensiones

Salidas: 1 matriz de reales (o enteros) de dimensiones $n \times m$ $(C \in \mathbb{R}^{n \times m})$ ***Nota:*** Se toma el conjunto de los reales dado que este contiene al conjunto de los reales

Relación:

$$\begin{array}{rcl} suma : \mathbb{R}^{n \times m} \times \mathbb{R}^{n \times m} & \rightarrow & \mathbb{R}^{n \times m} \\ suma(A, B) & \rightarrow & C \text{ tal que } \forall_{i=0}^{n-1}(\forall_{j=0}^{m-1} C_{ij} = A_{ij} + B_{ij}) \end{array}$$

Nota: En la codificación la suma se trabajará sobre la matriz A, es decir, a los elementos de A se le sumarán los elementos de B, por lo que C = A

3. Codificación:

El código en el que se resuelve el ejercicio se encuentra en el archivo llamado «Ejercicio_50_SimonRamos.py»