**Universidad Católica Andrés Bello**

**Facultad de Ingeniería informática**

**Cátedra de Algoritmos y Programación I**

**Docentes: Franklin Bello y Zulma Díaz**

Alumnos:

Edwin Rodríguez – C.I: 30437298

Juan Quijada – C.I: 30212474

Práctica #3 – Estructuras repetitivas.

*Ejercicio #1:*

1.- Dado una secuencia de números naturales, finalizada con un cero, construya un

algoritmo que determine : el mayor, el menor y el promedio de los números Perfectos y

a su vez Malvados, según la definición dada a continuación.

Número perfecto: es todo número natural que es igual a la suma de sus divisores propios

(es decir, todos sus divisores excepto el propio número). Por ejemplo, 6 es un número

perfecto ya que sus divisores propios son 1, 2, y 3 y se cumple que 1+2+3=6.

Número malvado: todo número natural cuya expresión en base 2 (binaria) contiene un

número par de unos. Por ejemplo el 12 y el 15 son números Malv ya que 12=1100

contiene dos(2) unos en su representación binaria y 15=1111 contiene cuatro(4) unos

en su representación binaria ( dos y cuatro son pares).

Nota : Para llevar un número natural a binario, hacer divisiones sucesivas entre 2 y tomar

el resto.

**Análisis del problema:**

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| Los números ingresados en la secuencia (Num) | Pedir dato de entrada (num).  Verificar si el número es cero.  Si no es cero, verificar si es perfecto, malvado, o ambos.  Si se cumple cualquiera de las condiciones anteriores, acumular y contar.  Verificar si el número es el nuevo mayor o menor, y almacenarlo si lo es.  Pedir dato (num) nuevamente y repetir la secuencia hasta que se ingrese un cero.  Calcular el promedio.  Mostrar resultado. | Mostrar resultado:  Numero mayor, número menor, promedio de números.  O en el caso contrario:  ‘No se ingresaron números malvados o perfectos’. |

**Algoritmo:**

Programa: NumerosPerfectos\_o\_Malvados;

Variables:

Num, AuxBin, AuxBin2, AuxP, PerfoMalv, MenorPM, MayorPM, i: Entero;

PPM: Real;

Perf, Malv : Booleano;

Inicio

PerfoMalv := 0;

MayorPM := 0;

MenorPM := 0;

PPM := 0;

AuxP := 0;

AuxBin2 := 0;

Escribir('Este programa calcula el promedio, el mayor, y el menor de los numeros perfectos y Malvados de una secuencia numerica. ');

Escribir('Ingrese una secuencia de numeros, utilizando el cero (0) para terminarla.');

Leer(Num);

Repita mientras (Num <> 0)

Perf = falso;

Malv = falso;

Si (PerfoMalv = 0) entonces

MenorPM = Num+1;

Fin-si

AuxP = 0;

Para (i = 1) hasta (Num-1) haga

Si (Num Mod i = 0) entonces

AuxP := AuxP + i;

Fin-si

Fin-para

Si (AuxP = Num) entonces

perf = verdadero;

Fin-si

AuxBin = Num;

i = 0;

Repita

AuxBin2 = AuxBin Mod 2;

AuxBin = AuxBin Div 2;

Si (AuxBin2 <> 0) entonces

i = i+1;

Fin-si

Hasta que (AuxBin=0);

Si (i Mod 2 = 0) entonces

Malv = verdadero;

Fin-si

Si (Malv = verdadero) o (Perf = verdadero) entonces

Si (Num > MayorPM) entonces

MayorPM = Num;

Fin-si

Si (Num < MenorPM) entonces

MenorPM = Num;

PPM = PPM+Num;

PerfoMalv = PerfoMalv+1;

Fin-si

Fin-si

Escribir('Ingrese otro numero. Recuerde usar cero (0) para terminar la secuencia.');

Leer(Num);

Fin repita-mientras

Si (PerfoMalv <> 0) entonces

PPM = PPM/PerfoMalv;

Escribir('Numeros Perfectos o Malvados: ');

Escribir('El mayor es: ', MayorPM);

Escribir('El menor es: ', MenorPM);

Escribir('Y su promedio fue de: ', PPM:0:2);

De lo contrario

Escribir('No hubieron numeros perfectos o malvados en la secuencia.');

Fin-si

Fin

**Código fuente:**

Program NumerosPerf\_Y\_NumerosMalvados;

Uses Crt;

Var

//AuxBin,AuxBin2: Auxiliares para emular la obtencion del numero binario para verificar los numeros malvados; AuxP: Auxiliar para verificar los numeros perfectos; PerfOmalv: Contador de perfectos o malvados; MenorPM, MayorPM: Mayor y menor de perfectos o malvados; i: Contador general.

Num,AuxBin,AuxBin2,AuxP,PerfoMalv,MenorPM,MayorPM,i: integer;

//PPM: Promedio de los perfectos o malvados

PPM: Real;

Perf,Malv : Boolean;

Begin

PerfoMalv := 0;

MayorPM := 0;

MenorPM := 0;

PPM := 0;

AuxP := 0;

AuxBin2 := 0;

Writeln('Este programa calcula el promedio, el mayor, y el menor de los numeros perfectos y Malvados de una secuencia numerica. ');

Writeln('Ingrese una secuencia de numeros, utilizando el cero (0) para terminarla.');

Readln(Num);

While Num<>0 Do

Begin

Perf := false;

Malv := false;

If PerfoMalv=0 Then

MenorPM := Num+1;

//Se verifica si el numero es perfecto

AuxP := 0;

For i := 1 To Num-1 Do

Begin

If Num Mod i = 0 Then

AuxP := AuxP + i;

End;

If AuxP = Num Then

perf := true;

//Se verifica si el numero es malvado

AuxBin := Num;

i := 0;

Repeat

AuxBin2 := AuxBin Mod 2;

AuxBin := AuxBin Div 2;

If AuxBin2 <> 0 Then

i := i+1;

Until (AuxBin=0);

If i Mod 2 = 0 Then

Malv := True;

//Si es perfecto o malvado, se acumula para el promedio, se cuenta dentro de un contador, y se verifica si hay un nuevo mayor o menor.

If (Malv=true) Or (Perf=true) Then

Begin

If Num>MayorPM Then

MayorPM := Num;

If Num<MenorPM Then

MenorPM := Num;

PPM := PPM+Num;

PerfoMalv := PerfoMalv+1;

End;

Writeln('Ingrese otro numero. Recuerde usar cero (0) para terminar la secuencia.');

Readln(Num);

End;

If PerfoMalv<>0 Then

Begin

PPM := PPM/PerfoMalv;

Writeln('Numeros Perfectos o Malvados: ');

Writeln('El mayor es: ', MayorPM);

Writeln('El menor es: ', MenorPM);

Writeln('Y su promedio fue de: ',PPM:0:2);

End

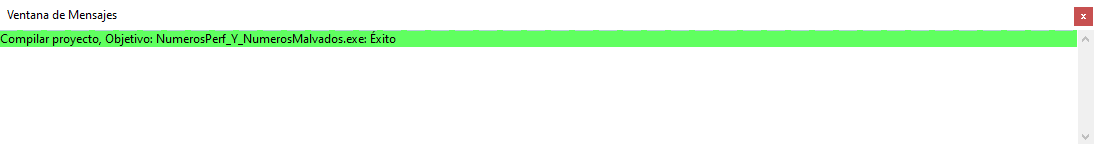
Else

Writeln('No hubieron numeros perfectos o malvados en la secuencia.');

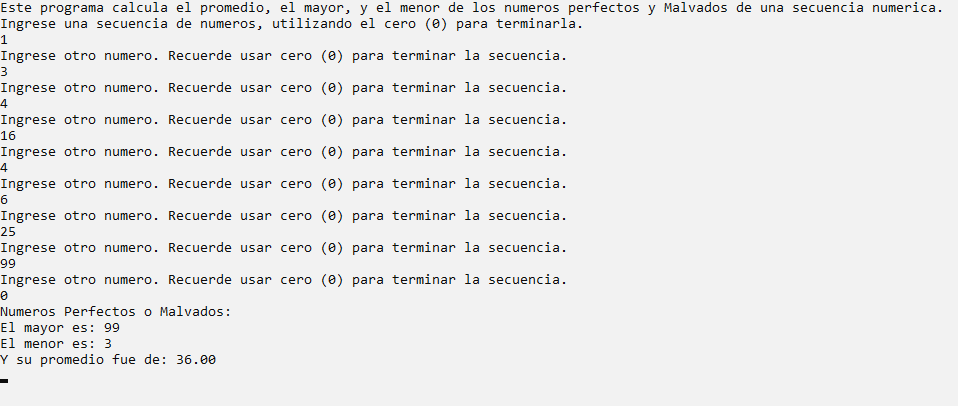
Readkey();

End.

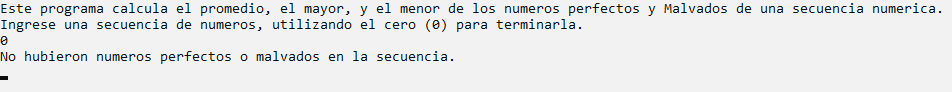
**Resultado de la compilación:**



**Ejecución del programa:**



*Caso 1*



*Caso 2*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ejercicio #2:*

2.- Construya un programa que lea los elementos de una secuencia de números enteros que termina con un 0. Si la secuencia es estrictamente descendente el programa debe escribir el mensaje “Su lista de N elementos es descendente”, donde N es la cantidad de elementos de la lista, incluyendo el 0. Si la secuencia no es estrictamente descendente debe escribir el mensaje “Su lista de N elementos no es descendente, ya que aparece A seguido de B”,donde A y B son dos números que aparecen en la secuencia, A inmediatamente antes de B y B no es menor que A.

Por ejemplo, si la secuencia en la entrada es 10, 9, 4, 2, 0, la salida debe ser :

Su lista de 5 elementos es descendente.

Por otra parte, si la secuencia en la entrada es 8, 6, 5, 7, 6, 8, 0, la salida debe ser:

Su lista de 7 elementos NO es descendente, ya que aparece 5 seguido de 7.

De igual manera para la lista 6, 3, 3, 0, la salida debe ser:

Su lista de 4 elementos NO es descendente ya que aparece 3 seguido de 3.

Nota : Para la corrida de este programa usar los datos de los ejemplo dados.

**Análisis del problema:**

|  |  |  |
| --- | --- | --- |
| Entrada | Proceso | Salida |
| Los números de la secuencia (Num) | Pedir el dato de entrada (num).  Verificar si el número ingresado es cero.  Si no es cero, verificar si la secuencia es descendente.  Repetir la secuencia hasta que se ingrese un cero.  Mostrar resultado | El resultado de la evaluación, es decir, si la secuencia es descendente o no.  ‘La secuencia es descendente’.  O en el caso contrario:  ‘La secuencia no es descendente’. |

**Algoritmo:**

Programa:DescendienteEstrictamente;

Variables:

Num, i, NumPost, NumAnt: entero;

Descendente: Booleano;

Inicio

i = 1;

Descendente = verdadero;

Escribir('Este programa verifica si una secuencia de numeros es descendiente.');

Escribir('Ingrese una secuencia de numeros enteros terminada por 0.');

Leer(Num);

NumAnt = Num+1;

Repita mientras (Num <> 0)

i := i+1;

Si (NumAnt <= Num) y (descendente = verdadero) entonces

Descendente := falso;

NumPost := Num;

Fin-si

Si (Descendente = verdadero) entonces

NumAnt := Num;

Escribir('Ingrese otro número');

Leer(Num);

Fin-si

Fin repita-mientras

Si (i=1) entonces

Escribir(‘Usted no ha ingresad una secuencia de números a evaluar’)

De lo contrario

Si (Descendente = verdadero) entonces

Escribir('Su lista de ', i,' elementos es descendente.')

De lo contrario

Escribir('Su lista de ', i,' elementos NO es descendente, ya que aparece ', NumAnt,' seguido de ', NumPost,'.');

Fin-si

Fin-si

Fin.

**Código fuente:**

Program DescendienteEstrictamente;

Uses Crt;

Var

// NumAnt: El numero anterior al que se ingresa; NumPost: El Numero que rompe la secuencia descendente; Descendente: El booleano que almacena si es descendente o no.

Num, i, NumPost,NumAnt: Integer;

Descendente: Boolean;

Begin

//Se inicializa el contador a 1 para que cuente el 0 dentro de la secuencia; tambien se inicializa el booleano en verdadero porque idealmente todos los numeros ingresados seran mayores que 0, el ultimo de la secuencia.

i := 1;

Descendente := true;

Writeln('Este programa verifica si una secuencia de numeros es descendiente.');

Writeln('Ingrese una secuencia de numeros enteros terminada por 0.');

Readln(Num);

//Se aumenta el numero anterior en 1 al ingresado para que siempre sea mayor en la primera iteracion.

NumAnt := Num+1;

While (Num<>0) Do

Begin

i := i+1;

//Se verifica si la secuencia esta siendo descendente, y si no, se asigna un falso al booleano y se guardan los valores de los numeros de la secuencia que rompieron el patron, es decir, el anterior y el ingresado que es mayor al anterior.

If (NumAnt <= Num) And (descendente=true) Then

Begin

Descendente := false;

NumPost := Num;

End;

//Se verifica si sigue siendo descendente para asignar el numero ingresado al numero anterior para la siguiente iteracion

If (Descendente=true) Then

NumAnt := Num;

Writeln('Ingrese otro numero');

Readln(Num);

End;

//Se envia un mensaje mostrando si la secuencia es descendente o no.

If i=1 then

Writeln(‘Usted no ha ingresado una secuencia de numeros a evaluar’)

else

If Descendente=true Then

Writeln('Su lista de ',i,' elementos es descendente.')

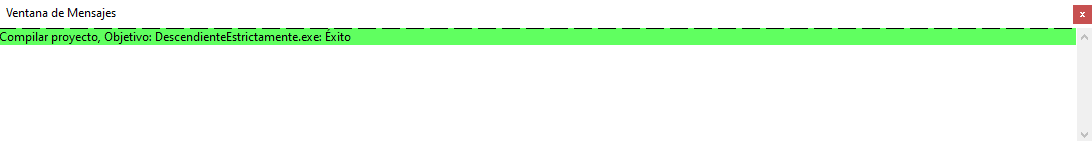
Else

Writeln('Su lista de ',i,' elementos NO es descendente, ya que aparece ',NumAnt,' seguido de ',NumPost,'.');

Readkey();

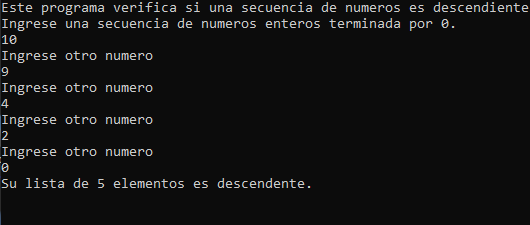
End.

**Resultado de la compilación:**

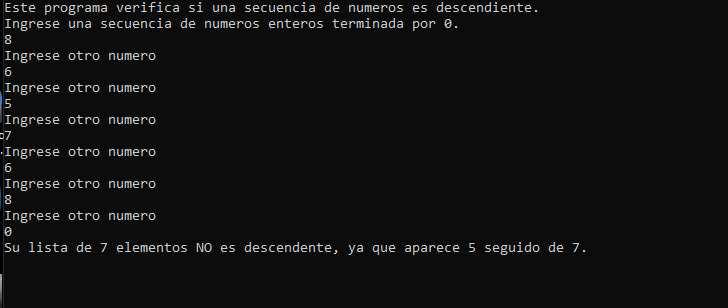


**Ejecución del programa:**

*Caso 1*



Caso 2



Caso 3

