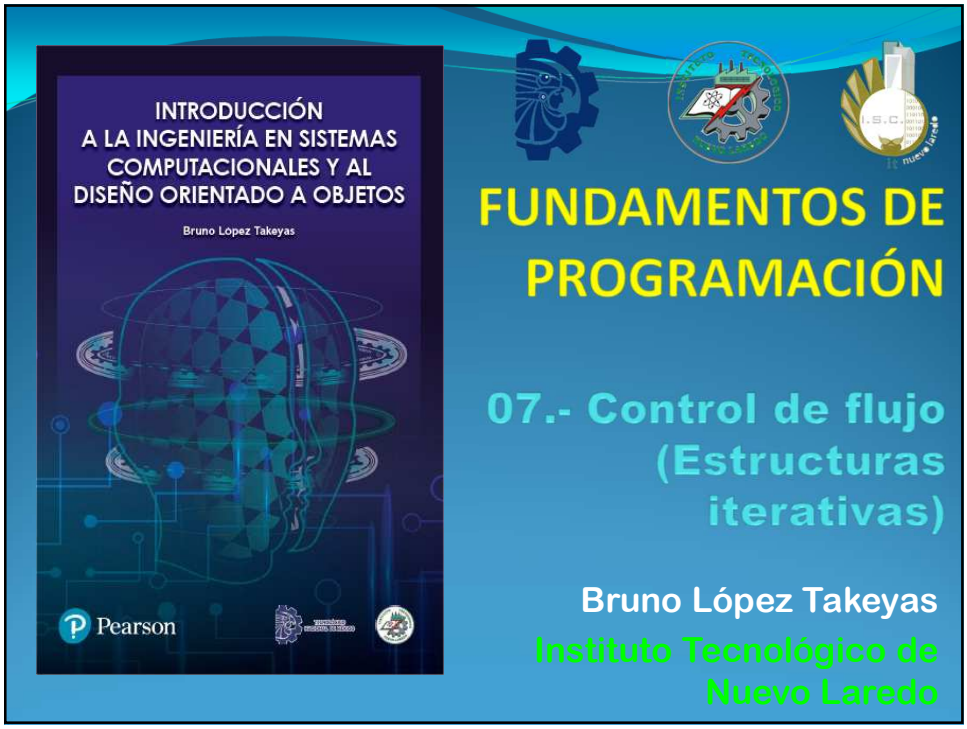

INTRODUCCIÓN
A LA INGENIERÍA EN SISTEMAS
COMPUTACIONALES Y AL
DISEÑO ORIENTADO A OBJETOS
Bruno López Takeyas
Pearson
FUNDAMENTOS DE
PROGRAMACIÓN
07.- Control de flujo
(Estructuras
iterativas)
Bruno López Takeyas
Instituto Tecnológico de
Nuevo Laredo


CAPÍTULO 7
CONTROL DE FLUJO
(Expresiones Iterativas)
2

Preguntas detonadoras

- ¿Qué es una estructura iterativa? ¿Para qué sirve?
- ¿Cuántos tipos existen de estructuras iterativas?
- ¿Cuál es la diferencia entre las diversas estructuras iterativas?
- ¿Cuándo se recomienda utilizar una estructura iterativa de tipo *do-while*?
- ¿... cuándo una de tipo *while*?
- ¿... cuándo una de tipo *for*?
- ¿... cuándo una de tipo *foreach*?

3

Incrementos

- Son contadores ascendentes
- Se representan mediante expresiones de tipo:
  - `x = x + 1`
- Donde al valor actual de la variable `x` se le suma `1` y el resultado se almacena en la misma variable `x`

4

Operador de incremento en C#

- En C# se usa el operador **++** para representar un incremento
- Una expresión algorítmica de incremento como:
  - `x = x + 1`
- se puede representar como:
  - `++x;  // Notación prefijo`
  - `x++;  // Notación postfijo`

5

Ejemplos de incrementos

Incremento	Expresión algorítmica	C#

i = i + 1	`i = i + 1`	`i = i + 1;`
		`i++;`
		`++i;`
		`i+=1;`

y = y + 1	`y = y + 1`	`y = y + 1;`
		`y++;`
		`++y;`
		`y+=1;`

Edad = Edad + 1	`Edad = Edad + 1`	`Edad = Edad + 1;`
		`Edad++;`
		`++Edad;`
		`Edad+=1;`

6

Ejemplo notación prefijo

```
static void Main(string[] args)
{
   int x = 3, y;

   y = ++x; // notación prefijo
   Console.WriteLine("x="+x);  // x = 4
   Console.WriteLine("y="+y);  // y = 4
}
```

7

Ejemplo notación postfijo

```
static void Main(string[] args)
{
   int x = 3, y;

   y = x++; // notación postfijo
   Console.WriteLine("x="+x);  // x = 4
   Console.WriteLine("y="+y);  // y = 3
}
```

8

Decrementos

- Son contadores descendentes
- Se representan mediante expresiones de tipo:
  - `y = y - 1`
- Donde al valor actual de la variable `y` se le resta `1` y el resultado se almacena en la misma variable `y`

9

Operador de decremento en C#

- En C# se usa el operador `--` para representar un decremento
- Una expresión algorítmica de decremento como:
  - `y = y - 1`
- se puede representar como:
  - `--y;  // Notación prefijo`
  - `y--;  // Notación postfijo`

10

Ejemplos de decrementos

Incremento	Expresión algorítmica	C#

i = i - 1	`i = i - 1`	`i = i - 1;`
		`i--;`
		`--i;`
		`i-=1;`

y = y - 1	`y = y - 1`	`y = y - 1;`
		`y--;`
		`--y;`
		`y-=1;`

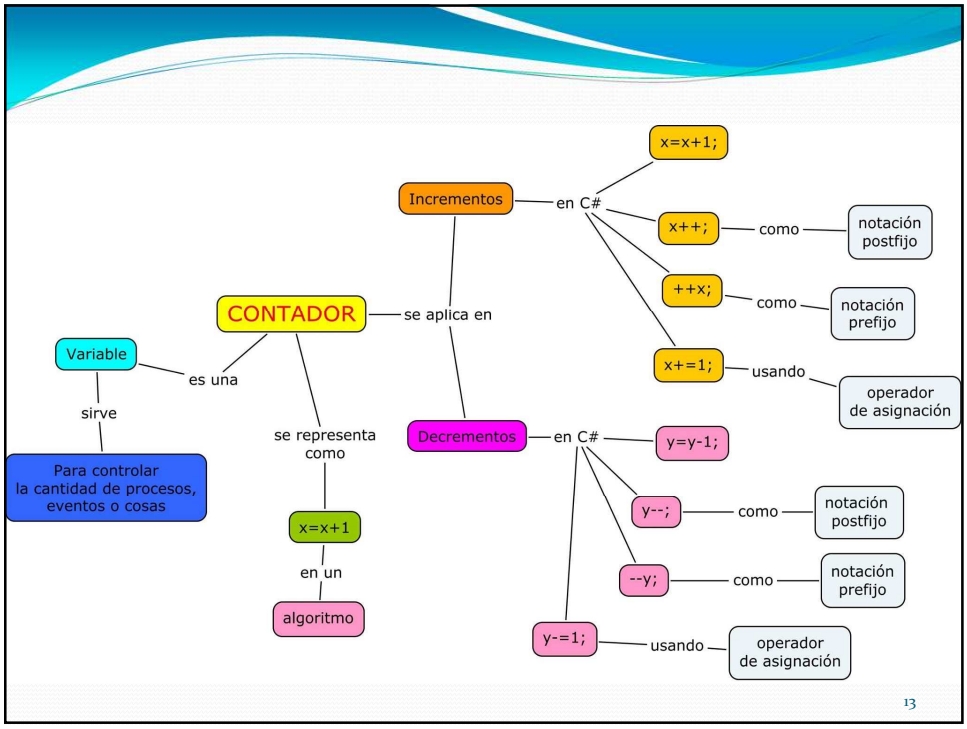
Edad = Edad - 1	`Edad = Edad - 1`	`Edad = Edad - 1;`
		`Edad--;`
		`--Edad;`
		`Edad-=1;`

11

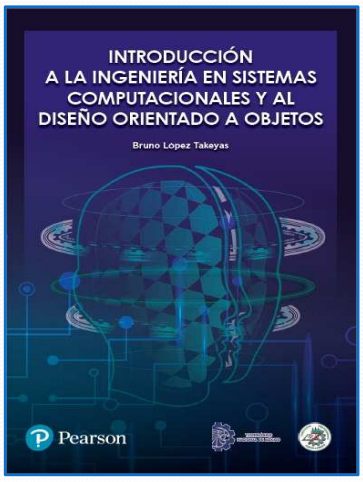
Operadores de asignación en C#

Operador	Descripción	Equivalencia	Ejemplo
`+=`	Suma	`x=x+2;`	`x+=2;`
`-=`	Resta	`y=y-3;`	`y-=3;`
`*=`	Multiplicación	`w=w*4;`	`w*=4;`
`/=`	División	`a=a/7.3;`	`a/=7.3;`
`%=`	Residuo	`b=b%7;`	`b%=7;`
`<<=`	Desplazamiento izquierda (bits)	`r=r<<1;`	`r<<=1;`
`>>=`	Desplazamiento derecha (bits)	`t=t>>2;`	`t>>=2;`
`&=`	And (bits)	`q=q&3;`	`q&=3;`
`|=`	Or (bits)	`f=f|3;`	`f|=3;`

12



Lectura



Para reforzar el tema de ***Contadores*** *se recomienda la lectura de:*

Sección 5.6.- Contadores

- *Incrementos*
- *Decrementos*

14

Acumuladores

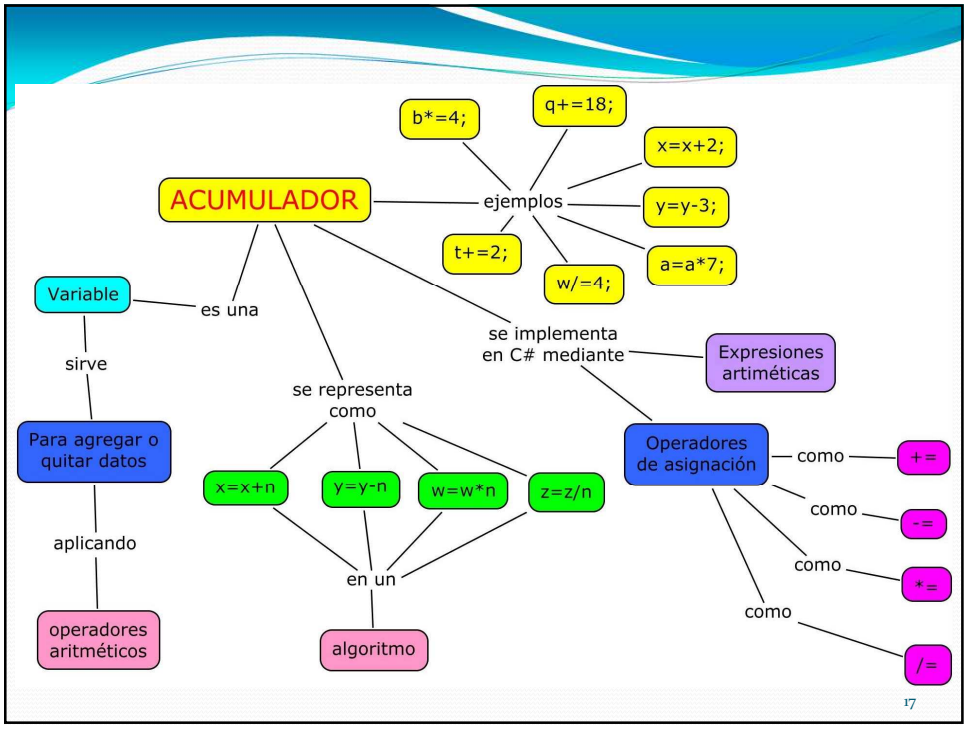
- Los acumuladores son muy semejantes a los contadores, pero con las siguientes diferencias:
- Un contador solamente puede tener operadores de suma o resta y el acumulador puede tener cualquier operador aritmético.
- Un contador solamente puede aumentar o disminuir su valor de uno en uno, mientras que el acumulador puede modificar su valor de n en n.
- Utilizan los operadores de asignación en C#

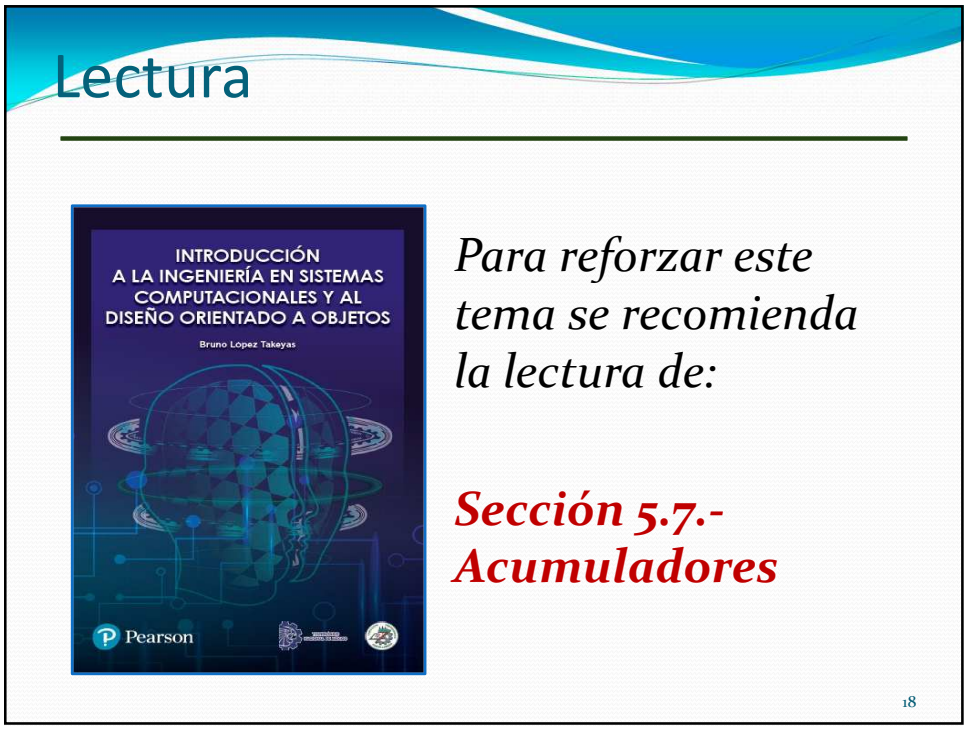
15

Ejemplos de acumuladores

Acumulador	*Expresión algorítmica*	*C#*
i = i + 2	`i = i + 2`	`i = i + 2;` `i+=2;`
y = y – 3	`y = y - 3`	`y = y - 3;` `y-=3;`
a = a * 4	`a = a * 4`	`a = a * 4;` `a*=4;`
w = w / 2	`w = w / 2`	`w = w / 2;` `w/=2;`

16


ACUMULADOR
ejemplos
b*=4;
q+=18;
x=x+2;
y=y-3;
t+=2;
w/=4;
a=a*7;
Variable
es una
sirve
Para agregar o quitar datos
aplicando
operadores aritméticos
se representa como
x=x+n
y=y-n
w=w*n
z=z/n
en un
algoritmo
se implementa en C# mediante
Expresiones artiméticas
Operadores de asignación
como
+=
-=
*=
/=
17


Lectura
INTRODUCCIÓN A LA INGENIERÍA EN SISTEMAS COMPUTACIONALES Y AL DISEÑO ORIENTADO A OBJETOS
Bruno Lopez Takeyas
Pearson
Para reforzar este tema se recomienda la lectura de:
Sección 5.7.- Acumuladores
18

Estructuras iterativas

- Son bloques de código que se repiten cierta cantidad de veces
- Cada vez que se ejecuta el bloque se conoce como "iteración"
- También se conocen como:
  - *Estructuras cíclicas*
  - *Ciclos*

19

Tipos de estructuras iterativas

Tipos de estructuras iterativas en C#

1. *for*
2. *while*
3. *do-while*
4. *foreach* (solamente en C#)

20

Componentes de una estructura iterativa

Componentes de una estructura iterativa

1. Inicialización
2. Condición
3. Incremento/decremento (paso)
4. Cuerpo

21

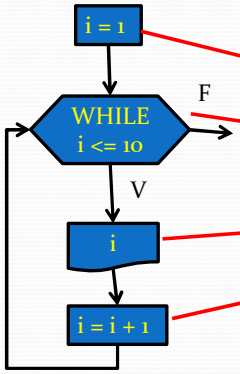
Ciclos en C#

- *while*
- *do – while*
- *for*
- *foreach*

- En C# todos los ciclos son de tipo "**mientras**"; es decir, iteran mientras la condición es **verdadera**
- Abandonan las iteraciones del ciclo cuando la condición es **falsa**

22

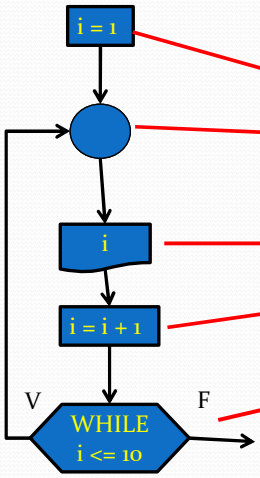
Ciclo *while*




```
i=1;
while(i<=10)
{
   Console.WriteLine(i);
   i++;
}
```

23

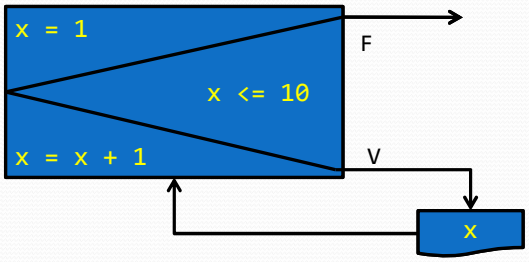
Ciclo *do-while*




```
i=1;
do
{
   Console.WriteLine(i);
   i++;
}
while(i<=10);
```

24

Ciclo *for*




```
for(x = 1; x<=10; x++)
{
   Console.WriteLine(x);
}
```

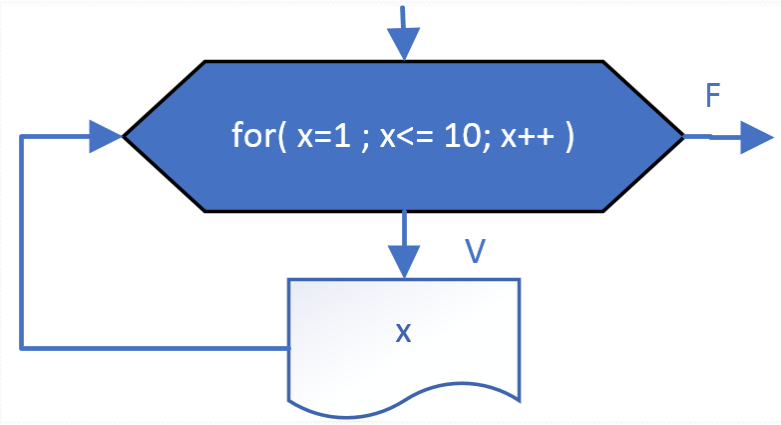
25

Símbolo del ciclo *for* en software

- El símbolo "tradicional" del ciclo *for* no siempre está incluido en software de diseño de diagramas de flujo
- Algunos programas permiten diseñar el símbolo y agregarlo a su biblioteca
- También se puede usar este símbolo:

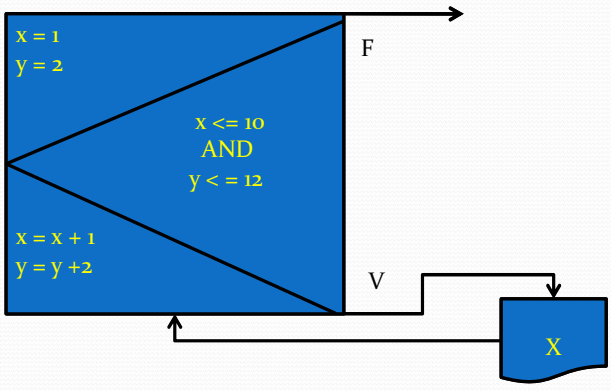
26

Otra forma de representar el ciclo *for*



27

Ejemplo de ciclo *for*




```
for(x = 1, y = 2; x<=10 && y<=12; x++, y+=2)
{
   Console.WriteLine("x = " + x);
}
```

28

Ciclo *foreach*

- Este ciclo es especialmente útil cuando se desea recorrer todos los elementos de una lista, matriz o colección de datos

```
foreach(int x in miListaEnteros)
{
   Console.WriteLine("x = " + x);
}
```

29

Las sentencias *break* y *continue*

Sentencias que afectan la ejecución de ciclos

- `break;` — Provoca la salida inmediata del ciclo
- `continue;` — Provoca la siguiente iteración del ciclo

30

Ejemplo de uso de *break*

```
static void Main(string[] args)
{
    for (int i = 0; i<=10; i++)
    {
       if (i == 5)
          break;
       Console.WriteLine(i);
    }
}
```

```
Salida:

0
1
2
3
4
```

31

Ejemplo de uso de *continue*

```
static void Main(string[] args)
{
   for (int i = 0; i<=10; i++)
   {
      if (i == 5)
         continue;
      Console.WriteLine(i);
   }
}
```

```
Salida:

0
1
2
3
4
6
7
8
9
10
```

¡¡¡ Nótese que **NO** imprime el número **5** !!!

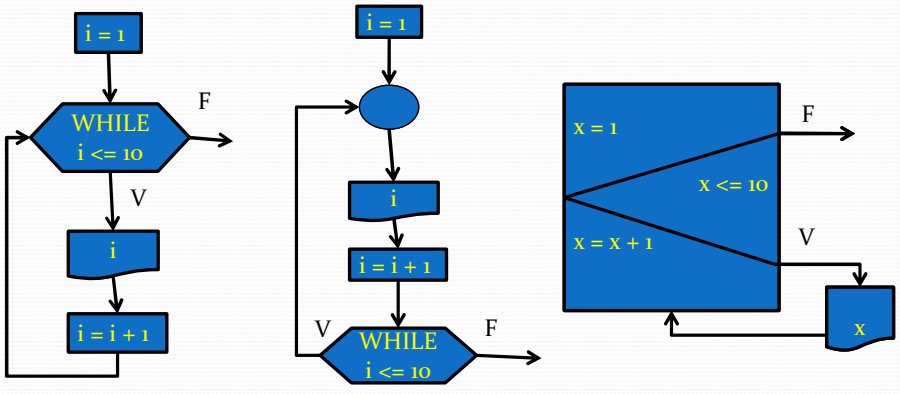
32

Diferencias de los ciclos

- *do – while*. Se recomienda cuando se desea ejecutar el cuerpo al menos `1` vez. Es útil para las validaciones en la captura de datos.
- *while*. Se recomienda cuando se desea ejecutar el cuerpo del ciclo `0 (cero)` o más veces.
- *for*. Es recomendable cuando se conoce la cantidad de iteraciones que ejecutará el ciclo.

33

Diferencias de los ciclos (*cont.*)



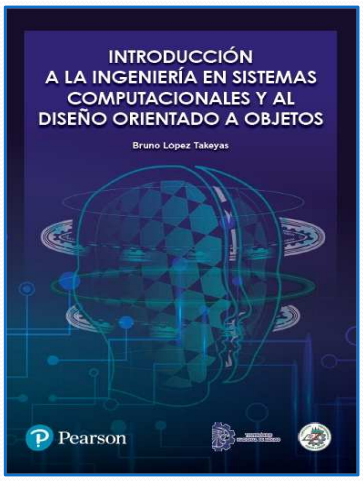
Cero o más veces

Una o más veces

Se conoce la cantidad de iteraciones

34

Lectura



Para reforzar el tema de los ciclos se recomienda la lectura de:

Sección 7.5.- Estructuras iterativas, repetitivas o cíclicas

35

Métodos de la clase *Math*

Método	Descripción	Ejemplo
`Abs(x)`	Valor absoluto	`Abs(-23.7) es 23.7`
`Ceiling(x)`	Redondeo	`Ceiling(9.2) es 10.0`
`Cos(x)`	Coseno (x en radianes)	`Cos(0.0) es 1.0`
`Exp(x)`	Método exponencial e^x	`Exp(1.0) es 2.7182818284509451`
`Floor(x)`	Redondea al valor inmediato menor	`Floor(9.2) es 9.0`
`Log(x)`	Logaritmo natural (base e)	`Log(2.7182818284509451) es 1.0`
`Max(x, y)`	Valor mas grande de x e y	`Max(2.3, 12.7) es 12.7`
`Min(x, y)`	Valor mas pequeño de x e y	`Min(2.3, 12.7) es 2.3`
`Pow(x, y)`	x^y	`Pow(2.0, 3.0) es 8.0`
`Sin(x)`	Seno (x en radianes)	`Sin(0.0) es 0.0`
`Sqrt(x)`	Raíz cuadrada	`Sqrt(9.0) es 3.0`
`Tan(x)`	Tangente (x en radianes)	`Tan(0.0) es 0.0`

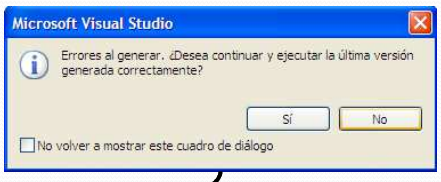
Ámbito de las variables

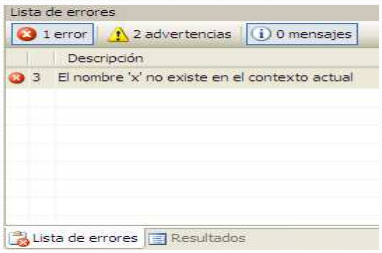
- El ámbito de una variable define dónde puede usarse una variable
- Una variable **local** declarada en un bloque de programa, sólamente puede ser usada en ese bloque
- El ámbito de una variable también aplica a los métodos y a los ciclos

37

Ámbito de las variables

```
for(int x = 1; x<=10; x++)
{
   Console.Write(x);
}

Console.Write(x);
```





38

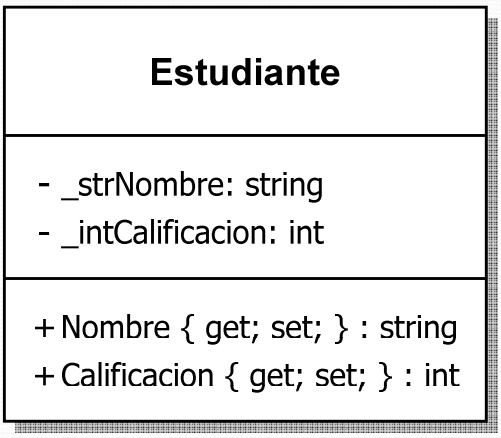
Ejercicio de algoritmo contador

- Un profesor imparte clase a `10` estudiantes y desea capturar el nombre y la calificación de cada uno de ellos. Al finalizar debe imprimir la **cantidad de estudiantes con calificación aprobatoria.**

- NOTAS:
  - *La calificación mínima aprobatoria es `70`.*
  - *Valide que solamente se capturen calificaciones entre `0` y `100`.*
  - *Valide que no se capture el nombre en blanco.*

39

Diagrama de clase



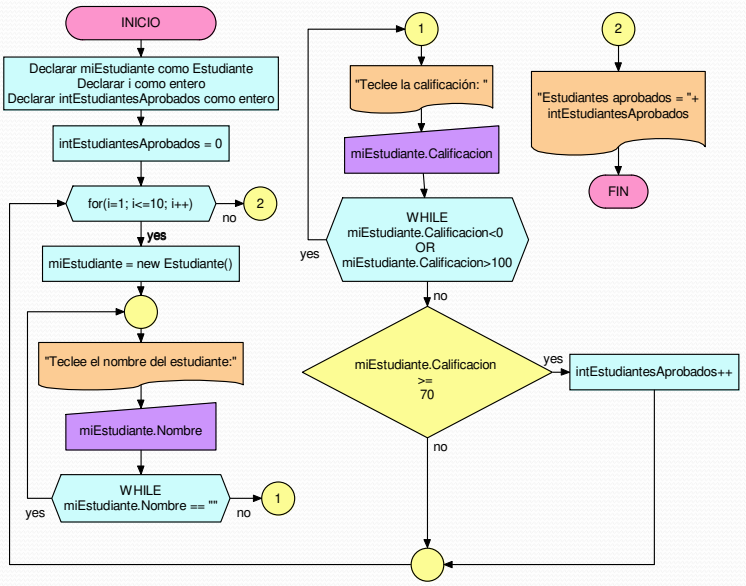
40

La clase Estudiante

```
class Estudiante
    {
        // Atributos privados
        private string _strNombre;
        private int _intCalificacion;

        // Propiedades públicas
        public string Nombre
        {
            get { return _strNombre; }
            set { _strNombre = value; }
        }

        public int Calificacion
        {
            get { return _intCalificacion; }
            set { _intCalificacion = value; }
        }
    }
```

41

Diagrama de flujo



42

El programa principal

```
static void Main(string[] args)
{
    // Declaraciones
    Estudiante miEstudiante;
    int intEstudiantesAprobados = 0;
    for (int i = 1; i <= 10; i++)
    {
        miEstudiante = new Estudiante();

        do
        {
            Console.Write("Teclee el nombre del estudiante: ");
            miEstudiante.Nombre = Console.ReadLine();
        } while (miEstudiante.Nombre=="");

        do
        {
            Console.Write("Teclee la calificación: ");
            miEstudiante.Calificacion = int.Parse(Console.ReadLine());
        } while (miEstudiante.Calificacion<0 || miEstudiante.Calificacion>100);

        if (miEstudiante.Calificacion >= 70)
            intEstudiantesAprobados++;
    }

    // Imprime el resultado
    Console.WriteLine("Estudiantes aprobados = "+intEstudiantesAprobados);
    Console.ReadKey();
}
```

43

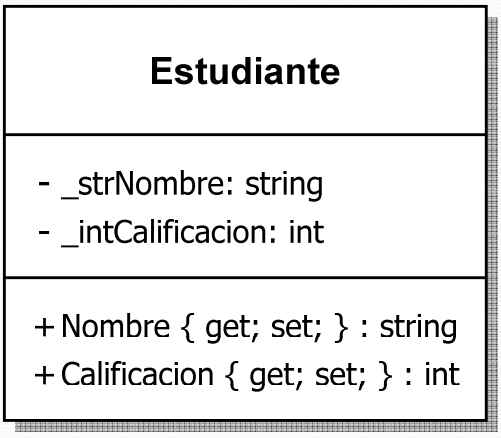
Ejercicio para calcular el promedio

- Un profesor imparte clase a `10` estudiantes y desea capturar el nombre y la calificación de cada uno de ellos. Al finalizar debe imprimir el **promedio de calificaciones.**

- NOTAS:
  - *Valide que solamente se capturen calificaciones entre `0` y `100`.*
  - *Valide que no se capture el nombre en blanco.*

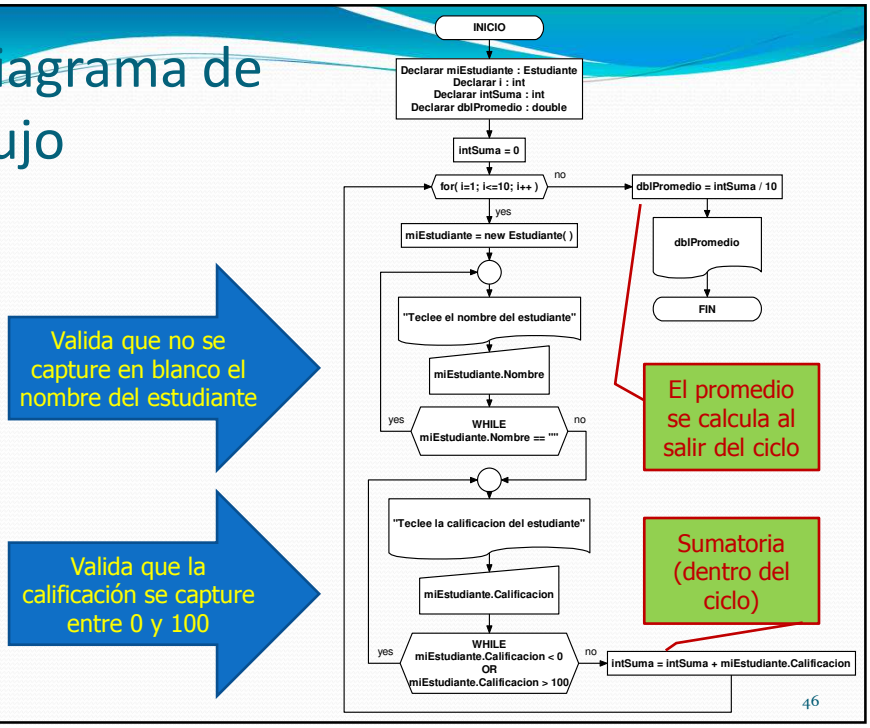
44

Diagrama de clase



45

Diagrama de flujo



46

El programa principal

```
static void Main(string[] args)
{
  Estudiante miEstudiante;
  int i, intSuma = 0;
  double dblPromedio = 0.0;

  for (i = 1; i <= 10; i++)
  {
    miEstudiante = new Estudiante();

    do
    {
       Console.WriteLine("Teclee el nombre del estudiante");
       miEstudiante.Nombre = Console.ReadLine();
    } while (miEstudiante.Nombre=="");

    do
    {
       Console.WriteLine("Teclee la calificación del estudiante");
       miEstudiante.Calificacion = int.Parse(Console.ReadLine());
    } while (miEstudiante.Calificacion<0 || miEstudiante.Calificacion>100);

    // Calcular la sumatoria de calificaciones
    intSuma += miEstudiante.Calificacion;
  }

  // Al salir del ciclo "for" se calcula el promedio
  dblPromedio = (double) intSuma / 10;

  Console.WriteLine("El promedio de calificaciones es : "+dblPromedio);
  Console.ReadKey();
}
```

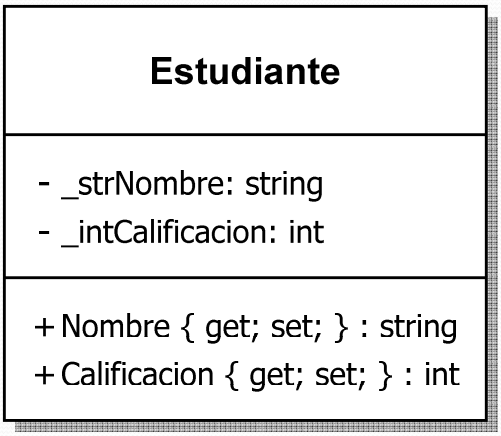
Ejercicio de algoritmo que encuentra el dato mayor

- Un profesor imparte clase a `10` estudiantes y desea capturar el nombre y la calificación de cada uno de ellos. Al finalizar debe imprimir el **nombre del estudiante con la calificación más alta.**

- NOTAS:
  - *Valide que solamente se capturen calificaciones entre `0` y `100`.*
  - *Valide que no se capture el nombre en blanco.*

48

Diagrama de clase



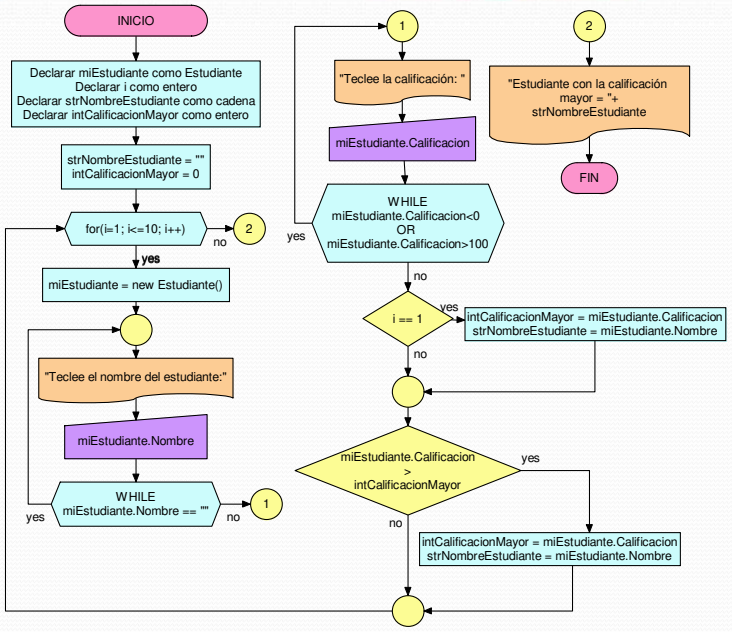
49

Opciones para solucionarlo

- Para controlar el estudiante con la calificación más alta se tienen 2 opciones:
  1. *Usar **variables independientes** para almacenar el nombre del estudiante y su calificación más alta.*
  2. *Usar un **objeto** con los datos del estudiante con la calificación más alta.*

50

Diagrama de flujo (opción 1)



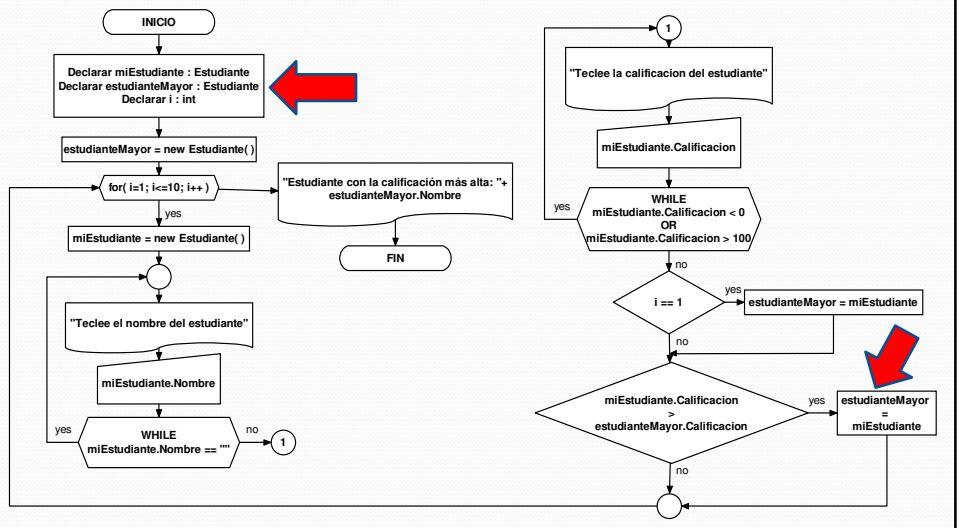
51

El programa principal (opción 1)

```
static void Main(string[] args)
{
   // Declaraciones
   Estudiante miEstudiante;
   string strNombreEstudiante="";
   int intCalificacionMayor=0;

   for (int i = 1; i <= 10; i++)
   {
      miEstudiante = new Estudiante();

      do
      {
         Console.Write("Teclee el nombre del estudiante: ");
         miEstudiante.Nombre = Console.ReadLine();
      } while (miEstudiante.Nombre == "");

      do
      {
         Console.Write("Teclee la calificación: ");
         miEstudiante.Calificacion = int.Parse(Console.ReadLine());
      } while (miEstudiante.Calificacion < 0 || miEstudiante.Calificacion > 100);

      if(i==1)
      {
         intCalificacionMayor = miEstudiante.Calificacion;
         strNombreEstudiante = miEstudiante.Nombre;
      }

      if(miEstudiante.Calificacion>intCalificacionMayor)
      {
         intCalificacionMayor = miEstudiante.Calificacion;
         strNombreEstudiante = miEstudiante.Nombre;
      }
   }

   // Imprime el resultado
   Console.WriteLine("Estudiante con la calificación mayor = "+strNombreEstudiante);
   Console.ReadKey();
}
```

52

Diagrama de flujo (opción 2)



53

El programa principal (opción 2)

```
static void Main(string[] args)
        {
            // Declaraciones de variables
            Estudiante miEstudiante, estudianteMayor;
            int i;

            // Creación del objeto con los datos del estudiante con la calificación más alta
            estudianteMayor = new Estudiante();

            for (i = 1; i <= 10; i++)
            {
                // Creación del objeto miEstudiante
                miEstudiante = new Estudiante();

                // Validar que no deje en blanco el nombre
                do
                {
                    Console.WriteLine("Teclee el nombre del estudiante");
                    miEstudiante.Nombre = Console.ReadLine();
                } while (miEstudiante.Nombre=="");

                // Validar la captura de la calificación entre 0 y 100
                do
                {
                    Console.WriteLine("Teclee la calificación del estudiante");
                    miEstudiante.Calificacion = int.Parse(Console.ReadLine());
                } while (miEstudiante.Calificacion<0 || miEstudiante.Calificacion>100);

                if (i == 1)
                    estudianteMayor = miEstudiante;

                if (miEstudiante.Calificacion > estudianteMayor.Calificacion)
                    estudianteMayor = miEstudiante;
            }

            // Al salir del ciclo "for" se muestra el resultado
            Console.WriteLine("El estudiante con la calificación más alta es "+estudianteMayor.Nombre);
            Console.ReadKey();
        }
```

CUESTIONARIO

Contestar el

Cuestionario 04.- Control de flujo (estructuras selectivas e iterativas)

en la plataforma EaD IT Nuevo Laredo - Synesi

55

TAREA 3.3

Entregar la

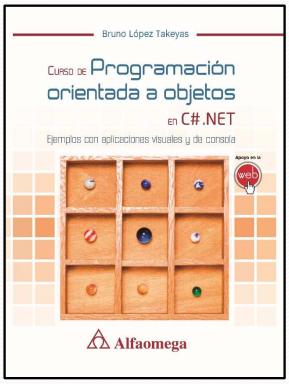
Tarea 3.3.- Estructuras Iterativas

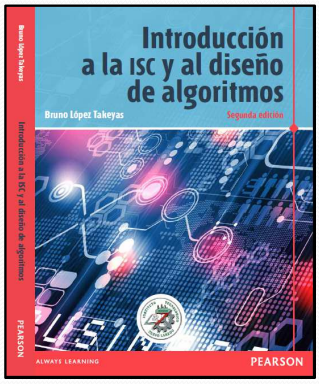
en MS Teams

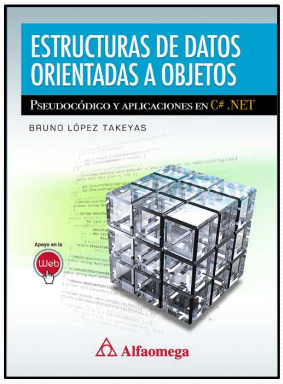
56

Otros libros del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas