Índice

Actividades prácticas

Metodología de programación

[Actividades prácticas 3](#_Toc432431439)

[Actividad práctica 0. Arquitecturas 3](#_Toc432431440)

[Actividad práctica 1. Conversiones entre unidades 3](#_Toc432431441)

[Actividad práctica 2. Análisis de ordinogramas 4](#_Toc432431442)

[Actividad práctica 3. Realización de ordinograma 6](#_Toc432431443)

[Actividad práctica 4. Seguimiento de algoritmos 6](#_Toc432431444)

[Actividad práctica 5. Realización de algoritmos en pseudocódigo 11](#_Toc432431445)

[Actividad práctica 6. Realización de algoritmos con programación estructurada 11](#_Toc432431446)

[Actividad práctica 7. Realización de algoritmos con arrays 12](#_Toc432431447)

[Actividad práctica 8. Realización de un algoritmo con array multidimensional 13](#_Toc432431448)

[Actividad práctica 9. Realización de algoritmo para un programa con ordenación de arrays 13](#_Toc432431449)

[Actividad práctica 10. Realización de algoritmo utilizando programación modular 14](#_Toc432431450)

[Actividad práctica 11. Realización de algoritmo basado en la programación por capas. 14](#_Toc432431451)

[Actividad práctica 12. Realización de aplicación basada en el uso de clases. 15](#_Toc432431452)

[Actividad práctica 13. Realización de aplicación basada en el uso de clases con herencia. 16](#_Toc432431453)

[Soluciones 18](#_Toc432431454)

[Actividad práctica 1 18](#_Toc432431455)

[Actividad práctica 2 19](#_Toc432431456)

[Actividad práctica 3 20](#_Toc432431457)

[Actividad práctica 4 21](#_Toc432431458)

[Actividad práctica 5 22](#_Toc432431459)

[Actividad práctica 6 24](#_Toc432431460)

[Actividad práctica 7 29](#_Toc432431461)

[Actividad práctica 8 34](#_Toc432431462)

[Actividad práctica 9 36](#_Toc432431463)

[Actividad práctica 10 41](#_Toc432431464)

[Actividad práctica 11 46](#_Toc432431465)

[Actividad práctica 12 54](#_Toc432431466)

[Actividad práctica 13 60](#_Toc432431467)

# Actividades prácticas

## Actividad práctica 0. Arquitecturas

Explicar las características de las arquitecturas Von Newman y Harvard, resaltando las diferencias entre las mismas.

## Actividad práctica 1. Conversiones entre unidades

Realiza las siguientes conversiones entre sistemas de numeración.

1. Convierte a binario los siguientes números decimales:

120

493

1. Convierte a binario los siguientes números hexadecimales:

47F

A8C

1. Convierte a hexadecimal los siguientes números binarios:

110010100

1000010111

1. Convierte a hexadecimal los siguientes números decimales:

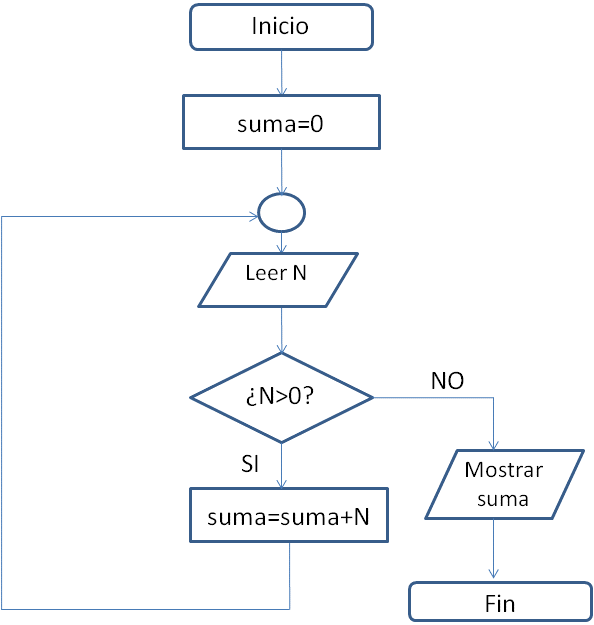
1789

455

## Actividad práctica 2. Análisis de ordinogramas

Ordinograma 1

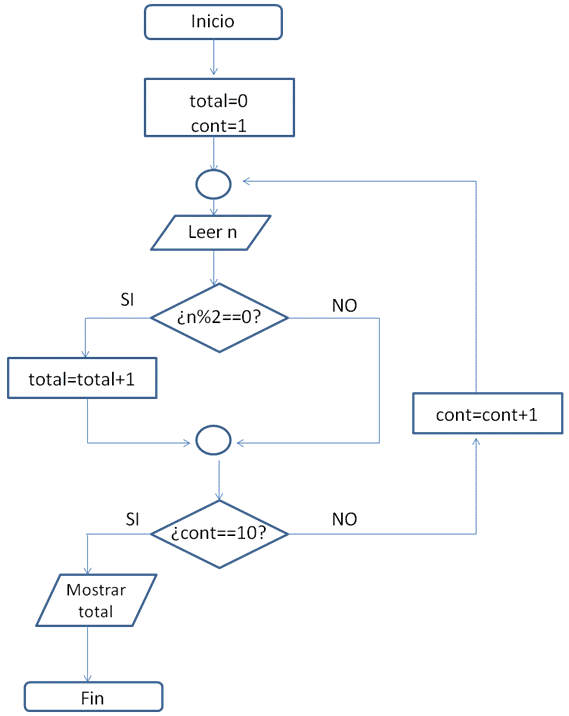
A continuación, dado el siguiente ordinograma, te proponemos que respondas a una serie de preguntas:



¿Qué es lo que hace el algoritmo anterior?. ¿Qué se mostraría si la secuencia de números introducidos fuera 2, 5, 9, 3, -1, 4?

Ordinograma 2

Seguidamente te proponemos el análisis de un nuevo ordinograma. Teniendo en cuenta que el operador “%” calcula el resto de la división entre dos números, dado el siguiente diagrama de flujo:



Explica el funcionamiento del programa y que es lo que hace.

Si se leen los siguientes números: 2, 4, 5, 1, 9, 17, 25, 16, 30 y 11. ¿Cuál sería el valor de la variable total al finalizar el programa?¿y de la variable cont?

## Actividad práctica 3. Realización de ordinograma

Realiza un ordinograma para un algoritmo que calcule el total de números naturales que hay que sumar, comenzando por el 1, hasta que la suma alcance el valor 1000.

## Actividad práctica 4. Seguimiento de algoritmos

Algoritmo 1

Indica el valor de las variables A y B al finalizar el siguiente algoritmo:

Inicio

Datos:

A, B, C entero

Código:

A=7

B=4

C=9

Mientras (A<>C) hacer

Si (C>9) entonces

B=B+5

Sino

A=A+2

Fin Si

Fin Mientras

Fin

Algoritmo 2

Indica los valores de las variables X, Y y Z al finalizar el siguiente proceso.

Inicio

Datos:

X entero

Y entero

Z entero

Código:

X=0

Y=7

Z=-4

Mientras (X > Z) hacer

Si (Y<15) entonces

Y=Y+4

Sino

Si(Z<0) entonces

Z=Z+2

Sino

Z=Z+1

X=X-1

Fin si

Y=Y+3

Fin si

Fin mientras

Fin

Algoritmo 3

Indica el valor que se imprimirá al finalizar el proceso:

Inicio

Datos:

P,X, Contador entero

Código:

P=4

X=0

Contador=0

Mientras(P<>8) hacer

X=X+5

P=P+1

Contador=Contador+1

Fin Mientras

Mostrar “El valor de X es”, X

Fin

Algoritmo 4

Indica el valor de la variable X al finalizar el proceso siguiente:

Inicio

Datos:

A, B, X, Y, Z entero

Código:

A=15

B=15

X=100

Y=100

Z=0

Mientras(A==B) hacer

A=A+3

Mientras(X==Y) hacer

X=X+Z

Fin Mientras

B=B+3

Fin Mientras

Fin

Algoritmo 5

Indica el valor de la variable Z al finalizar el proceso siguiente:

Inicio

Datos:

A, B, X, Y, Z entero

Código:

A=60

B=0

X=100

Y=100

Z=100

Mientras(A<>B) hacer

A=A+10

Mientras(X==Y) hacer

X=X+Z

Fin Mientras

B=B+20

A=A-10

Y=X

Fin Mientras

Fin

Algoritmo 6

Indica el valor al que habrá que inicializar P para que al finalizar el proceso el valor que se imprima de X sea 10:

Inicio:

Datos:

X, P entero

Código:

X=0

P=**?**

Mientras(P<=7) hacer

X=X+1

P=P+1

Fin Mientras

Mostrar “El valor de X es “, X

Fin

## Actividad práctica 5. Realización de algoritmos en pseudocódigo

Algoritmo 1. Realiza el algoritmo correspondiente a un programa que lea un número y muestre un texto indicando si el número es o no primo

Algoritmo 2. Diseñar un algoritmo que lea dos números y realice la suma de todos los números pares comprendidos entre ambos números.

## Actividad práctica 6. Realización de algoritmos con programación estructurada

Algoritmo 1. Diseña un algoritmo para la lectura de 20 números y que nos muestre la suma de los pares por un lado y la de los impares por otro.

Algoritmo 2. Diseña un algoritmo para un programa que lea 10 textos y nos muestre aquel que tenga el mayor número de caracteres. Si hay más de un texto con el mismo número de caracteres, mostrará el último de ellos. Puedes utilizar la función Longitud para este ejercicio.

Algoritmo 3. Diseñar un algoritmo para la lectura de 20 números y que nos muestre la media de todos los números leídos y cuál ha sido el más pequeño

Algoritmo 4. Realizar un algoritmo para un programa que lea 15 números y nos diga cuántos de ellos son primos. De cara a la lectura de los números, si el número introducido es negativo se le deberá volver a pedir de nuevo el número hasta que se introduzca uno positivo.

Algoritmo 5. Realizar el diseño de un algoritmo para un programa que lea un texto y una letra, para, posteriormente indicarnos el número de veces que la letra aparece en el texto. Puedes emplear la función ObtenerLetra utilizada en los ejercicios de ejemplo del manual del alumno.

## Actividad práctica 7. Realización de algoritmos con arrays

Algoritmo 1. Realizar un algoritmo para programa que lea 10 números y los guarde en un array. Posteriormente, se pedirá otro número y el programa nos indicará cuantos de los elementos del array son mayores que este número.

Algoritmo 2. Realizar un algoritmo que lea un número, si es positivo, lo guardará en un array, sino volverá a leer un nuevo número, así hasta conseguir rellenar el array, que tendrá 5 elementos. Tras rellenar el array, nos mostrará su contenido en orden inverso a como han sido leídos los números

Algoritmo 3. Tenemos un array de enteros de tamaño 20. Realizar un algoritmo que vaya leyendo números y los guarde en el array, pero si la suma de los números leídos alcanza o supera 100, se dejarán de leer números y rellenará con 0 las posiciones que queden libres.

Algoritmo 4. Realizar un algortimo que genere una combinación de la lotería primitiva. El objetivo es generar 6 números aleatorios entre 1 y 49 que no se repitan. Puedes utilizar la función GenerarAleatorio utilizada en los ejemplos del manual

## Actividad práctica 8. Realización de un algoritmo con array multidimensional

Realizar un algoritmo para un programa que lea el nombre y temperatura de una ciudad y los guarde en un array, hasta un total de 10 ciudades. A continuación, nos mostrará el nombre y temperatura de todas las ciudades cuya temperatura sea inferior a la media. Podemos usar una función ConvertirEntero, que nos convierte una cadena formada por caracteres numéricos a entero.

## Actividad práctica 9. Realización de algoritmo para un programa con ordenación de arrays

Algoritmo 1

Realizar un algoritmo para un programa que se encargue de leer 10 textos y, posteriormente, nos muestre los textos ordenados de menor a mayor número de caracteres. En caso de que dos textos tengan el mismo número de caracteres, se pondrá primero el que tenga mayor número de vocales. Para este algoritmo podemos utilizar las funciones Longitud y ObtenerLetra.

Algoritmo 2

Dado un array de dos dimensiones de 4x6, realizar un programa que lea 24 números y los coloque en el array. Posteriormente, se ordenará el array de menor a mayor (la posición 1,1 tendrá el número más pequeño, mientras que 4,6 tendrá el mayor) y se mostrará ordenado

## Actividad práctica 10. Realización de algoritmo utilizando programación modular

Algoritmo 1

Aplicando las técnicas de la programación modular, realizar una nueva versión del algoritmo 1, desarrollado en la práctica 8 para ordenación de un array. El cálculo del número de vocales de una cadena y la muestra de su contenido se realizarán en procedimientos/funciones independientes.

Algoritmo 2

Aplicando técnicas de programación modular, realizar una serie de procedimientos y funciones para realizar las siguientes operaciones:

* A partir de dos números nos devuelva el menor de los dos
* A partir de dos números nos devuelva el mayor de los dos
* A partir de dos números, devuelva la suma de todos los números comprendidos entre ellos
* A partir de un número nos devuelva el factorial de dicho número.

Después, el programa solicitará dos números y, usando los procedimientos y funciones anteriores, mostrará la suma de los números comprendidos entre estos dos números y el factorial de cada uno.

## Actividad práctica 11. Realización de algoritmo basado en la programación por capas.

Algoritmo 1

Realizar un algoritmo basado en un programa que solicite 10 números y los guarde en un array. Posteriormente, el programa solicitará otro número y nos indicará:

* Cuantos números del array son mayores que el número leído
* La suma de todos los números del array que sean menores que el número leído
* Los números del array que sean múltiplos del número leído

Para resolver este ejercicio, utilizaremos programación modular con separación en capas, es decir, el módulo principal se encargará de la lectura de datos y presentación de resultados y la realización del los cálculos se harán en procedimientos y/o funciones independientes. También vamos a considerar que los arrays disponen de una propiedad llamada *length* con la que podemos conocer el número de elementos que tienen en todo momento, y que se utilizaría: *variablearray.length*

Algoritmo 2

Realizar un algoritmo para un programa que solicitará un texto y lo almacenará en un array, siempre que el texto no esté repetido, en cuyo caso volverá a solicitar uno nuevo, así hasta completar 10 textos. Después, se solicitará la introducción de dos letras y el programa nos mostrará, de todos los textos guardados, aquellos cuya primera y última letra coincidan con las dos letras leídas, respectivamente. Se pueden utilizar las funciones Longitud y ObtenerLetra utilizadas en otros ejercicios.

## Actividad práctica 12. Realización de aplicación basada en el uso de clases.

Algoritmo 1

Realizar una clase que ofrezca la funcionalidad de una Pila. Dispondrá de un método para agregar una cadena a la pila y otro recuperar que nos devolverá la última cadena agregada a la pila y simulará la eliminación de la misma. El tamaño máximo de la pila será de 10, por lo que no se podrán añadir nuevos elementos una vez alcanzado dicho tamaño

Algoritmo 2

Se trata de realizar una nueva versión de la aplicación para la gestión de notas empleando la separación de código por capas a través de clases.

El programa principal mostrará un menú con las siguientes opciones:

1. Añadir nota
2. Mostrar media
3. Total aprobados
4. Mostrar notas introducidas
5. Salir

Tras el tratamiento de cada opción, volverá a mostrarse el menú hasta la elección de la opción 6.

El funcionamiento de cada opción será el siguiente:

* Añadir nota. Se solicitará la introducción de una nota y dicha nota será guardada. El máximo de notas a almacenar es de 10, por lo que si se elije esta opción cuando ya no hay espacio para más notas se le informará al usuario de la situación y se volverá a mostrar el menú
* Mostrar media. Se mostrará la media de las notas almacenadas hasta el momento
* Total aprobados. Se mostrará el total de aprobados entre las notas almacenadas hasta el momento
* Mostrar notas introducidas. Se mostrarán las notas almacenadas hasta el momento, ordenadas de menor a mayor.

La gestión de las notas se realizará en una clase independiente al módulo principal.

## Actividad práctica 13. Realización de aplicación basada en el uso de clases con herencia.

Algoritmo 1

Crear una clase Punto que represente un punto en el plano y disponga, además de propiedades y constructores, de un método imprimir() que simplemente mostrará las coordenadas del punto y otro Cambiar() que intercambia los valores de las coordenadas X e Y.

Después, crearemos otra clase Punto3D que heredará Punto e incorporará la nueva coordenada Z. Además de sobrescribir imprimir() para adaptarlo a la nueva clase, sobrecargará Cambiar con una versión que hará lo mismo que la heredada y, además, recibirá un parámetro que servirá de nuevo valor de la coordenada Z

Algoritmo 2

Crear una nueva clase, llamada CuentaMovimientos, que herede la clase Cuenta utilizada durante la explicación de la herencia. La nueva clase incorporará una nueva propiedad, llamada Movimientos, que devolverá los cinco últimos movimientos realizados en la cuenta. Cada movimiento estará caracterizado por un tipo (extracción o inserción) y la cantidad involucrada. Los datos se devolverán como un array de 5X2.

# Soluciones

## Actividad práctica 1

1. Convierte a binario los siguientes números decimales:

120-> 1111000

493->1111010001

1. Convierte a binario los siguientes números hexadecimales:

47F->10001111111

A8C->10010001100

1. Convierte a hexadecimal los siguientes números binarios:

110010100->194

1000010111->217

1. Convierte a hexadecimal los siguientes números decimales:

1789->6FD

455->1C7

## Actividad práctica 2

Ordinograma 1

El algoritmo lee números hasta introducir uno negativo, momento en el cual muestra la suma de los números positivos leídos. Si se introduce la secuencia de números indicada, se mostrará el valor 19. El último número (4) no llegará a leerse.

Ordinograma 2

El algoritmo lee 10 números y muestra cuántos de ellos son pares. En el caso de los números indicados, la variable *total* marcará el valor 4 al finalizar el algoritmo, puesto que hay cuatro números pares. La variable *cont* por su parte tendrá el valor 10, que es el total de números leídos.

## Actividad práctica 3

## Actividad práctica 4

Algoritmo 1

A=9 y B=4

Algoritmo 2

X=0, Y=21 y Z=0

Algoritmo 3

X=20

Algoritmo 4

Bucle infinito

Algoritmo 5

X=400

Algoritmo 6

P=-2

## Actividad práctica 5

Algoritmo 1

Inicio

Datos:

Primo texto

N,Cont entero

Código:

Primo=”si” //variable que indicará si es o no primo

Cont=1 //inicia el contador en 1

Leer N

Mientras (Cont<N) hacer

//si es divisible entre algún número

//menor que él, no es primo

Si(N%Cont==0) entonces

Primo=”no”

Fin Si

Cont=Cont+1 //incrementa el contador

Fin Mientras

Si(Primo==”si”) entonces

Mostrar “El número es primo”

Sino

Mostrar “El número no es primo”

Fin Si

Fin

Algoritmo 2

Inicio

Datos:

N1, N2, Cont entero

Mayor, Menor entero

Suma entero

Codigo:

Suma=0

Leer N1, N2

Si(N1>N2) entonces

Mayor=N1

Menor=N2

Sino

Mayor=N2

Menor=N1

Fin Si

Cont =Menor

Mientras(Cont <=Mayor) hacer

Suma=Suma+ Cont

Cont = Cont +1

Fin Mientras

Mostrar “La suma es “, Suma

Fin

## Actividad práctica 6

Algoritmo 1

Inicio

Datos:

Spares, Simpares Integer

N,I Integer

Código:

Spares=0

Simpares=0

For I=1 To 20 Step 1

Leer N

If(N%2==0) Then

Spares=Spares+1

Else

Simpares=Simpares+1

End If

End For

Mostrar “El total de pares es: “, Spares

Mostrar “El total de impares es:”,Simpares

Fin

Algoritmo 2

Inicio

Datos:

Texto, Largo String

I Integer

Código:

Largo=”” //guardará el texto con más caracteres

For I=1 To 10 Step 1

Leer Texto

//si el nuevo texto leído tiene longitud mayor

//lo guarda en la variable Largo

If(Longitud(Texto)>Longitud(Largo)) Then

Largo=Texto

End If

End For

Mostrar “El texto más largo es “, Largo

Fin

Algoritmo 3

Inicio

Datos:

Media, Menor, N, I Integer

Código:

Media=0

For I=1 To 20 Step 1

Leer N

//si es el primer número leído lo utiliza para

//inicializar la variable Menor

If(I==1) Then

Menor=N

Else

//si el número leído es más pequeño que menor

//actualiza la variable con dicho número

If(N<Menor) Then

Menor=N

End If

End If

Media = Media+N

End For

Media=Media/20

Mostrar “La media es: “, Media

Mostrar “El menor leído es: “, Menor

Fin

Algoritmo 4

Inicio

Datos:

Total, I, K, Num Integer

Primo String

Código:

Total=0

For I=1 To 15 Step 1

Leer Num

Do While(Num<0)

Mostrar “Número negativo, debe introducir otro”

Leer Num

End While

//comprobación de primo

Primo=”si”

For K=1 To (Num-1) Step 1

If(Num%K==0) Then

Primo=”no”

End If

End For

//si el número leído era primo, incrementa contador

If(Primo==”si”) Then

Total=Total+1

End If

End For

Mostrar “El total de números primos leídos es “, Total

Fin

Algoritmo 5

Inicio

Datos:

Letra, Comparar, Cadena String

I, Cont, Lon Integer

Código:

Cont=0

Leer Cadena

Leer Letra

Lon=Longitud(Cadena)

//recorre las letras de la cadena

For I=1 To Lon Step 1

Comparar=ObtenerLetra(Cadena, I)

If(Letra==Comparar)

Cont=Cont+1

End If

End For

Mostrar “El total de veces que aparece la letra es: “, Cont

Fin

## Actividad práctica 7

Algoritmo 1

Inicio

Datos:

Nums 1 To 10 Integer

Cont, I, Dato, N Integer

Código:

Cont=0

For I=1 To 10 Step 1

Leer N

Nums[I]=N

End For

Leer Dato

For I=1 To 10 Step 1

If(Nums[I]>Dato) Then

Cont=Cont+1

End If

End For

Mostrar “El total de elementos mayores es: “, Cont

Fin

Algoritmo 2

Inicio

Datos:

Positivos 1 To 5 Integer

N, I, Guardados Integer

Código:

Guardados=0

//mientras haya espacio, lee número

//y si es positivo lo guarda

Do While (Guardados<5)

Leer N

If(N>0) Then

Guardados=Guardados+1

Positivos[Guardados]=N

End If

End While

For I=5 To 1 Step -1

Mostrar Positivos[I]

End For

Algoritmo 3

Inicio

Datos:

Nums 1 To 20 Integer

Suma, I, Guardados Integer

Código:

Suma=0

I=0

//realiza la lectura y grabado de números en el array

//mientras la suma no alcance 100 y haya espacio

Do While(Suma<100 AND I<20)

Leer N

Suma=Suma +N

I=I+1

Nums[I]=N

End While

//si no ha llegado a completer la lectura de los 20 númeos

//rellena el resto con 0

If (I<20) Then

For I=I+1 To 20 Step 1

Nums[I]=0

End For

End If

Fin

Algoritmo 4

Inicio

Datos:

Combi 1 To 6 Integer

Totales Integer //guarda la cantidad de números generados

Num,I Integer

Existe Boolean

Código:

Totales=0

Existe=false

//genera número aleatorio, si no ha sido generado antes lo guarda

//en el array, y sino vuelve a generar otro. Repite el proceso

//hasta tener 6

Do

Num=GenerarAleatorios(1, 49)

//comprueba si el número generado ya existe

//entre los almacenados en el array, si existe, pone

//la variable a true

For I=1 To Totales Step 1

If(Num==Combi[I]) Then

Existe=true

End If

End For

//si el número generado no existe, lo guarda

//en la posición correspondiente

If(Existe==false) Then

Totales=Totales+1

Combi[Totales]=Num

End If

//se inicializa a falso para la siguiente comprobación

Existe=false

While(Totales<6)

//muestra todo el contenido del array

For I=1 To 6 Step 1

Mostrar Combi[I]

End For

Fin

## Actividad práctica 8

Inicio

Datos:

//Array para guardar las 10 parejas de datos, ciudad y temperatura

//como todos los elementos del array deben ser del mismo tipo, las

//temperaturas se guardan como String

Ciudades 1 to 10, 1 to 2 String

Fila, Vtemp Integer

Nombre, Temp String

Media Decimal

Código:

Media=0

For Fila=1 To 10 Step 1

//solicita el nombre y temperatura de cada ciudad

//y lo guarda en una fila del array

Leer Nombre

Leer Temp

Ciudades[Fila][1]=Nombre

Ciudades[Fila][2]=Temp

//usamos un function que nos convierte un valor de texto

//formado por caracteres numéricos a número entero

Vtemp=ConvertirEntero(Temp)

Media=Media+Vtemp

End For

//calcula la media final

Media=Media/10

For Fila=1 To 10 Step 1

//convierte cada temperatura a entero

Vtemp= ConvertirEntero(Ciudades[Fila][2])

IfVtemp<Media) Then

//muestra la ciudad unido a la temperatura

Mostrar Ciudades[Fila][1] & “-“ & Ciudades[Fila][2]

End If

End For

Fin

## Actividad práctica 9

Algoritmo 1

Inicio

Datos:

Cadenas 1 To 10 String

Texto, Aux, Letra String

Vocales1, Vocales2, Len1, Len2 Integer

I, K, Pos Integer

Código:

Vocales1=0

Vocales2=0

For Pos=1 To 10 Step 1

Leer Texto

Cadenas[Pos]=Texto

End For

//ordenación

For I=1 To 10 Step 1

For K=I To 10

//si la longitud de la Cadenas[K] es menor, hace

//el intercambio

If(Longitud(Cadenas[K])<Longitud(Cadenas[I])) Then

Aux=Cadenas[I]

Cadenas[I]=Cadenas[K]

Cadenas[K]=Aux

Else

//comprueba si tienen la misma longitud

//en cuyo caso, la ordenación dependerá

//del número de vocales

If(Longitud(Cadenas[K])==

Longitud(Cadenas[I]) Then

//cuenta vocales de Cadenas[I]

Len1=Longitud(Cadenas[I])

For Pos=1 To Len1 Step 1

Letra=ObtenerLetra(Cadenas[I], Pos)

Switch(Letra)

Case “a”, “e”, “I”, “o”, “u”:

Vocales1=Vocales1+1

End Switch

End For

//cuenta vocales de Cadenas[K]

Len2=Longitud(Cadenas[K])

For Pos=1 To Len2 Step 1

Letra=ObtenerLetra(Cadenas[K], Pos)

Switch(Letra)

Case “a”, “e”, “I”, “o”, “u”:

Vocales2=Vocales2+1

End Switch

End For

//compara el número de vocales

//y si Cadenas[K] tiene más, hace

//el intercambio

If(Vocales2>Vocales1) Then

Aux=Cadenas[I]

Cadenas[I]=Cadenas[K]

Cadenas[K]=Aux

End If

End If

End If

End For

End For

//muestra el array ya ordenado

For I=1 To 10 Step 1

Mostrar Cadenas[I]

End For

Fin

Algorimo 2

Inicio

Datos:

Nums 1 To 4, 1 To 6 Integer

Aux, I, K, J, P Integer

Código:

//lee los numerous y los guarda

For I=1 To 4 Step 1

For K=1 To 6 Step 1

Leer N

Nums[i][k]=N

End For

End For

//se necesitan cuatro bucles for

//con los dos primeros, recorremos cada elemento del array,

//que a través de los dos segundos for, vamos comparando

//con cada elemento

For I=1 To 4 Step 1

For K=1 To 6 Step 1

For J=1 To 4 Step 1

For P=1 To 6 Step 1

//con la condición del AND evitamos que se compare

//el elemento con elementos anteriores

If((Nums[J][P]>Nums[I][K]) AND ((J<I) OR (J==I AND P<K )) )Then

Aux= Nums[I][K];

Nums[I][K]= Nums[J][P];

Nums[J][P]=aux;

End If

End For

End For

End For

End For

For I=1 To 4 Step 1

For K=1 To 6 Step 1

Mostrar Nums[I][K]

End For

End For

Fin

## Actividad práctica 10

Algoritmo 1

Inicio

Datos:

Cadenas 1 To 10 String

Texto, Aux, Letra String

Vocales1, Vocales2, Len1, Len2 Integer

I, K, Pos Integer

Código:

Vocales1=0

Vocales2=0

For Pos=1 To 10 Step 1

Leer Texto

Cadenas[Pos]=Texto

End For

//ordenación

For I=1 To 10 Step 1

For K=I To 10

//si la longitud de la Cadenas[K] es menor, hace

//el intercambio

If(Longitud(Cadenas[K])<Longitud(Cadenas[I])) Then

Aux=Cadenas[I]

Cadenas[I]=Cadenas[K]

Cadenas[K]=Aux

Else

//comprueba si tienen la misma longitud

//en cuyo caso, la ordenación dependerá

//del número de vocales

If(Longitud(Cadenas[K])==

Longitud(Cadenas[I]) Then

//Llamada a funciones

Vocales1=ContarVocales(Cadenas[I])

Vocales2=ContarVocales(Cadenas[K])

If(Vocales2>Vocales1) Then

Aux=Cadenas[I]

Cadenas[I]=Cadenas[K]

Cadenas[K]=Aux

End If

End If

End If

End For

End For

//Llamada a la función que muestra el contenido

MostrarContenido(Cadenas)

Fin

Function ContarVocales(Cadena String)

Datos:

Len, I, Vocales Integer

Letra String

Código:

Vocales=0

Len=Longitud(Cadena)

For I=1 To Len Step 1

Letra=ObtenerLetra(Cadena,I)

Switch(Letra)

Case “a”, “e”, “I”, “o”, “u”:

Vocales=Vocales+1

End Switch

End For

Return Vocales

End Function

Sub MostrarContenido(Textos 1 To 10 String)

Datos:

I Integer

Código:

For I=1 To 10 Step 1

Mostrar Textos [I]

End For

End Sub

Algoritmo 2

Funtion Mayor(A Integer, B Integer)

If(A>B) Then

Return A

Else

Return B

End If

End Function

Funtion Menor(A Integer, B Integer)

If(A<B) Then

Return A

Else

Return B

End If

End Function

Function SumaIntervalo(N1 Integer, N2 Integer)

Datos:

Suma, I Integer

Código:

Suma=0

For I=N1 To N2 Step 1

Suma=Suma+I

End For

Return Suma

End Function

Function Factorial(Num Integer)

Datos:

Producto, I Integer

Código:

Producto=1

For I=1 To Num Step 1

Producto=Producto\*I

End For

Return Producto

End Function

//módulo principal

Inicio

Datos:

A, B, Nmayor, Nmenor entero

Código:

Leer A

Leer B

//calcula cual es el mayor y el menor antes de llamar a SumaIntervalo

Nmayor=Mayor(A, B)

Nmenor=Menor(A, B)

Mostrar “Suma de números: “, SumaIntervalo(Nmenor, Nmayor)

Mostrar “Factorial de “ & A, Factorial (A)

Mostrar “Factorial de “ & B, Factorial (B)

Fin

## Actividad práctica 11

Algoritmo 1

En primer lugar, definiremos las funciones que realizar las diferentes operaciones requeridas por el algoritmo:

Function CalcularMayores(Datos 1 To 10 Integer, Num Integer)

Datos:

Total, I Integer

Código:

Total=0

For I=1 To 10 Step 1

If(Datos[I]>Num) Then

Total=Total+1

End If

End For

Return Total

End Function

Function SumaMenores(Datos 1 To 10 Integer, Num Integer)

Datos:

Suma, I Integer

Código:

Suma=0

For I=1 To 10 Step 1

If(Datos[I]<Num) Then

Suma=Suma+Datos[I]

End If

End For

Return Suma

End Function

Function Multipos(Datos 1 To 10 Integer, Num Integer)

Datos:

Total, I, K Integer

Código:

Total=0

K=0 //se utilizará como índice para el nuevo array

//se recorre una primera vez el array para contar cuantos

//múltiplos del número existen

For I=1 To 10 Step 1

If(Datos[I]%Num==0) Then

Total=Total+1

End If

End For

//se declara un nuevo array con el tamaño del total de

//múltiplos del numero

Multi 1 To Total Integer

//recorremos el array principal y trasladamos al nuevo

//los múltiplos encontrados

For I=1 To 10 Step 1

If(Datos[I]%Num==0) Then

K=K+1

Multi[K]=Datos[I]

End If

End For

//devuelve el nuevo array con los múltiplos del número

Return Multi

End Function

//modulo principal del programa

//realiza las operaciones de recogida y presentación de resultados,

//llamando a las funciones definidas anteriormente para

//los cálculos

Inicio:

Datos:

Numeros 1 To 10 Integer

Num, I, Len Integer

//la siguiente variable almacenará

//el array de múltiplos, pero como se desconoce

//su tamaño, no lo indicamos

Multis Integer

Código:

//lee el contenido del array

For I=1 To 10 Step 1

Leer Num

Numeros[I]=Num

End For

//lee el número

Leer Num

//llama a las dos primeras funciones y muestra

//el resultado

Mostrar “Total Mayores: “, CalcularMayores(Numeros, Num)

Mostrar “Suma menores:”, SumaMenores(Numeros, Num)

//llama a la función que devuelve el array de múltiplos

Multis=Multiplos(Numeros, Num)

//obtiene la longitud del array para saber hasta donde recorrer

Len=Multis.length

//recorre el array y muestra su contenido

For I=1 To Len Step 1

Mostrar Multis[I]

End For

Fin

Algoritmo 2

//definimos las funciones que se van a encargar de realizar

//la operativa de programa

Function existe(Textos 1 To 10 String, Valor String, Guardados Integer)

Datos:

Encontrado Boolean

I Integer

Código:

Encontrado=False

For I = 1 To Guardados Step 1

If(Textos[I]==Valor) Then

Encontrado=True

End If

End For

Return Encontrado

End Function

Function DevolverTextos (Textos 1 To 10 String, L1 String, L2 String)

Datos:

I, K, Len Integer

Texto String

//guarda el total de cadenzas que cumple

//la condición

Totales Integer

Código:

Totales=0

//recorre una primera vez el array para saber cuantas cadenzas

//cumplen la condición

For I= 1 To 10 Step 1

Texto=Textos[I]

Len=Longitud(Texto)

//si la primera letra del texto coincide con L1

//y la última con L2, cumple con la condición

If (ObtenerLetra(Texto, 1)==L1 AND

ObtenerLetra(Texto, Len)==L2) Then

Totales=Totales+1

End If

End For

//declara una variable array del tamaño de textos

//encontrados, que será lo que devuelva la función

Encontrados 1 To Totales String

K=0 //posición donde se guarda el encontrado

//vuelve a recorrer el array principal para guardar los textos en el

//array Encontrados

For I= 1 To 10 Step 1

Texto=Textos[I]

Len=Longitud(Texto)

If (ObtenerLetra(Texto, 1)==L1 AND

ObtenerLetra(Texto, Len)==L2) Then

K=K+1

Encontrados[K]=Texto

End If

End For

Return Encontrados

End Function

//programa principal

Inicio

Datos:

I, Guardados Integer

Existe boolean

Tex, Letra1, Letra2 String

Cadenas 1 To 10 String

//array que guardará los textos que cumplan la condición

//no se le asigna tamaño

Recuperados String

Código:

//lleva la cuenta de los textos guardados

Guardados=0

//lee los textos

For I=1 To 10 Step 1

//repite la lectura del texto mientras

//esté repetido

Do

Leer Tex

Existe=Existe(Cadenas, Tex, Guardados)

While(Existe<>True)

//guarda el texto en su posición

Guardados=Guardados+1

Cadenas[I]=Tex

End For

Leer Letra1

Leer Letra2

//recupera los textos que cumplen la condición y los muestra

Recuperados=DevolverTextos(Cadenas, Letra1, Letra2)

For I=1 To Recuperados.length Step 1

Mostrar Recuperados[I]

End For

Fin

## Actividad práctica 12

Algoritmo 1

Class Pila

Property Datos 1 To 10 String

Guardados Integer

Pila()

Guardados=0

End Class

Sub Guardar(Texto String)

If(Guardados<10) Then

Guardados=Guardados+1

Datos[Guardados]=Texto

End If

End Sub

Function Recuperar()

Result String

Result= Datos[Guardados]

Guardados=Guardados-1 //simula su eliminación

Return Result

End Function

End Class

Algoritmo 2

Debemos primeramente crear una clase que encapsule la funcionalidad requerida para la gestión de notas por la aplicación. Esta clase la llamaremos GestionNotas y tendría el siguiente código

Class GestionNotas

//propiedades

Property Notas 1 To 10 Decimal

Property Guardadas Integer

//en el constructor se inicializa la propiedad guardadas a 0

//que contiene las notas guardadas hasta el momento

GestionNotas()

Guardadas=0

End

Sub GuardarNota(Nota Decimal)

//si hay espacio para añadir una nueva nota

//se guarda en el array

If(Guardadas<10) Then

Guardadas=Guardadas+1

Notas[Guardadas]=Nota

End If

End Sub

Function Media()

Datos:

I Integer

M Decimal

Código:

M=0

For I=1 To Guardadas Step 1

M=M+Notas[I]

End For

Return M/Guardadas

End Function

Function Aprobados()

Datos:

I Integer

Ap Integer

Código:

Ap=0

For I=1 To Guardadas Step 1

If(Notas[I]>=5) Then

Ap=Ap+1

End If

End For

Return Ap

End Function

//genera un nuevo array con las notas ordenadas

Function Ordenadas()

Datos:

//array de tamaño igual al de notas guardadas

Ordenadas 1 To Guardadas Decimal

I, K Integer

Aux Decimal

Código:

//traslada las notas al nuevo array

For I=1 To Guardadas Step 1

Ordenadas[I]=Notas[I]

End For

//ordena el Nuevo array

For I=1 To Guardadas Step 1

For K=I To Guardadas Step 1

If(Ordenadas[k]<Ordenadas[I]) Then

Aux=Ordenadas[I]

Ordenadas[I]=Ordenadas[K]

Ordenadas[K]=Aux

End If

End For

End For

Return Ordenadas

End Function

End Class

//Módulo principal

Inicio

Datos:

Gestion GestionNotas //Guarda objeto GestionNotas

//array para recibir las notas guardadas, aunque

//sin tamaño inicial

Totales Decimal

Op Integer

Nota Decimal

Código:

Gestion=New GestionNotas() //crea el objeto GestionNotas

Do

Mostrar “1.- Añadir Nota”

Mostrar “2.- Mostrar Media”

Mostrar “3.- Total Aprobados”

Mostrar “4.- Mostrar notas almacenadas”

Mostrar “5.- Salir”

Leer Op

Switch(Op)

Case 1:

If(Gestion.Guardadas<10) Then

Mostrar “Introduce nota”

Leer Nota

Gestion.GuardarNota(Nota)

Else

Mostrar “No se pueden guardar más notas”

End If

Case 2:

Mostrar “Media : ”, Gestion.Media()

Case 3:

Mostrar “Aprobados: “, Gestion.Aprobados()

Case 4:

//recupera las notas guardadas, ya ordenadas,

//y las muestra

Totales=Gestion.Ordenadas()

For I=1 To Totales.length Step 1

Mostrar Totales[I]

End For

End Switch

While(Op<>5)

Fin

## Actividad práctica 13

Algoritmo 1

La clase Punto sería:

Class Punto

Property X Integer

Property Y Integer

Punto (A Integer, B Integer)

X=A

Y=B

End

Punto (N Integer)

X=N

Y=N

End

Sub Imprimir()

Mostrar X &”,” & Y

End Sub

Sub Cambiar()

Aux Integer

Aux=X

X=Y

Y=Aux

End Sub

End Class

Class Punto3D extends Punto

Property Z Integer

Punto3D (A Integer, B Integer, C Integer):base(A, B)

Z=C

End

//sobrescritura

Sub Imprimir()

Mostrar X &”,” & Y & “,” & Z

End Sub

//sobrecarga

Sub Cambiar(N Integer)

//llamada a método heredado de la superclase

Cambiar()

Z=N

End Sub

End Class

Algoritmo 2

Teniendo en cuenta que la clase Cuenta es como se indica en el siguiente listado:

Class Cuenta

property saldo Decimal

//mediante el constructor se da la posibilidad de inicializar el saldo

Cuenta(s decimal)

saldo=s

End

Sub Ingresar(cantidad Decimal)

saldo=saldo+cantidad

End Sub

Sub Extraer(cantidad Decimal)

saldo=saldo-cantidad

End Sub

End Class

La nueva clase CuentaMovimientos quedaría:

Class CuentaMovimientos extends Cuenta

property Movimientos 1To 5, 1 To 2 String

//indica los movimientos registrados

//no se expone como propiedad

Existentes Integer

CuentaMovimientos(Saldo Decimal):base (Saldo)

Existentes=0

End

//sobrescribe el método ingresar para guardar movimiento

Sub Ingresar(Cantidad Decimal)

Saldo=Saldo+Cantidad

//procedimiento interno de la clase

//que registra los movimientos

RegistrarMovimiento(“ingreso”, cantidad)

End Sub

//sobrescribe el método extraer para guardar movimiento

Sub Extraer(Cantidad Decimal)

Saldo=Saldo-Cantidad

//procedimiento interno de la clase

//que registra los movimientos

RegistrarMovimiento(“extracción”, cantidad)

End Sub

//procedimiento para registro de movimientos

Sub RegistrarMovimiento(Tipo String, Cant Decimal)

//si hay espacio, se guarda en la última posición

//si no hay espacio, se desplazan hacia abajo

//y se pierde el más antiguo

If(Existentes<5) Then

Existentes=Existentes+1

Movimientos [Existentes][1]=Tipo

Movimientos [Existentes][2]=Cant

Else

//desplaza el contenido a la posición anterior

//y el primero se pierde

For K=2 To 5 Step 1

Movimientos [K-1][1]= Movimientos [K][1]

Movimientos [K-1][2]= Movimientos [K][2]

End For

//lo guarda en la última posición

Movimientos [5][1]=Tipo

Movimientos [5][2]=Cant

End If

End Sub

End Class