Práctico Eje 6
Subarreglo y Búsquedas

Ejercicio 1

El siguiente algoritmo permite ingresar por teclado números positivos y por cada uno de ellos informar la cantidad de divisores que tiene.

- Indicar un lote de prueba y realizar el seguimiento.

```
Booleano ES_DIVISOR (entero num1, entero num2)
Comienzo
 Si (num2 mod num1 == 0)
        entonces retorna(verdadero)
        sino retorna (falso)
 finsi
Fin

Entero   CUENTA_DIVISORES (entero num)
Comienzo
Entero i, cont
Cont = 0
Para i desde 1 hasta num
     si (ES_DIVISOR (i,num))
        entonces   cont = cont + 1
     finsi
finpara
Retorna(cont)
Fin

/*Algoritmo Principal*/
Comienzo
Entero num
Leer num
Mientras (num > 0)
   Escribir “la cantidad de divisores de”, num, “es”, CUENTA_DIVISORES (num)
   Leer num
finmientras
fin
```

Ejercicio 2

El siguiente algoritmo permite ingresar por teclado números positivos e informar los números que son primos. (Nota: un número es primo cuando tiene dos divisores.)

- Indicar un lote de prueba y realizar el seguimiento.

```
Booleano ES_DIVISOR (entero num1, entero num2)
Comienzo
  Si (num2 mod num1 == 0)
        entonces retorna(verdadero)
        sino retorna (falso)
  finsi
Fin
```

```
Entero   CUENTA_DIVISORES (entero num)
Comienzo
Entero i, cont
cont = 0
Para i desde 1 hasta num
    si (ES_DIVISOR (i,num))
        entonces   cont = cont + 1
    finsi
Finpara
Retorna(cont)
Fin

/*Algoritmo Principal*/
Comienzo
Entero num
Leer num
Mientras (num > 0)
  Si (CUENTA_DIVISORES (num) == 2)
          entonces Escribir “el número”, num, “es primo”
   finsi
   Leer num
finMientras
fin
```

Ejercicio 3

El siguiente algoritmo permite ingresar por teclado números positivos e informar los números que son perfectos (Nota: un número es perfecto cuando la suma de sus divisores propios coincide con él, un número no es divisor propio de sí mismo; por ejemplo, el número 6 es perfecto pues 1+2+3=6).

- Indicar un lote de prueba y realizar el seguimiento.

```
Booleano ES_DIVISOR (entero num1, entero num2)
Comienzo
Si (num2 mod num1 == 0)
          entonces retorna(verdadero)
          sino retorna (falso)
finsi
Fin

Entero   ACUMULA_DIVISORES (entero num)
Comienzo
Entero i, acum
acum = 0
Para i desde 1 hasta num-1
   si (ES_DIVISOR(i,num))
        Entonces   acum = acum + i
   finsi
Finpara
Retorna(acum)
Fin
```

```
/*Algoritmo Principal*/
Comienzo
Entero num
Leer num
Mientras (num > 0)
    Si (ACUMULA_DIVISORES (num) == num)
      entonces   Escribir “el número”, num, “es un número perfecto”
   finsi
   Leer num
finMientras
fin
```

Ejercicio 4

El siguiente algoritmo permite ingresar por teclado números positivos y los clasifique en números: perfectos, excesivos o defectuosos.

Notas:

A) un número es perfecto cuando la suma de sus divisores propios coincide con él (un número no es divisor propio de sí mismo; (un ejemplo de número perfecto es el número 6, pues 1+2+3=6).

B) un número es excesivo cuando la suma de sus divisores propios es menor a él. (un ejemplo de número excesivo es el número 8, pues 1+2+4=7 y 7 es menor que 8)

C) un número es defectuoso cuando la suma de sus divisores propios es mayor a él. (un ejemplo de número defectuoso es el número 12, pues 1+2+3+4+6=16 y 16 es mayor que 12)

- Indicar un lote de prueba y realizar el seguimiento.

```
Booleano ES_DIVISOR (entero num1, entero num2)
Comienzo
 Si (num2 mod num1 == 0)
          entonces retorna (verdadero)
          sino retorna (falso)
 finsi
Fin
```

```
Entero   ACUMULA_DIVISORES (entero num)
Comienzo
Entero i, acum
acum = 0
Para i desde 1 hasta num-1
  si (ES_DIVISOR(i,num))
       entonces   acum = acum + i
  finsi
Finpara
Retorna (acum)
Fin
```

```
Cadena CLASIFICACIÓN (entero num1, entero num2)
Comienzo
Si (num1 == num2)
          entonces retorna (“PERFECTO”)
          sino      si (num1<num2)
                              entonces retorna (“EXCESIVO”)
                              sino retorna (“DEFECTUOSO”)
                    finsi
finsi
```

```
fin

/*Algoritmo Principal*/
Comienzo
Entero num, suma
Leer num
Mientras (num > 0)
  Si (Suma == ACUMULA_DIVISORES (num))
          entonces Escribir “el número”, num, “es un número:”, CLASIFICACIÓN (suma,num)
   finsi
   Leer num
finMientras
fin
```

Ejercicio 5

El siguiente algoritmo permite ingresar por teclado pares de números positivos e indique si ambos son amigables. Nota: dos números positivos son amigables cuando la suma de los divisores propios de uno de ellos nos da el otro y viceversa. Por ejemplo, 284 y 220 son números amigables.

- Indicar un lote de prueba y realizar el seguimiento.

```
Booleano ES_DIVISOR (entero num1, entero num2)
Comienzo
  Si (num2 mod num1 == 0)
          entonces retorna(verdadero)
          sino retorna (falso)
   finsi
Fin

Entero   ACUMULA_DIVISORES (entero num)
Comienzo
Entero i, acum
acum = 0
Para i desde 1 hasta num-1
    si (ES_DIVISOR(i,num))
        entonces   acum = acum + i
    finsi
Finpara
Retorna (acum)
Fin

Booleano SON_AMIGABLES (entero num1, entero num2)
Comienzo
Si ((ACUMULA_DIVISORES(num2) == num1) y (ACUMULA_DIVISORES(num1) == num2))
          entonces retorna (verdadero))
          sino retorna (falso)
finsi
fin
```

```
/* Algoritmo Principal*/
Comienzo
Entero num1, num2
Leer num1, num2
Mientras ((num1>0) y (num2>0))
  Si (SON_AMIGABLES (num1, num2))
        entonces   Escribir “los números”, num1, “y”, num2, “SON AMIGABLES”
        sino    Escribir “los números”, num1, “y”, num2, “NO SON AMIGABLES”
  finsi
  Leer num1, num2
finMientras
fin
```

Ejercicio 6

Construya un algoritmo que usando subprogramas permita:

1. Generar una estructura arreglo que almacene 20 números positivos diferentes.
2. Ingresar un valor y decir en el algoritmo principal si se encuentra en el arreglo.
3. Generar un subarreglo que contenga los números positivos del arreglo que tengan más de 4 divisores.
4. Mostrar el subarreglo.

Ejercicio 7

Construya un algoritmo que usando subprogramas permita:

1. Generar una estructura arreglo que almacene 20 números primos. Debe verificar que cada número que se lea sea primo antes de cargarlo en el arreglo.
2. Ingresar un valor y decir en el algoritmo principal qué lugar ocupa en el arreglo.
3. Generar un subarreglo que contenga los números primos mayores a 1000.
4. Mostrar de los datos del subarreglo que oscilan entre dos valores previamente ingresados por teclado.

Ejercicio 8

Construya un algoritmo que usando subprogramas permita:

1. Generar una estructura arreglo que almacene 20 números. Los números se ingresan en forma ordenada de menor a mayor.
2. Ingresar un valor y decir en el algoritmo principal en qué posición (el índice) se encuentra en el arreglo.
3. Generar 3 subarreglos, una con los números perfectos, otra con los excesivos y otra con los defectuosos.
4. Mostrar cada uno de los subarreglos generados.

Ejercicio 9

Construya un algoritmo que usando subprogramas permita:

1. Generar una estructura arreglo que almacene 20 números. Los números se cargan en orden ascendente.
2. Ingresar un valor y decir cuál es el primer número del arreglo con el cual es amigable.
3. Generar un subarreglo, con los números mayores al número amigable encontrado en el punto 2, sino no se encontró ninguno generar el subarreglo con los números mayores al valor ingresado.
4. Mostrar el valor promedio de los valores cargados en el subarreglo.

Ejercicio 10

Se cuenta con los datos de cada uno de los 300 corredores de una maratón de enero del 2020, auspiciada por el gobierno de San Juan. Los mismos son Número de corredor (1 a 300) Tiempo total de carrera (en minutos) y Edad.
Construya un algoritmo que usando subprogramas permita:

1. Generar las estructuras adecuadas para almacenar y guardar la información de los corredores de la maratón (hacer uso de arreglos paralelos).
2. Indicar la edad promedio y el número de corredor que supera esa Edad.
3. Generar una nueva estructura que a partir de la anterior guarde el Tiempo total de carrera de aquellos corredores mayores de 50 años.
4. A partir de la estructura generada en el ítem 3, hacer un listado que muestre el Número de corredor y su Tiempo total de carrera.

Ejercicio 11

Un gremio ha comprado una finca y la dividió en 164 lotes, de cada lote se conoce su número (entre 1 y 164) y su superficie. La información se ingresa ordenada ascendentemente por superficie.
Construya un algoritmo que usando subprogramas permita:

1. Almacenar solamente la información de aquellos lotes cuya superficie sea mayor a 650 metros cuadrados.
2. Ingresar una superficie mayor a 650 metros cuadrados y decir si algún lote tiene una superficie igual

Ejercicio 12

Se posee información de 500 abonados al servicio de gas. De cada abonado se conoce el número de cliente entre 200 y 700 (200,201,202...700), consumo de gas y el importe mensual a abonar.
Construya un algoritmo que usando subprogramas permita:

1. Generar arreglos paralelos para almacenar los datos de los abonados, sabiendo que la información se ingresa ordenada de manera ascendente por importe (desde menor hasta el mayor).
2. Decir si algún usuario superó los 5000 pesos de importe en su factura.
3. Ingresar por teclado un consumo de gas y decir el importe mensual a abonar.
4. Generar una nueva estructura con el número de usuario de aquellos que superan el importe promedio de consumo.
5. Mostrar los números de usuarios que fueron almacenados en la nueva estructura.