0\_ Diseño descendente

**0.1¿Qué técnica utiliza el diseño descendente y en qué consiste?**

La técnica que se utiliza se denomina divide y vencerás ya que lo que se realiza es dividir un problema en problemas mas pequeños denominados subproblemas y estos a su vez en otros subproblemas y así sucesivamente.

**0.2 Definir un problema (distinto del teórico) y utilizar la técnica de programación modular para resolverlo.**

DEFINICION DE VARIABLES.

VARIABLES DE ENTRADA

double Pared1, Pared2, Altura;

VARIABLE SALIDA

double SupTot;

METODO ADESARROLLAR

1. Leer la medida de la pared 1.
2. Leer la medida de la pared 2.
3. Leer la medida de la altura.
4. Calcular la superficie de las paredes opuestas 1.
5. Calcular la superficie de las paredes opuestas 2.
6. Calcular la superficie total de la habitación.
7. Visualizar los valores de la superficie a pintar.

REFINAMIENTO (Desarrollo de los modulos)

1. Leer la medida de la pared 1.

Imprimir(“Ingrese la medida de la pared 1”);

Pared1 = LEER();

1. Leer la medida de la Pared 2.

Imprimir(“Ingrese la medida de la pared 2”);

Pared2 = LEER();

1. Leer el valor de la altura.

Imprimir(“Ingrese el valor de la altura”);

Altura = LEER();

1. Calcular la superficie de la pared 1.

SupPared1 = Lado1 \* Altura;

1. Calcular la superficie de la pared 2.

SupPared2 = Lado2 \*Altura;

1. Calcular la superficie total de la habitación.

SupTot = SupPared1 \* 2 + SupPared2 \* 2;

1. Visualizar los valores de la superficie a pintar.

Imprimir(“La superficie total a pintar es:”, SupTot);

**0.3 Definir que son los subprogramas, que tipos de subprogramas hay.**

Con los subprogramas se diseñan los subproblemas o módulos. Los subprogramas se clasifican en **procedimientos y funciones.**

|  |  |
| --- | --- |
| SUBPROGRAMAS | PROCEDIMIENTOS |
| FUNCIONES |

La descomposición de un problema se denomina *descomposición modular* y la programación relativa a ellos se denomina *programación modular.*

**0.4 Definir qué es un procedimiento , cuáles son sus formatos.**

Es un programa que desarrolla una tarea específica, se compone de un conjunto de sentencias a las que se le asignara un nombre. Este procedimiento constituye una unidad de programa.

Consta de tres partes:

Cabecera del procedimiento: Se le proporciona un nombre y si se necesita una lista de parámetros.

Declaración de variables: Estas variables solo serán validas dentro del procedimiento.

Cuerpo del procedimiento: Se declaran las sentencias de acción del procedimiento.

Hay dos clases de formatos:

Formato 1:

void NombreProcedimiento()

{

declaraciones locales a l procedimiento

Conjunto de instrucciones de acción

}

Formato 2:

void NombreProcedimiento(lista de parámetros formales)

{

declaraciones locales a l procedimiento

Conjunto de instrucciones de acción

}

**0.5 Cual es la palabra que identifica un procedimiento y qué significa.**

La palabra que identifica que es un procedimiento es “void”, esto indica que luego de esta palabra lo que sigue es un procedimiento.

**0.6 Cómo se invoca un procedimiento y desde donde se invoca.**

Los procedimientos se llaman desde el programa principal o desde otro modulo, siempre respetando los formatos 1 y 2.

EJEMPLOS

1 PROCEDIMIENTO IMPRIMIR NOMBRE

Imprime Nombre

A) CÓDIGO DEL PROCEDIMIENTO

void ImprimirNombre()

{

imprimir(“HERNAN”)

}

B) INVOCACIÓN DESDE EL PROGRAMA PRINCIPAL

INICIO

ImprimirNombre();

FIN

**0.7 Definir un programa principal que invoque a un subprocedimiento y definir el subprocedimiento.**

void Nombre()

{

imprimir("HERNAN")

}

public void Iniciar()

{

ImprimirNombre();

}

**0.8 Definir que son parámetros formales en un procedimiento, que es una lista de parámetros, que formato tiene.**

Estos parámetros son una copia de los parámetros invocantes, la única diferencia es que los parámetros formales pueden modificar su valor en el procedimiento en cambio los parámetros invocantes no modifican su valor.

La lista de los parametros formales se debe conformar de la siguiente manera:

1 la lista estará entre paréntesis

2 Para cada variable dentro de la lista tiene que estar precedida por su tipo aunque los tipos se repitan

3 Deben separarse unas de otras por coma

EJEMPLO:

void ImprimirSumaDeTresNumeros( int N1 , int N2 ,int N3)

{

Imprimir(N1+N2+N3);

}

**0.9 ¿Cómo se relaciona la lista de parámetros con la invocación del procedimiento?.**

**VARIABLE**

**int X, Y;**

**Multiplicar (int X, int Y)**

Siendo (int X, int Y) los parametros que se invocan al procedimiento Multiplicar

**void Multiplicar(int Num1, int Num2)**

Cuando se inicia el procedimiento se pasan los valores de la variable X a la Num1 y la Variable Y a la variable Num2.

**0.10 Crear un procedimiento que imprima mayor ingresando 3 números enteros.**

Void Mayor(int Num1, int Num2, int Num3)

{

SI (Num1 > Num2)

SI (Num1 > Num3)

Imprimir(“El numero ingresado mayor es:” Num1);

SI NO

Imprimir(“El numero ingresado mayor es:” Num3);

FIN SI

SI NO

SI (Num2 > Num3)

Imprimir(“El numero ingresado mayor es:” Num2);

SI NO

Imprimir(“El numero ingresado mayor es:” Num3);

FIN SI

FIN SI

}

**0.11 Desde el procedimiento principal iniciar leer tres números e invocar al procedimiento del ejercicio anterior y hacer una prueba escritorio**.

VARIABLES

int Numero1, Numero2, Numero3;

public void iniciar()

{

Imprimir(“Ingrese el primer numero”);

Numero1 = LEER();

Imprimir(“Ingrese el segundo numero”);

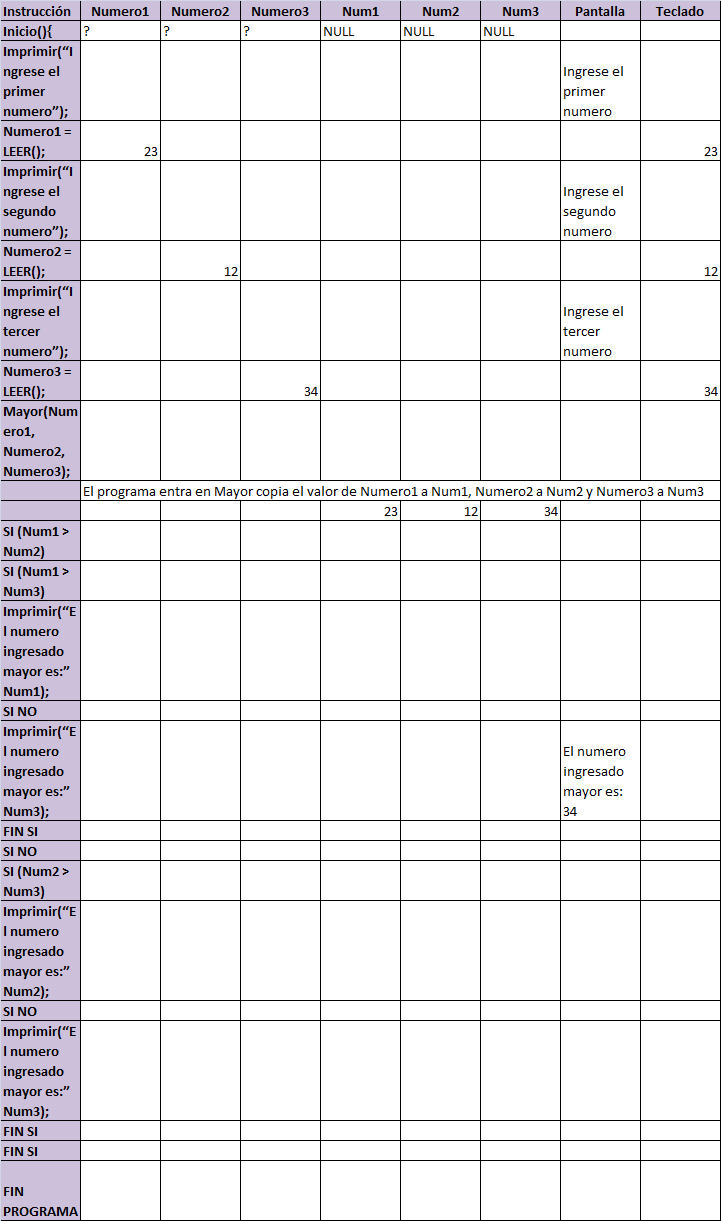
Numero2 = LEER();

Imprimir(“Ingrese el tercer numero”);

Numero3 = LEER();

Mayor(Numero1, Numero2, Numero3)

}



**0.12 Definir que es una variable local y la diferencia con respecto a una variable creada en forma externa a los procedimientos.**

Una variable local es una variable que solo funciona en el procedimiento que se declara, cuando el procedimiento termina las variables desaparecen. Las variables externas pueden ser llamadas desde cualquier procedimiento y desde el programa principal.

**0.13 que es una función y que diferencia hay con un procedimiento, que tipo de dato debe devolvera la clausula RETORNA.**

Una función es un subprocedimiento que devuelve un valor. La diferencia con un procedimiento es que la función devuelve un valor al programa principal. La clausula retorna devolverá el mismo tipo de dato de lafuncioncon que se solicite en el programa principal.

El formato de una función es:

tipo NombreFuncion( lista de parámetros)

{

\*\*\*\*

\*\*\*\*

RETORNA valor del tipo de la función;

}

**0.14 Idear un programa que ingresando los lados de un rectángulo obtenga el perímetro y la superficie del mismo utilizando descomposición modular y aplicando funciones de ingreso de datos y de impresión de perímetro y de superficie agregar un procedimiento que defina si la figura es un cuadrado o un rectángulo, hacer la prueba de escritorio.**

public void iniciar()

{

double Lado1, Lado2;

Imprimir(“Ingrese la medida del primer lado de la figura”);

Lado1 = LEER();

Imprimir(“Ingrese la medida del segundo lado de la figura”);

Lado2 = LEER();

RectOCuadrad(Lado1, Lado2)

Superfic(Lado1, Lado2)

Perimetro(Lado1, Lado2)

}

void RectOCuadrad(double Lad1, double lad2)

{

SI (Lad1 = = Lad2)

Imprimir(“La figura es un cuadrado”);

SI NO

Imprimir(“La figura es un rectángulo”);

FIN SI

}

void Superfic(double Lad1, double lad2)

{

double Superf;

Superf = Lad1 \* Lad2;

Imprimir(“La superficie de la figura es:”,Superf);

}

void Perimetro(double Lad1, double Lad2)

{

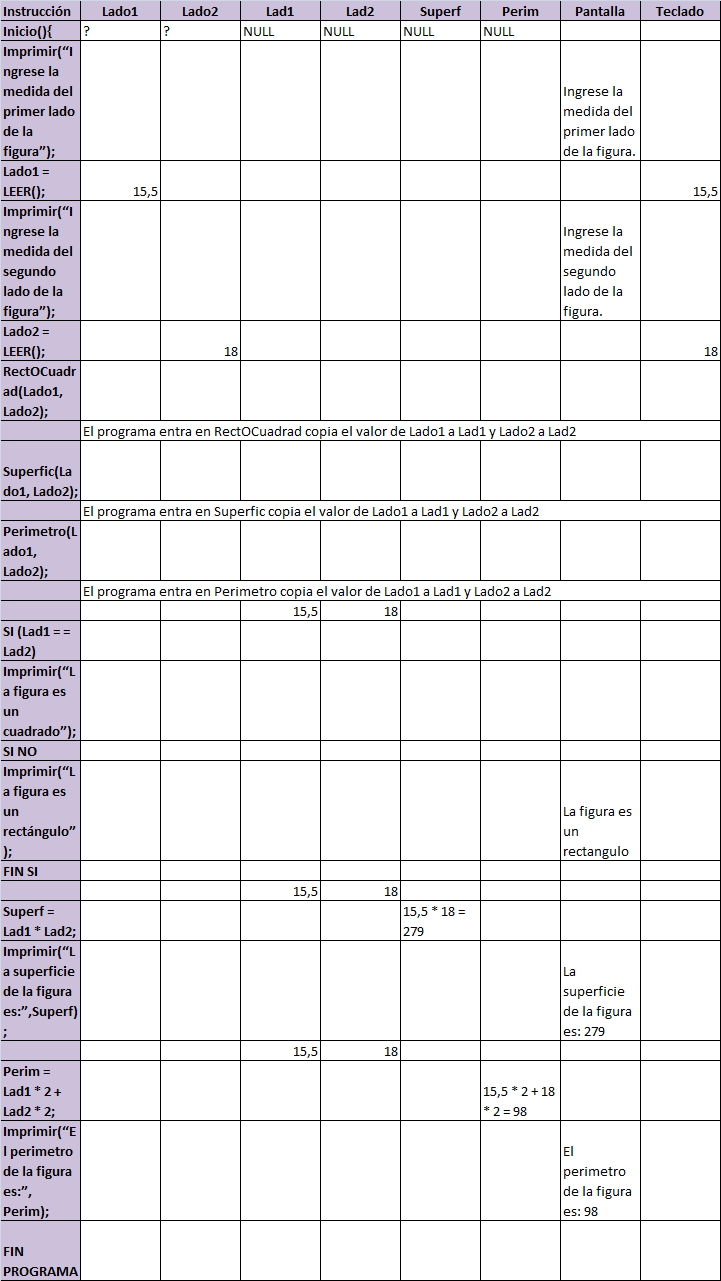
double Perimetro;

Perimetro = Lad1 \* 2 + Lad2 \* 2;

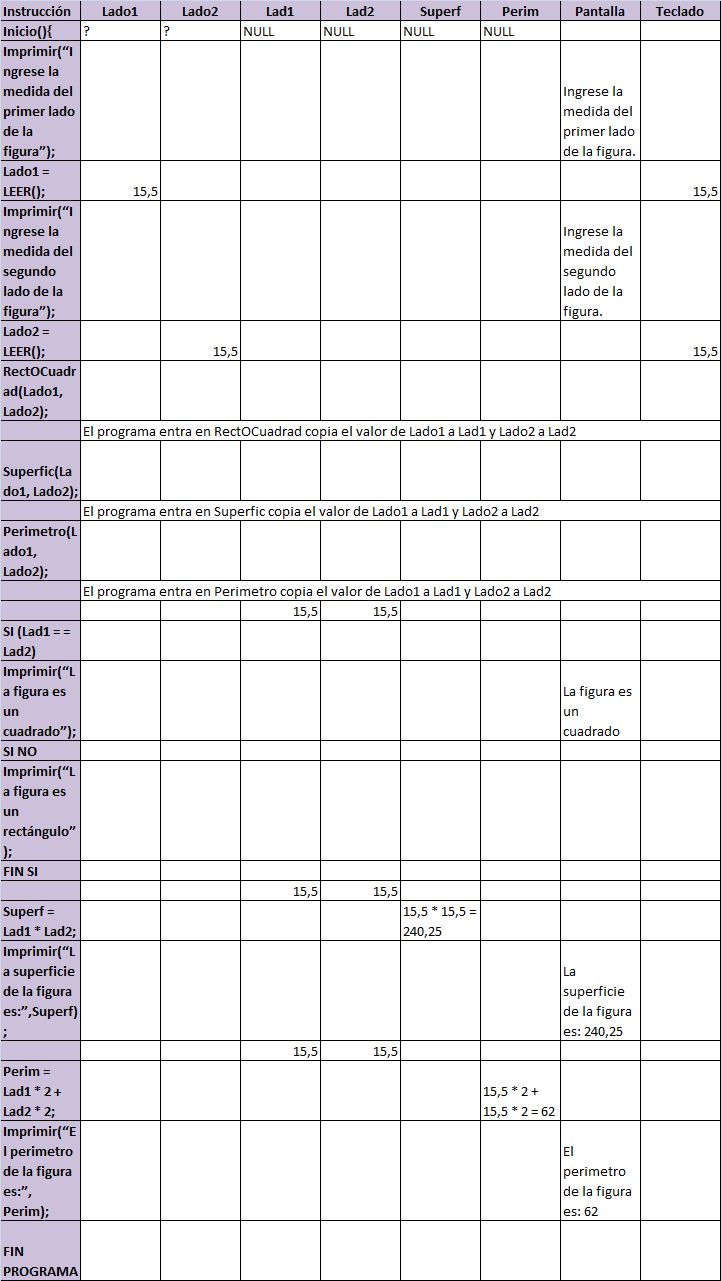
Imprimir(“El perimetro de la figura es:”, Perimetro);

}

Prueba de escritorio para el caso en el que la figura es un rectángulo.



Prueba de escritorio para el caso en el que la figura es un cuadrado.



**1\_ Pseudocodigo en c#**

**1,1 ¿ Qué hay que tener en cuenta para codificar en C#?**

Se tiene que tener en cuenta que existen librerías de procedimientos y funciones que ya están desarrolladas, estas pueden utilizarse sin necesidad de conocer su código, basta que se sepa invocarlas y saber lo que hacen en caso de ser procedimientos o lo que devuelven en caso de ser funciones.

**1.2 ¿Existen algunos procedimiento y funciones desarrolladas en C#, donde están guardadas?**

Están guardadas en la biblioteca de clases del programa Visual Studio.

**1.3 ¿Cual es la instrucción en C# para imprimir en pantalla una variable?**

Para imprimir una variable en C# se utiliza la instrucción:

System.Console.WriteLine(valor);

**1.4 ¿Qué es lo que puede imprimir esta instrucción?**

Lo que puede imprimir puede ser una variable o una expresión de cualquier tipo.

EJEMPLOS

a) System.Console.WriteLine(5\*5+5); //imprime 30

b) System.Console.WriteLine("Hola "+"Hernan"); //imprime Hola Hernan

c) int X=55;

System.Console.WriteLine(X); //Imprime 55

**1.5 ¿Qué tipos de formato de impresión hay?**

El formato es el siguiente:

System.Console.WriteLine(cadena de formato, valor1,valor2....);

EJEMPLOS

a)

int Valor=10;

System.Console.WriteLine( "El valor a Mostrar es {0}",Valor);

//Imprimirá El valor a Mostrar es 10

b)

int Num1=5 , Num2=10;

System.Console.WriteLine(" Operacion de suma: {0} + {1} ={2}", Num1,Num2,Num1+Num2);

esta expresión imprimirá en pantalla

Operación de suma: 5 + 10 = 15

Explicación:

La cadena de formato tiene en su código llaves que encierran números, estos indican los valores (variables o expresiones) que preceden a la cadena de formato. Estas tienen un orden, la número cero {0} es la primera variable precedente a la cadena, la número 1 {1} es la siguiente ,etc.

**1.6 Resolver que es lo que imprime el siguiente código:**

**1.6.1 System.Console.WriteLine("En la casa de mi tía {0} hay una tetera grabada con el nombre de {0}" ,"Juana");**

Imprimirá: **En la casa de mi tía Juana hay una tetera grabada con el nombre de Juana**

**1.6.2**

**int A =10;**

**int B=23;**

**System.Console.WriteLine(" {1} \* {0} = {2}" , A , B , A\*B);**

Imprimirá: **B \* A = A \* B**

**1.6.3** **System.Console.WriteLine( (7+4)/3);**

Imprimirá: **3,666666666666667**

**1.7 Definir de que tipo es la función de lectura del teclado en C# y cuál es el formato de la instrucción.**

El tipo de la función lectura es string. Su invocación se realiza de la siguiente manera:

string Dato= System.Console.ReadLine();

**1.8 Definir la invocación para leer del teclado y cargar en C# las siguientes variables**

1.8.1 int A= int.Parse(System.Console.ReadLine());

1.8.3 double A= double.Parse(System.Console.ReadLine());

1.8.3 string A= System.Console.ReadLine();

1.8.4 bool A= bool.Parse(System.Console.ReadLine());

**2\_ Acciones de secuencia en c#**

**2.1 Leer tres números enteros que deben ser distintos si cuando se lee alguno es igual a algun otro imprimir el mensaje de error y salir en caso de que los tres números sean distintos , definir cuál es el mayor y cual el menor y cual es el promedio**

Utilizar programación modular con pasaje de parámetros ,usando funciones por ejemplo

int Leerentero(string Mensaje)

bool ComparaIguales(int N1,i nt N2)

bool EsMayor(int N1, N2)

Hacer prueba de escritorio

public void iniciar()

{

System.Console.WriteLine(“Ingrese el primer número entero”);

int Num1 = int.Parse(System.Console.ReadLine());

System.Console.WriteLine(“Ingrese el segundo número entero”);

int Num2 = int.Parse(System.Console.ReadLine());

while(Num1 == Num2)

{

System.Console.WriteLine(“Los números deben ser distintos”);

System.Console.WriteLine(“Ingrese el segundo número entero”);

int Num2 = int.Parse(System.Console.ReadLine());

}// **Fin de ciclo mientras**

System.Console.WriteLine(“Ingrese el tercer número entero”);

int Num3 = int.Parse(System.Console.ReadLine());

while(Num1 == Num3) ΙΙ (Num2 == Num3)

{

System.Console.WriteLine(“Los números deben ser distintos”);

System.Console.WriteLine(“Ingrese el tercer número entero”);

int Num3 = int.Parse(System.Console.ReadLine());

}// **Fin del ciclo mientras**

RespMayor(Num1, Num2, Num3)

RespMenor(Num1, Num2, Num3)

Promedio(Num1, Num2, Num3)

}

void RespMayor(int Nume1, int Nume2, int Nume3)

{

int NumMayor=Mayor(Nume1, Nume2)

NumMayor=Mayor(NumMayor, Nume3)

System.Console.WriteLine(“El numero mayor introducido es:”, NumMayor );

}

int Mayor(int Num1, int Num2)

{

If(Num1>Num2)

{

Return Num1;

}

else

{

Return Num2;

}

}

void RespMenor(int Nume1, int Nume2, int Nume3)

{

int NumMenor=Menor(Nume1, Nume2)

NumMenor=Menor(NumMayor, Num3)

System.Console.WriteLine(“El numero menor introducido es:”, NumMenor );

}

int Menor(int Num1, int Num2)

{

If(Num1<Num2)

{

Return Num1;

}

else

{

Return Num2;

}

}

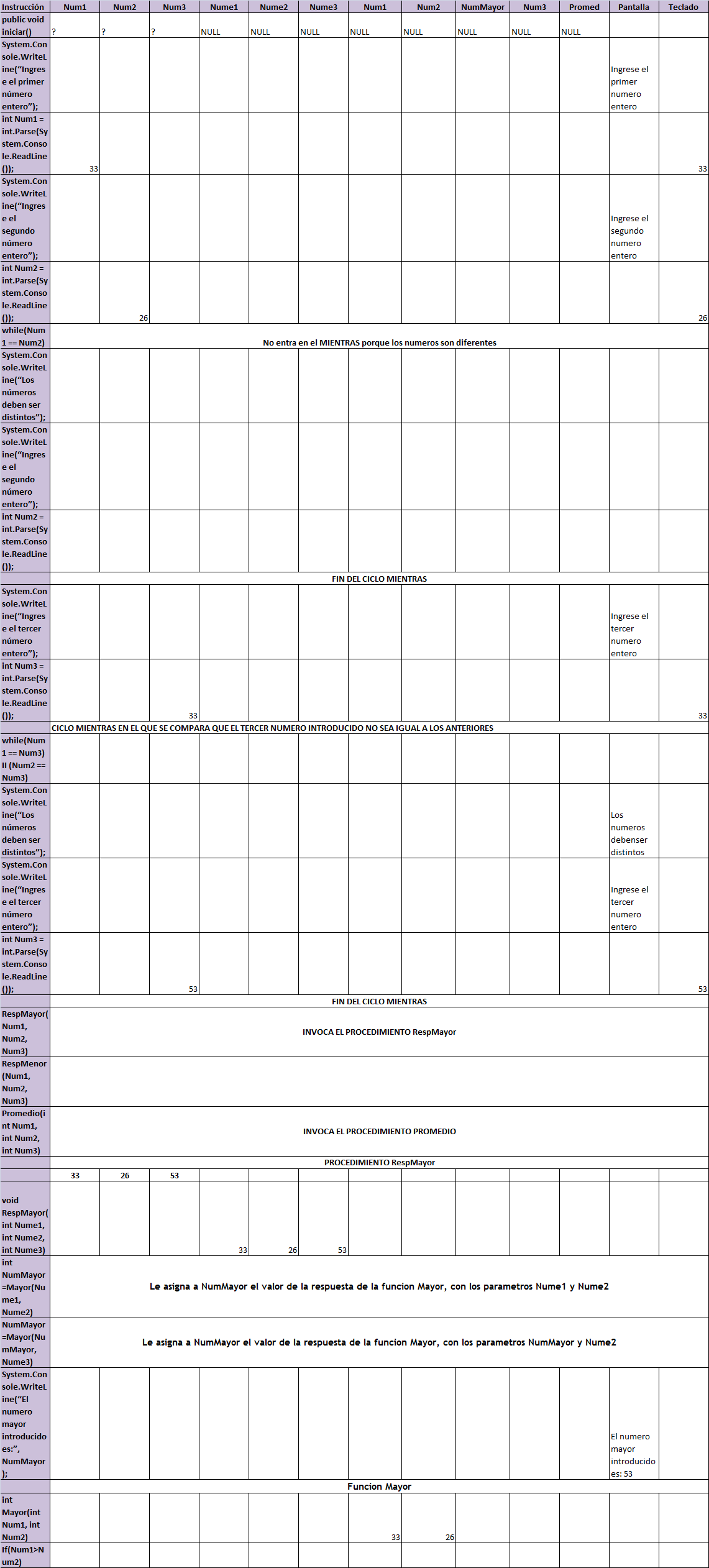
void Promedio(int Nume1,int Nume2, int Nume3)

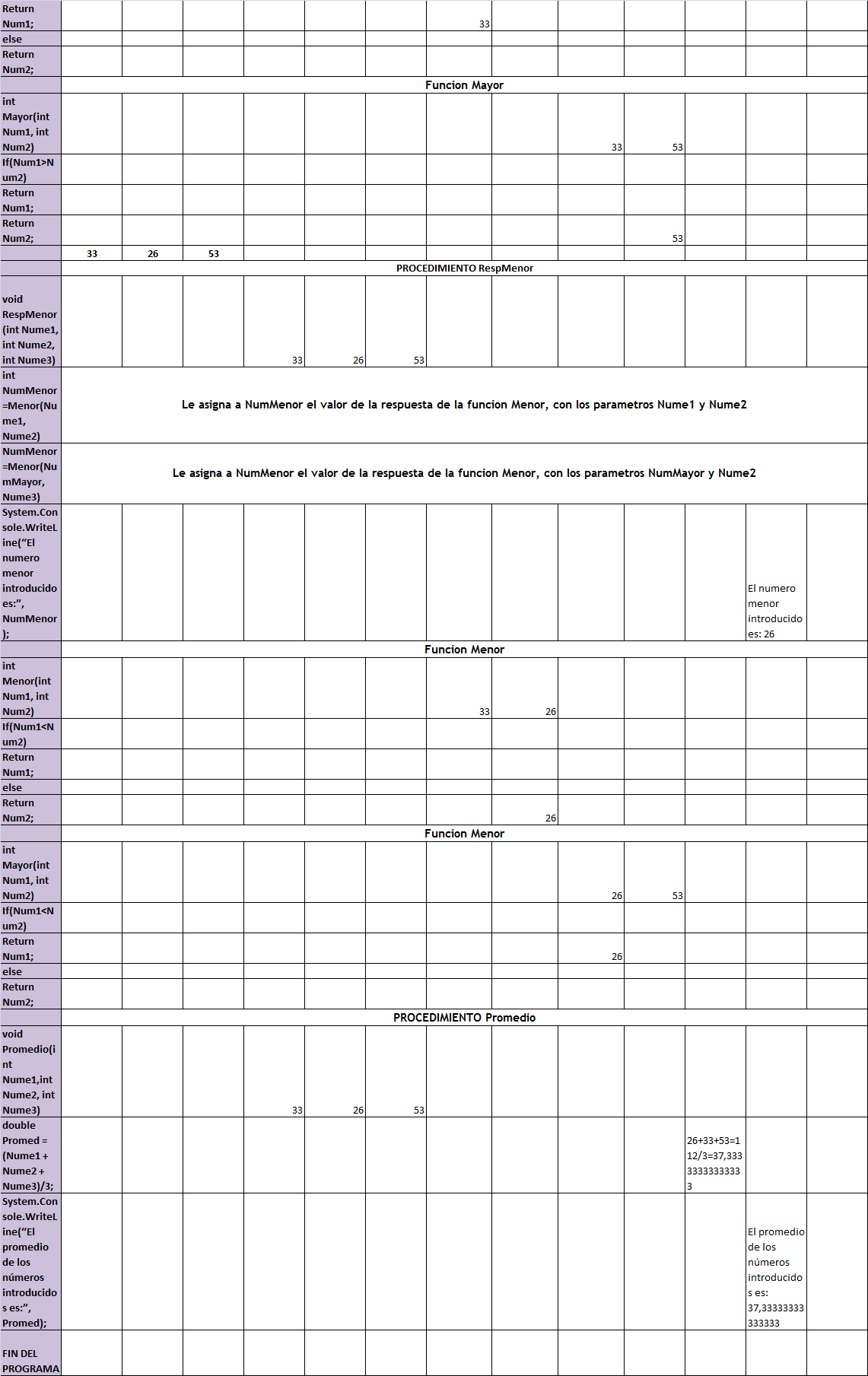
{

double Promed = (Nume1 + Nume2 + Nume3)/3;

System.Console.WriteLine(“El promedio de los números introducidos es:”, Promed);

}





**2.2 Una agencia de alquiler de automóviles sin conductor desea un programa de facturas de sus clientes teniendo en cuenta los siguientes puntos.**

**1 Cantidad fija de 4000 pesos si no rebasan los 300Km.**

**2 Si distancia >300 y <=1000 ademas de la cantidad Fija agregar al los km excedentes a 300 Km, 100$/Km.**

**3 Si la distancia excede los 1000 Km agregar lo calculado hasta 300Km a los excedentes de 1000 Km 70$/Km.**

**Ejemplo si cuenta kilómetros inicial 31000Km y**

**cuenta kilómetros final 32500 Km**

**kilometros recoridos 32500-31000=1500**

**Factura 4000+ 700\*100+350\*70**

**Hacer funciones para LeerEntero Facturarhasta300 FacturrarHasta1000 FactrurarMasDe1000 y aplicarlas con if anidados**

public void iniciar()

{

int KmIniciales, KmFinales;

System.Console.WriteLine(“Ingrese los kilometros iniciales”);

int KmIniciales = int.Parse(System.Console.ReadLine());

System.Console.WriteLine(“Ingrese los kilometros finales”);

int KmFinales = int.Parse(System.Console.ReadLine());

int KmRecorridos=KmRecorr(KmIniciales, KmFinales)

int FacturaHasta1000=KmHasta1000(KmRecorridos)

int FacturaMasDe1000=KmMasDe1000(KmRecorridos)

int FactTotal = KmTot(FacturaHasta1000, FacturaMasDe1000)

System.Console.WriteLine(“Lo que se deberá facturar al cliente es:{0}”,FactTotal);

}

int KmRecorr(int KmInicio, int KmFinal)

{

int KmRecorrid = KmFinal – KmInicio;

Return KmRecorrid;

}

int KmMasDe1000(int KmRec)

{

Int KmMas1000;

if ( KmRec > 1000)

{

KmMas1000 = KmRec – 1000;

Return KmMas1000;

}

else

{

KmMas1000 = 0;

Return KmMas1000;

}

}

int KmHasta1000(int KmRec)

{

int KmHast300;

if ( KmRec > 300)

{

KmHast300 = KmRec – 300;

RETORNA KmHast1000;

}

else

{

KmHast300 = 0;

Return KmHast1000;

}

}

int KmTot(int KmHast1000, int KmMas1000)

{

int Total;

Total=4000 + KmHast1000 \* 100 + KmMas1000 \* 70;

Return Total;

}

**2.3 ingresar Mes y Año año de producción de un producto en forma numérica y emitir el mensaje Fabricado en (Nombre Del Mes) de año o indicar Error en caso que sea un mes que no corresponda**

**Por Ejemplo:Para Mes =1 y Año=2010 ,emitir el mensaje Fabricado en Enero de 2010**

**Para Mes 14 y Año 2010 , emitir el mensaje "14 no corresponde a un mes válido"**

**Hacer una prueba escritorio que contemple ambos casos**

public void inicio()

{

System.Console.WriteLine(“Ingrese el mes de fabricación del producto”);

int Mes = int.Parse(System.Console.ReadLine());

int MesFabric = MesCorrecto(Mes)

string NomMes = NombMes(MesFabric)

System.Console.WriteLine(“Ingrese el año de fabricacion del producto”);

int Año = int.Parse(System.Console.ReadLine());

System.Console.WriteLine(“Fabricado en {0} de {1}”, NomMes, Año);

}

int MesCorrecto(int MesProduct)

{

int MesCorrect;

if (MesProduct <= 12)

{

MesCorrect = MesProduct;

Retorna MesCorrect();

}

else

{

While(MesProduct >12)

{

System.Console.WriteLine(“{0} no es un mes valido”, MesProduct);

System.Console.WriteLine(“Ingrese el mes de fabricación del producto”);

int MesProduct =int.Parse(System.Console.ReadLine());

}

MesProduct = MesCorrect;

Retorna MesCorrect();

}

}

string NombMes(int opcion)

{

string MesProduct

switch(opcion)

{

Case “1”:

MesProduct = Enero;

break;

Case “2”:

MesProduct = Febrero;

break;

Case “3”:

MesProduct = Marzo;

break;

Case “4”:

MesProduct = Abril;

break;

Case “5”:

MesProduct = Mayo;

break;

Case “6”:

MesProduct = Junio;

break;

Case “7”:

MesProduct = Julio;

break;

Case “8”:

MesProduct = Agosto;

break;

Case “9”:

MesProduct = Septiembre;

break;

Case “10”:

MesProduct = Octubre;

break;

Case “11”:

MesProduct = Nobiembre;

break;

Case “12”:

MesProduct = Diciembre;

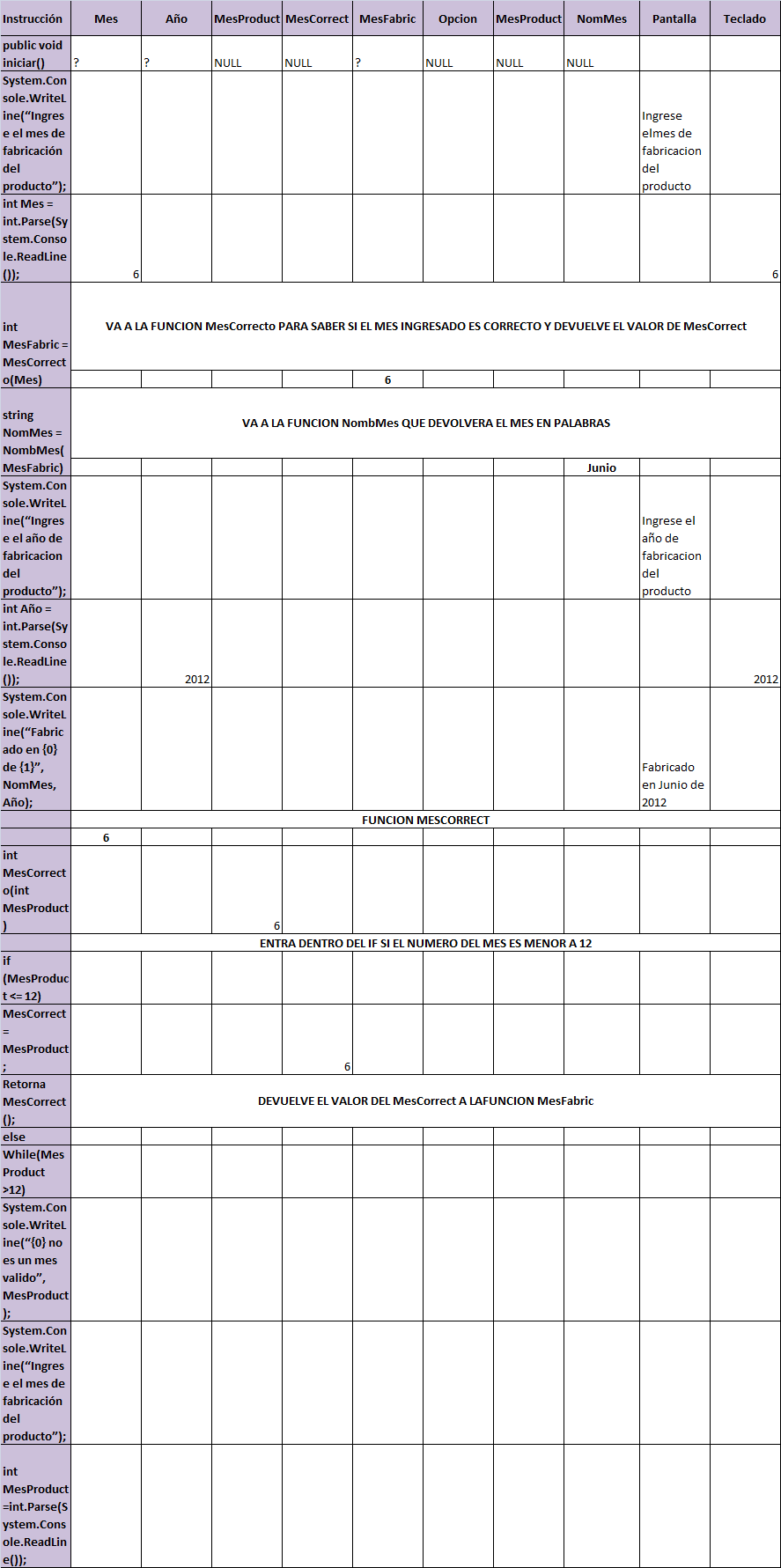
break;

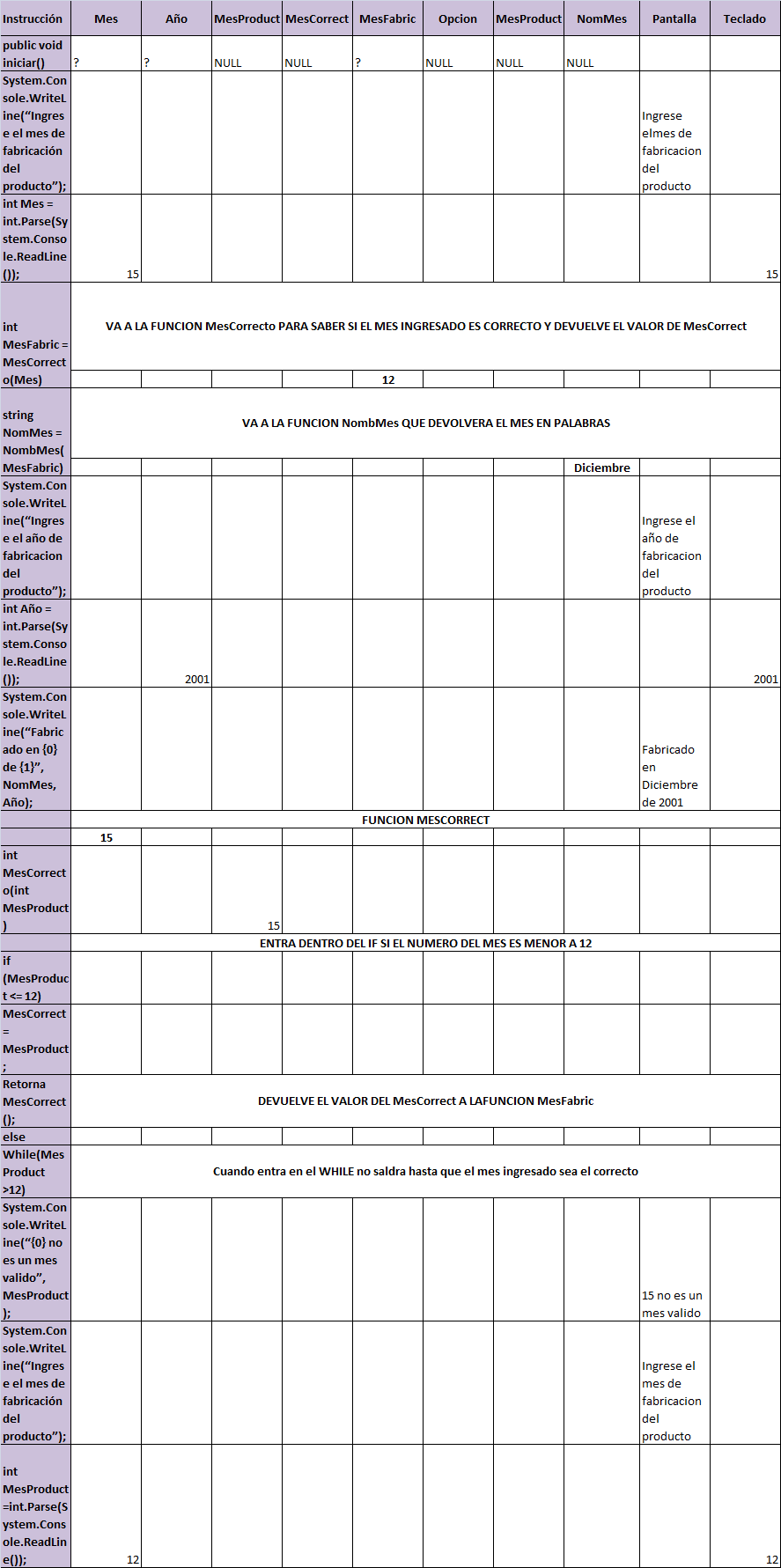
}

Retorna MesProduct();

}

Prueba de escritorio para el caso en el que el me introducido es correcto.

** **

Prueba de escritorio para el caso en el que el mes introducido es incorrecto. ****

****

**2.4\_ Hacer que la máquina adivine el dígito el usuario piensa un dígito de 0 a 9 la maquina pregunta si es par o impar y pide cual es el resto de dividir por 5 y devuelve el digito.**

**Nota el cero es considerado como par**

**Ejemplo pienso en el 8**

**Lamáquina pregunta ¿Es par (Si/No)?**

**la respuesta es Sí (cuatro es par**

**Lamaquina pregunta ¿Cual es el resto de dividir por 5?**

**La respuesta es 3**

**la maquina adivina y responde es un 8**

**los datos se deducen de la siguiente tabla**

|  |  |  |
| --- | --- | --- |
| ***Digito*** | ***Es par*** | ***Resto de***  ***dividir***  ***por 5*** |
| **0** | **Si** | **0** |
| **1** | **No** | **1** |
| **2** | **Si** | **2** |
| **3** | **No** | **3** |
| **4** | **Si** | **4** |
| **5** | **No** | **0** |
| **6** | **Si** | **1** |
| **7** | **No** | **2** |
| **3** | **Si** | **3** |
| **9** | **No** | **4** |

**Ayuda hacer un procedimiento que adivine el valor utilizado un swicht del resto anidando un if y dando el resultado.**

public void iniciar()

{

System.Console.WriteLine(“Piense en un numero del 0 al 9”);

System.Console.WriteLine(“¿Es par o impar?);

string ParOImpar = System.Console.ReadLine();

System.Console.WriteLine(“¿Cuanto es el resto de dividir por 5 el numero pensado?”);

int Resto = int.Parse(System.Console.ReadLine());

int NumPens = int NumeroPensad(ParOImpar, Resto)

System.Console.WriteLine(“El numero pensado es el {0}”, NumPens);

}

int NumeroPensad(string ParImp, int Rest)

{

if (ParImp == par)

{

switch(Rest)

{

Case “0”:

NumAdiv = 0;

break;

Case “1”:

NumAdiv = 6;

breack;

Case “2”:

NumAdiv = 2;

breack;

Case “3”:

NumAdiv = 8;

breack;

Case “4”:

NumAdiv = 4;

breack;

}

}

else

{

switch(Rest)

{

Case “0”:

NumAdiv = 5;

break;

Case “1”:

NumAdiv = 1;

breack;

Case “2”:

NumAdiv = 7;

breack;

Case “3”:

NumAdiv = 3;

breack;

Case “4”:

NumAdiv = 9;

breack;

}

}

Retorna NumAdiv();

}

**3\_Accion iterativa en c#**

### 3.1 Clausula Para en C# (for)

**3.1.1 ¿Cuál es el formato de la cláusula para C#?**

El formato de la clausula For es la siguiente:

for(int identificador = 1; identificador < Cantidad de ciclos; identificador++)

{

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

**3.1.2 Dentro de la cláusula for entre paréntesis se observan 3 sectores, Cómo se separan entre ellos , describir que se hace en cada uno de ellos.**

**Los sectores se separan por ;**

**El primer sector es el de inicializara la variable que controlara el ciclo, se la denomina variable de control. Un ejemplo para inicializar una variable de tipo int seria el siguiente:**

**1) for(int i=0; \*\*\*\*;\*\*\*\*)**

**{**

**\*\*\*\***

**}**

**2) for(int j=3;\*\*\*\*;\*\*\*\*)**

**{**

**\*\*\*\***

**}**

**El segundo sector será la variable de control, en este sector se controla la condición antes de empezar cada ciclo, si no se cumple sale del ciclo.**

**Un ejemplo es que mientras i sea menor que 10 entrara en el ciclo**

**1) for (int i=0;i<10;\*\*\*)**

**{**

**\*\*\*\***

**}**

**En este ejemplo mientras i sea menor igual a 20 entrara en el ciclo**

**2) for(int i=10; i<=20;\*\*\*\*)**

**{**

**\*\*\*\*\***

**}**

**En este sector se incrementara la variable al finalizar cada ciclo, luego de incrementarse se compara con el segundo sector, la variable de control.**

**1) for(int i=0;i<10; i++) //al fin del ciclo se incrementa en 1 a la variable de control**

**{**

**\*\*\*\***

**}**

**2) for (int i=0; i<=30;i+=2) //al fin del ciclo se incrementa en 2 la variable de control**

**{**

**\*\*\*\***

**}**

**3.1.3 ¿Se puede cambiar el valor de la variable de control dentro de un ciclo for, es aconsejable?**

No es aconsejable cambiar el valor de la variable de control dentro de un ciclo for.

**3.1.4 Escribir una función llamada *SumaEscala* , que devuelva la suma =1+2+3+4+.....+N siendo N un valor entero recibido por la función como parámetro.**

**La invocación será por ejemplo**

**int Suma= SumaEscala(100);//dará la suma desde el 1 al 100**

public void inicial()

{

Int ResultSumaEsscala = SumaEscala(“Ingrese el numero hasta donde quiere que llegue la escala”)

}

int SumaEscala(string Mensaje)

{

int Suma, Contador;

Suma = 0;

System.Console.WriteLine(Mensaje);

int Num = int.Parse(System.Console.ReadLine());

for (Contador = 1; Contador<=Num; Num++)

{

Suma = Suma + Contador;

}

Return Suma();

}

**3.1.5 Escribir una función llamada Promedio que lea N números enteros y devuelva el promedio en decimales.**

public void iniciar()

{

double ResultPromedio = Promedio(“Ingrese la cantidad de números para sacar el promedio”);

}

double Promedio(string Mensaje)

{

int Num, Contador, Suma;

double Prom

Suma = 0;

System.Console.WriteLine(Mensaje);

int Num = int.Parse(System.Console.ReadLine());

for( Contador = 1;Contador<=Num;Contador++)

{

System.Console.WriteLine(“Ingrese un numero entero”);

int NumASumar = int.Parse(System.Console.ReadLine());

Suma = Suma + NumASumar;

}

Prom = Suma / Num;

Return Prom();

}

**3.1.6 Escribir una función denominada Armónica que devuelve un double que es la suma.**

**1+1/2+1/3+....+1/N**

Public void iniciar

{

int Max;

System.Console.WriteLine(“Ingrese hasta que numero quiere que se calcule la función”);

int Max = int.Parse(System.Console.ReadLine());

double ResultArmonica = Armonica(Max)

}

double Armonica(int NumMax)

{

int Contador;

double Resultado;

Resultado = 1;

for(Contador = 2; Contador <=NumMax; Contador++)

{

Resultado = Resultado + 1/Contador;

}

Return Resultado();

}

**3.1.7 Escribir una función de calcular la siguiente suma.**

**1/2+2/(2\*2)+ 3/(2\*2\*2)+4/(2\*2\*2\*2)+ ....+n/(2 multiplicado N veces)**

**public void iniciar()**

**{**

System.Console.WriteLine(“Ingrese hasta que numero quiere que se calcule la función”);

int Max = int.Parse(System.Console.ReadLine());

double ResultSuma = Suma(Max)

}

double Suma(int NumMax)

{

int Contador;

double Resultado;

Resultado = 0;

for( Contador = 1; Contador <=NumMax; Contador++)

{

Resultado = Resultado + Contador/(2)Contador;

}

Return Resultado();

}

### 4.2 Clausula mientras

**4.2.1 ¿Cuál es el formato de la cláusula para C#?**

**while(condición de ciclo)**

**{**

**\*\*\*\*\*\*\*\*\***

**}**

**4.2.2 Ingresar un valor M calcular el primer valor N de suma Escala (1+2+3+...N) que supere a o iguale a M.**

**Ejemplos para M=10 1+2+3+4 da 10 por lo que N=4**

**M=11 M1+2+3+4+5 da15 por lo que N=5**

public void iniciar()

{

int Result, N;

N = 0;

Result = 0;

System.Console.WriteLine(“Ingrese el numero que desea calcular”);

int M = int.Parse(System.Console.ReadLine());

while(M < Result)

{

N++;

Result = Result + N;

}

System.Console.WriteLine(“La cantidad de números que se necesitan para completar la escala son: {0}”, N);

}

**4.2.3 Cual es la salida de este programa**

**public void Iniciar()**

**{**

**int X=10;**

**whlie(X>0)**

**{**

**X+=-3;**

**System.Console.WriteLine(X);**

**}**

**}**

Imprimirá el resultado de la variable X, cada vez que entra en el ciclo while se le restara 3 al valor de X y se imprimirá su valor hasta que sea mayor que 0.

1º ciclo

Si comenzamos con X = 10

X+ = -3

Se imprimirá 7

2º ciclo

Como X es mayor que 0 entra en el ciclo

X+=-3

Se imprimirá 4

3º ciclo

Como X es mayor que 0 entra en el ciclo

X+=-3

Se imprimirá 1

4º ciclo

Como X es mayor que 0 entra en el ciclo

X+=-3

Se imprimirá -2

Y saldrá del ciclo ya que el valor de X no es mayor 0.

**4.2.4 Hacer prueba escritorio de la funcion CalcularMonto del ejercicio de la teórica Capitulo3 pagina 33 teniendo N=3 , calcular que valor devuelve.**

**//FUNCION LEERDOUBLE**

**double LeerDouble(string Mensaje)**

**{**

**System.Console.WriteLine( Mensaje);**

**double N1=double.Parse(System.Console.ReadLine());**

**return N1;**

**}**

**//FUNCION LEERENTERO**

**int LeerEntero(string Mensaje)**

**{**

**System.Console.WriteLine( Mensaje);**

**int N1=int.Parse(System.Console.ReadLine());**

**return N1;**

**}**

**//FUNCION CALCULAR MONTO**

**double CalcularMonto( int CantProductos)**

**{**

**//DEFINO E INICIALIZO VARIABLES LOCALES**

**int Contador=0;**

**double MontoAcumulado=0;**

**//REALIZO LA ITERACION**

**while(Contador<CantProductos)**

**{**

**int Cantidad=LeerEntero("Ingrese Cantidad que compra de este producto");**

**double Precio=LeerDouble("Ingrese el precio de este producto");**

**MontoAcumulado += Precio\*Cantidad;**

**//INCREMENTO EL CONTADOR**

**Contador++;**

**} //FIN DEL CICLO**

**//RETORNA EL VALOR CALCULADO DE MONTO ACUMULADO**

**//RETORNO EL MONTOACUMULADO**

**return MontoAcumulado;**

**} //FIN DE CALCULAR MONTO**

### 4.3 Repetir Hasta en C#

**4.3.1 ¿Cuál es el formato de la cláusula para C#?**

**do**

**{**

**\*\*\*\*\***

**}**

**while(condicion de ciclo);**

**4.3.2 Establecer un menú para meteorología que indique estas tres instrucciones:**

**Elija su opción**

**1 Convertir Grados Celsius a Fahrenheit**

**2 Convertir Grados Fahrenheit en Celsius**

**3 Salir del programa**

**este programa hará un ciclo que termina cuando la opción 3 es ingresada**

**ingresando la opción 1 o 2 se pedirá ingresar un numero entero**

**y en el caso 1 la conversión es gradosFarenheit=32 + GradosCelcius\*9/5;**

**en el caso 2 GradosCelcius=GradosFarenheit\*5/9-32**

**en caso de que no sea otro valor emitir un mensaje no corresponde a ninguna opción y seguir el ciclo**

**utilizar Diseño modular**

**public void iniciar()**

**{**

**do**

**{**

**System.Console.WriteLine(“Ingrese su opción”);**

**System.Console.WriteLine(“1-Convertir grados Celsius a Fahrenheit”);**

**System.Console.WriteLine(“2-Convertir grados Fahrenheit a Celsius ”);**

**System.Console.WriteLine(“3-Salir del Proceso”);**

**int Opc = int.Parse(System.Console.ReadLine())**

**S**