UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Estructura de Selección y Repetición en C#

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Septiembre de 2017

Objetivo

Al final de este módulo, los estudiantes serán capaces de:

- Describir los distintos tipos de instrucciones de control.
- Usar instrucciones de salto.
- Usar instrucciones condicionales.
- Usar instrucciones iterativas.

Introducción a las instrucciones

- Una de las cosas más importantes a la hora de utilizar un lenguaje de programación es saber escribir las instrucciones que forman la lógica de un programa en ese lenguaje.
- Un programa consiste en una sucesión de instrucciones. En tiempo de ejecución, estas instrucciones se ejecutan en el orden en que aparecen en el programa, de izquierda a derecha y de arriba a abajo.

Bloques de instrucciones

Se usan llaves para delimitar bloques

```
{
  // code
}
```

Un bloque y su bloque padre o pueden tener una variable con el mismo nombre

```
{
  int i;
  ...
  {
    int i;
    ...
  }
}
```

Bloques hermanos pueden tener variables con el mismo nombre

```
{
  int i;
  ...
}
...
{
  int i;
  ...
}
```

Tipos de instrucciones

Instrucciones Condicionales
Las instrucciones if y switch

Instrucciones de iteración
Las instrucciones while, do, for, y foreach

Instrucciones de salto
Las instrucciones goto, break, y continue

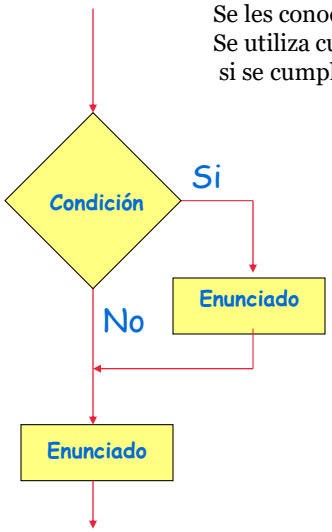
Uso de instrucciones condicionales

Se conocen como instrucciones condicionales, ya que toman decisiones en función del valor de expresiones y ejecutan unos comandos u otros en función de las decisiones tomadas.

- La instrucción if
- Instrucción if en cascada
- La instrucción switch

La instrucción if simple

Se les conoce como "Tomas de decisión"
Se utiliza cuando algunas instrucciones se deben ejecutar, si se cumple una condición y no existe otra alternativa.

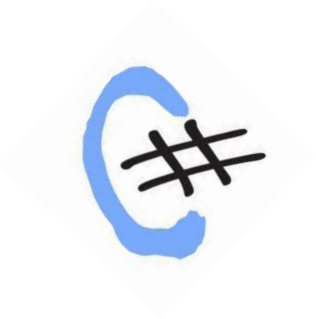


Sintaxis

```
if ( expresión-booleana )
{
    instrucción-incrustada
}
```

PSEUDOCÓDIGO
SI *Expresión-booleana* **ENTONCES**
instrucción-incrustada
FIN_SI

Estructuras C# • FIUAS



Codificación de Algoritmos al Lenguaje.

Construcción del *Algoritmo (Pseudocódigo)*

Objetivo: Determinar el total que debe pagar Ana, por la compra de una computadora.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/septiembre/2015

INICIO

```
//Definición de Variables y Constantes
CONST ENTERO PorcentajeDescuento =15
CONST REAL PrecioAplicarDescuento = 5000
REAL TotalPagar, PrecioComputadora, Descuento
//Lectura de Datos no Conocidos
IMPRIMIR "Teclee el precio de venta de la computadora:$"
LEER PrecioComputadora
```

Estructuras C# • FIUAS

Construcción del *Algoritmo (Pseudocódigo)*

```
//Procesamiento de los Datos
//Hacer que la cantidad a descontar sea cero pesos
Descuento = 0
//Calcular la cantidad a descontar, si el precio de la computadora es
mayor a 5000
SI PrecioComputadora > PrecioAplicarDescuento ENTONCES
   Descuento = PrecioComputadora * ( PorcentajeDescuento / 100)
FIN_SI
//Aplicar el descuento al precio de venta unitario de la computadora
TotalPagar = PrecioComputadora - Descuento
//Impresión de Resultados
IMPRIMIR "Total a pagar:$",TotalPagar
FIN
```

Carácter Coma (,)

Plan de Prueba o verificación del algoritmo

Valores de Entrada	Salidas Esperadas	
PrecioComputadora=2000	TotalPagar=2000	OK
PrecioComputadora=10000	TotalPagar=8500	OK
PrecioComputadora=5000	TotalPagar=5000	OK

Estructuras C# • FIUAS

Actividades

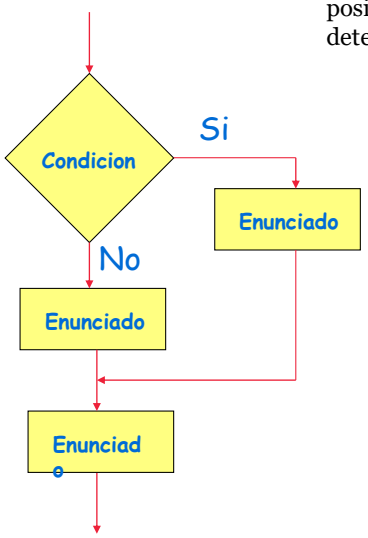
ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar pseudocódigos de diversos problemas, que utilizan procedimientos de solución de toma de decisiones simples.

La instrucción if Doble

Permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición.



Sintaxis C#

```
if ( expresión-booleana )
{
  primera-instrucción-incrustada
}
else
{
  segunda-instrucción-incrustada
}
```

PSEUDOCÓDIGO

SI *Expresión-booleana* **ENTONCES**
    Primera *instrucción-incrustada*
SINO
    Segunda *instrucción-incrustada*
FIN_SI

Ejemplos

Se puede utilizar una instrucción **if**, como las siguientes:

```
if (Numero % 2 == 0)
    Console.WriteLine("Par");
```

```
if (Numero % 2 == 0)
{
    Console.WriteLine("Par");
}
```

```
if (Minuto == 60)
{
    Minuto = 0;
    Hora++;
}
```

```
if (Porciento > 50)
    Console.WriteLine("Pasa");
else
    Console.WriteLine("Error: Fuera del intervalo");
```

Codificación de Algoritmos al Lenguaje.

Construcción del *Algoritmo (Pseudocódigo)*

Objetivo: Determinar el total que debe pagar Ana, por la compra de una computadora.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/Sep/2015

INICIO

```
//Definición de Variables y Constantes
CONST REAL PrecioAplicarImpuesto = 7000.0, PorcentajeImpuesto1=5.3
CONST REAL PorcentajeImpuesto2=15.3
REAL TotalPagar, PrecioComputadora, Impuesto
//Lectura de Datos no Conocidos
IMPRIMIR "Teclee el precio de venta de la computadora:$"
LEER PrecioComputadora
```

Construcción del *Algoritmo (Pseudocódigo)*

```
//Calcular el impuesto, sí el precio de venta es menor a $7,000.00
SI PrecioComputadora < PrecioAplicarImpuesto ENTONCES
   Impuesto = PrecioComputadora * ( PorcentajeImpuesto1 / 100)
SI_NO
      //Calcular el impuesto, sí la condición del paso 1 es falsa,
      // es decir, el precio de venta es $7,000.00 o más.
   Impuesto = PrecioComputadora * ( PorcentajeImpuesto2 / 100)
FIN_SI
//Calcular el total a pagar, sumando el impuesto al precio de la computadora
TotalPagar = PrecioComputadora + Impuesto
//Salida
IMPRIMIR "EL total que Ana debe pagar es:$", TotalPagar
FIN
```

Plan de Prueba o verificación del algoritmo

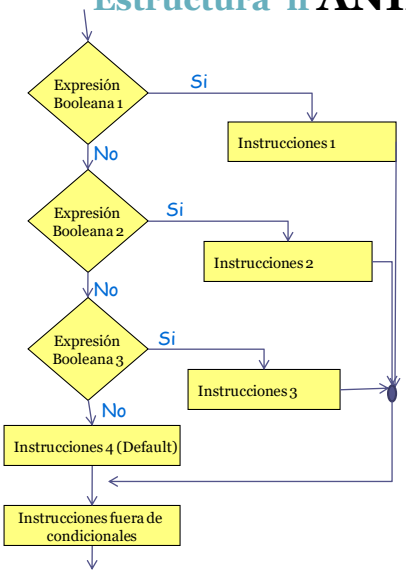
Valores de Entrada	Salidas Esperadas	
PrecioComputadora=2000	TotalPagar=2000	OK
PrecioComputadora=10000	TotalPagar=8500	OK
PrecioComputadora=5000	TotalPagar=5000	OK

Actividades
ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar pseudocódigos de diversos problemas, que utilizan procedimientos de solución de toma de decisiones dobles.

Estructura if ANIDADA



Permiten elegir entre dos o más opciones o alternativas posibles, en función del cumplimiento o no de las Expresiones Booleanas evaluadas.

```
SI expresión-booleana1 ENTONCES
   Instrucciones1 (acciones a realizar) caso true
SI_NO
    SI expresión-booleana2 ENTONCES
        Instrucciones2 (acciones a realizar) caso true
    SI_NO
            SI expresión-booleana3 ENTONCES
               Instrucciones3 (acciones a realizar) caso true
            SI_NO
               Instrucciones (acciones a realizar) caso Default
            FIN_SI
      FIN_SI
FIN_SI
```

Estructura de Selección if Anidada

Sintaxis: C#

```
if ( expresión-booleana1 )
{
   Instrucciones1 (acciones a realizar) caso true
}
else
{
   if ( expresión-booleana2 )
   {
      Instrucciones2 (acciones a realizar) caso true
    }
    else
    {
          if ( expresión-booleana3 )
          {
             Instrucciones3 (acciones a realizar) caso true
          }
          else
          {
             Instrucciones (acciones a realizar) caso Default
           }
     }
}
```


Estructura if en Cascada

Sintaxis: C#

```
if ( expresión-booleana1 )
{
   Instrucciones1 (acciones a realizar) caso true
}
else if ( expresión-booleana2 )
{
     Instrucciones2 (acciones a realizar) caso true
}
 else if ( expresión-booleana3 )
{
     Instrucciones3 (acciones a realizar) caso true
}
else
{
     Instrucciones (acciones a realizar) caso Default
}
```

Ejemplos

El anidamiento de una instrucción **if** dentro de otra puede crear una ambigüedad llamada *dangling else* (else pendiente), como se ve en el siguiente ejemplo:

```
if (Porciento >= 0  &&  Porciento <= 100)
   if (Porciento > 50)
       Console.WriteLine("Pasa");
else
       Console.WriteLine("Error: fuera del intervalo");
```

Solución

```
if (Porciento >= 0 && Porciento <= 100)
{
      if (Porciento > 50)
      {
            Console.WriteLine("Pasa");
       }
}
else
{
      Console.WriteLine("Error: Fuera del intervalo");
}
```


Codificación de Algoritmos al Lenguaje.

Construcción del *Algoritmo (Pseudocódigo)*

Objetivo: Imprimir en pantalla, si un número dado por el usuario es: positivo, negativo o nulo.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/sep/2015

INICIO

```
//Definición de Variables y Constantes
CADENA TipoNumero
ENTERO  Numero
//Lectura de Datos no Conocidos
IMPRIMIR "Proporcione el valor para número:"
LEER Numero
```

Estructuras C# • FIUAS

Construcción del *Algoritmo (Pseudocódigo)*

```
//Determinar el tipo de número
SI  Numero > 0 ENTONCES
        TipoNumero = "Positivo"
 SI_NO
      SI  Numero < 0  ENTONCES
           TipoNumero = "Negativo"
      SI_NO
           TipoNumero = "Nulo"
      FIN_SI
 FIN_SI
 //Imprimir como  resultado el tipo de número
 IMPRIMIR  "El Número  ", Numero, "  es: ", TipoNumero
FIN
```

Actividad: (Plan de Prueba)

Número	Resultado Esperado	Verificación
4	"Positivo"	√
-5	"Negativo"	√
0	"Nulo"	√

Estructuras C# • FIUAS

Actividades
ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar pseudocódigos de diversos problemas, que utilizan procedimientos de solución de toma de decisiones anidadas.

La instrucción switch

- La instrucción **switch** proporciona un mecanismo elegante para expresar condiciones complejas que, de lo contrario, requerirían el uso de instrucciones **if** anidadas.
- Consta de bloques de varios casos, cada uno de los cuales especifica una sola constante y una etiqueta **case** asociada.

Estructuras C# • FIUAS

La instrucción switch

- No está permitido agrupar varias constantes en una sola etiqueta **case**, sino que cada constante debe tener la suya propia
- Un bloque **switch** puede contener declaraciones.
- El ámbito de una constante o variable local declarada en un bloque **switch** se extiende desde su declaración hasta el final del bloque **switch**

Sintaxis de la instrucción switch

```
switch (<expresión>)
{
        case <valor1>: <bloque1>;
                         <siguienteAcción>;
                       break;
        case <valor2>: <bloque2>;
                         <siguienteAcción>;
        ...              Break;

        default:  <bloqueDefault>;
                    <siguienteAcción>;
                  break;
   }
```

Sintaxis Pseudocódigo

```
SEGUN_SEA (Variables a Evaluar)
    CASO VALOR1: ACCIONES1
                 FIN_CASO
    CASO VALOR2: ACCIONES2
                 FIN_CASO
        .
        .
        .
    CASO VALORN:  ACCIONESN
                 FIN_CASO
    CASO DEFAULT: ACCIONES DEFAULT
                 FIN_CASO
FIN_SEGUN_SEA
```


La instrucción switch

- Las instrucciones switch se usan en bloques de varios casos
- Se usan instrucciones break para evitar caídas en cascada (fall through)

```
switch (Digito)
{
   case 0:Nombre="Cero"
          break;
   case 1:Nombre="Uno"
          break;
   case 2:Nombre="Dos"
          break;
   case 3:Nombre="Tres"
          break;
   default:
         Nombre = "No se, intenta con otro Dígito";
          break;
}
```

Ejecución de instrucciones switch

Una instrucción **switch** se ejecuta de la siguiente forma:

1. Si una de las constantes especificada es una etiqueta **case** es igual al valor de la expresión **switch**, el control pasa a la lista de instrucciones que sigue a la correspondiente etiqueta **case**.
2. Si ninguna constante de las etiquetas **case** es igual al valor de la expresión **switch**, y la instrucción **switch** contiene una etiqueta **default**, el control pasa a la lista de instrucciones que sigue a la etiqueta **default**.
3. Si ninguna constante de las etiquetas **case** es igual al valor de la expresión **switch**, y la instrucción **switch** no contiene una etiqueta **default**, el control pasa al final de la instrucción **switch**.

Estructuras C# • FIUAS

Nota:

Una instrucción **switch** sólo se puede utilizar para evaluar los siguientes tipos de expresiones:

- cualquier tipo entero,
- un **char**,
- una **enum** o una **string**.

También es posible evaluar otros tipos de expresiones con la instrucción **switch**, siempre y cuando haya exactamente una conversión implícita definida por el usuario del tipo no permitido a uno de los tipos permitidos.

switch

Puede ser uno de estos tres tipos de instrucciones:

goto case <valori>**;**
goto default;
break;

Si es un **goto case** indica que se ha de seguir ejecutando el bloque de instrucciones asociado en el **switch** a la rama del <valori> indicado, si es un **goto default** indica que se ha de seguir ejecutando el bloque de instrucciones de la rama **default**, y si es un **break** indica que se ha de seguir ejecutando la instrucción siguiente al switch.

Ejemplo:

```
MesActual;
int TotalDias;
...
switch (MesActual)
{
   case "Febrero" :
                    TotalDias = 28;
                    break;
   case "Abril" :
   case "Abril" : // Error: etiqueta duplicada
   case "Septiembre" :
   case "Noviembre" :
                      TotalDias = 30;
                      break;
   default :
   default: // Error: etiqueta duplicada de nuevo
              TotalDias = 31;
              break;
 }
```

Los valores de las constantes en las etiquetas **case** deben ser únicos.

Ejemplo 3

```
string Sufijo = "th";
 switch (Dias % 10) {
    case 1 :
        if (Dias / 10 != 1) {
         Sufijo = "st";
         break;
        }
         // Error: Caída en cascada
    case 2 :
         if (Dias / 10 != 1) {
        Dufijo = "nd";
        break;
        }
        // Error: Caída en cascada
     case 3 :
        if (Dias / 10 != 1) {
         Sufijo = "rd";
        break;
        }
        // Error: Caída en cascada
     default :
        Sufijo = "th";
        // Error: Caída en cascada
 }
```

Estructuras C# • FIUAS

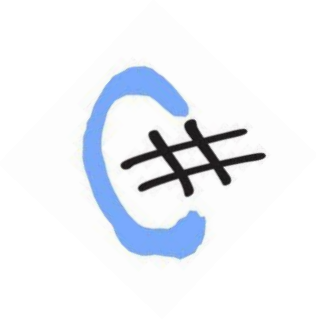
Uso de goto en instrucciones switch

```
switch (Dias % 10)
{
    case 1 :
        if (Dias / 10 != 1) {
        Sufijo = "st";
         break;
        }
        goto case 2;
    case 2 :
        if (Dias / 10 != 1) {
         Sufijo = "nd";
         break;
         }
        goto case 3;
    case 3 :
         if (Dias / 10 != 1) {
         Sufijo = "rd";
         break;
         }
        goto default;
     default :
         Sufijo = "th";
         break;
}
```

Nota para programadores de c++

Para los programadores habituados a lenguajes como C++ es importante resaltarles el hecho de que, a diferencia de dichos lenguajes, C# obliga a incluir una sentencia **break** o una sentencia **goto case** al final de cada rama del **switch** para evitar errores comunes y difíciles de detectar causados por olvidar incluir **break;** al final de alguno de estos bloques y ello provocar que tras ejecutarse ese bloque se ejecute también el siguiente.

Estructuras C# • FIUAS



Codificación de Algoritmos al Lenguaje.

Construcción del *Algoritmo (Pseudocódigo)*

Objetivo: Imprimir en pantalla, el nombre alfabético de un valor(dígito) ingresado.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/sep/2015

INICIO

//Definición de Constantes y Variables
CADENA NombreAlfabetico
ENTERO Digito
//Lectura de Datos no Conocidos
IMPRIMIR "Proporcione el valor del dígito a evaluar:"
LEER Digito

Estructuras C# • FIUAS

Continuación ….

```
//Determinar el tipo de número
SEGUN_SEA ( Digito)
    CASO 0: NombreAlfabetico = "Cero"
            FIN-CASO
    CASO 1: NombreAlfabetico = "Uno"
            FIN-CASO
    CASO 2: NombreAlfabetico = "Dos"
            FIN-CASO
    CASO 3: NombreAlfabetico = "Tres"
            FIN-CASO
    CASO 4: NombreAlfabetico = "Cuatro"
            FIN-CASO
    CASO 5: NombreAlfabetico = "Cinco"
            FIN-CASO
    CASO 6: NombreAlfabetico = "Seis"
            FIN-CASO
    CASO 7: NombreAlfabetico = "Siete"
            FIN-CASO
    CASO 8: NombreAlfabetico = "Ocho"
            FIN-CASO
    CASO 9: NombreAlfabetico = "Nueve"
            FIN-CASO
    CASO DEFAULT: NombreAlfabetico = "Error, el valor ingresado no es Dígito "
                  FIN-CASO
FIN_SEGUN_SEA
 //Imprimir el resultado
IMPRIMIR "El Nombre alfabético de", Digito," es: ", NombreAlfabetico
FIN
```

Actividad:
Verificación (Plan de Prueba)

Número	Resultado Esperado	Verificación
0	"Cero"	√
1	"Uno"	√
2	"Dos"	√
3	"Tres"	√
4	"Cuatro"	√
5	"Cinco"	√
6	"Seis"	√
7	"Siete"	√
8	"Ocho"	√
9	"Nueve"	√
14	"Error, el valor ingresado no es Dígito"	√

Estructuras C# • FIUAS

Actividades
ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar pseudocódigos de diversos problemas, que utilizan procedimientos de solución de toma de decisiones multiples.

Instrucciones Iterativas

while
do
for
foreach

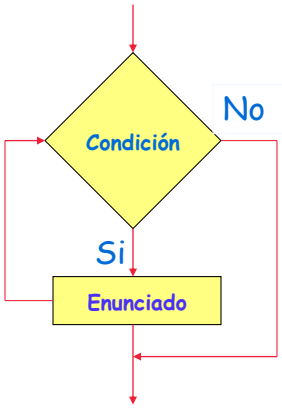
Uso de instrucciones Iterativas

- Se ejecutan repetidamente mientras se cumple una condición. También se conocen como instrucciones de bucle.
- Cada una de estas instrucciones está pensada para un estilo de iteración distinto.
  - La instrucción while
  - La instrucción do
  - La instrucción for
  - La instrucción foreach

La instrucción while

- Ejecuta instrucciones en función de un valor booleano
- Evalúa la expresión booleana al principio del bucle
- Ejecuta las instrucciones mientras el valor booleano sea True

Fujo de ejecución de while



Sintaxis C#:

inicialización
while (*expresión-booleana* **)**
{
  instrucción-incrustada
  actualización
}

```
Definición de Variable de control
. . .
Valor Inicial de Variable
MIENTRAS(Condición ó expresión-booleana)
   instrucciones (acciones a realizar)
   Incremento / decremento de variable
FIN_MIENTRAS
Otras acciones fuera del ciclo
. . .
```

Ejemplo

```
int Indice = 0;
while (Indice < 10)
{
    Console.WriteLine(Indice);
    Indice++;
}
```

0 1 2 3 4 5 6 7 8 9

Estructuras C# • FIUAS

Flujo de ejecución

Una instrucción **while** se ejecuta de la siguiente manera:

1. Se evalúa la expresión booleana que controla la instrucción **while**.
2. Si la expresión booleana se cumple (**true**), el control pasa a la instrucción incrustada. Al llegar al final de la misma, el control se transfiere implícitamente al inicio de la instrucción **while** y se vuelve a evaluar la expresión booleana.
3. Si la expresión booleana no se cumple (**false**), el control pasa al final de la instrucción **while**. Por lo tanto, el programa ejecuta repetidamente la instrucción incrustada mientras la expresión booleana de control sea **true**.

La expresión booleana se prueba al inicio del bucle **while**, por lo que es posible que la instrucción incrustada no se llegue a ejecutar.

Codificación de Algoritmos al Lenguaje.

Objetivo: Imprimir la tabla de multiplicar de un número.
Programador: MC. Gerardo Gálvez G.
Fecha: / / 2015

```
INICIO
   //Definición de Constante y variables
   CONST ENTERO ValorInicial = 1, ValorFinal = 10
   ENTERO Indice, NumeroTabla, Resultado
   //Lectura de Datos
   IMPRIMIR  "Indique el número de tabla de multiplicar a imprimir:"
   LEER NumeroTabla
   //Procesamiento y salida
   Indice  =  ValorInicial
   MIENTRAS( Indice <= ValorFinal)
      Resultado = NumeroTabla * Indice
      IMPRIMIR  NumeroTabla, " X", Indice, "=", Resultado
      Indice = Indice + 1
   FIN_MIENTRAS
FIN
```

Objetivo: Determinar e imprimir la cantidad ahorrada de una persona en 10 años.
Programador: MC. Gerardo Gálvez G.
Fecha: / / 2015

INICIO

```
//Definición de Constante y variables
CONST REAL CantidadDeposito = 1000, TasaInteresMensual = 3
CONST ENTERO  AñosInversion = 10, MesesAño =12
REAL Capital, Interes
ENTERO MesesInversion, Mes
```



```
//Procesamiento: Determinar el total de la inversión
MesesInversion = AñosInversion * MesesAño
Mes = 1
Capital = 0
MIENTRAS( Mes <= MesesInversion)
  Capital = Capital + CantidadDeposito
  Interes = Capital * (TasaInteresMensual / 100)
   Capital = Capital + Interes
   Mes = Mes + 1
FIN_MIENTRAS
//Salida de información
IMPRIMIR "La cantidad ahorrada seria de:$", Capital
```

FIN

Actividades
ExtraClases

Objetivo:

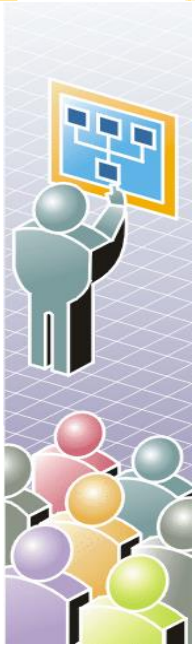
El alumno realizar los siguientes puntos por cada problema:

1. Análisis del problema
2. PSeudocódigo
3. Plan de prueba y verificación
4. Codificación en el lenguaje de programación

55

Actividad #2: Desarrollo de algoritmos para:

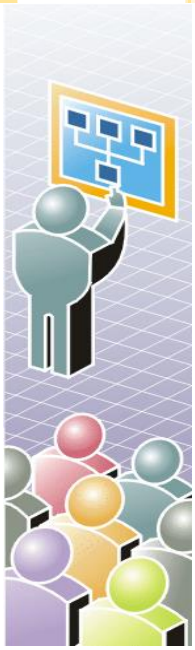
1. Calcular el factorial de un número entero.
2. Elevar un número real a una potencia entera e imprimir el resultado respectivo.
3. Calcular el valor máximo de un conjunto de *N número reales ingresados por el usuario.*



Continuación de Actividad #2:

Tomando como referencia a la **materia de Matemáticas discretas**, diseñe un algoritmo para plantear una solución algorítmica computacional a cada caso.

1. Decimos que un número es simpático, si todos sus dígitos son impares. ¿Cuántos números simpáticos de seis dígitos hay y cuales son?
2. ¿Cuántos números hay menores que 100 que no tienen los dígitos 0 ni 1 y cuales son?
3. Usted deposita $1000 en su cuenta de ahorro que da un 11% de interés anual. ¿Cuánto tendrá en 30 años?
4. ¿Cuántos y cuales números de tres cifras existen que son múltiplos de dos?



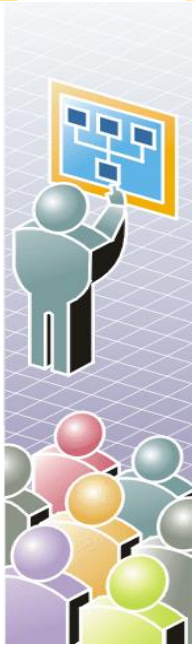
Estructuras C# • FIUAS

Continuación de Actividad #2:

1. Se tienen un grupo de Alumnos, por cada uno se solicita los siguientes datos:
   - Nombre,
   - Sexo(F/M),
   - Edad,
   - Estatura,
   - Peso,
   - Color de Ojos(Azul, Verde, Otro),
   - Color de Cabello(Castaño, Rubio, Otro).

Elabore un algoritmo que lea los datos de todos los alumnos del grupo y obtenga la siguiente información:

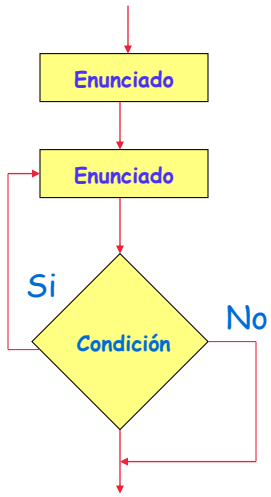
- Total de Mujeres de cabello rubio y ojos azules, que miden entre 1.65m y 1.75m y que pesen menos de 55kg.
- Total de Hombres de ojos castaños de más de 1.70m de estatura y pesen entre 70 y 90 kg



La instrucción do

- Ejecuta instrucciones en función de un valor booleano
- Evalúa la expresión booleana al final del bucle
- Ejecuta las instrucciones mientras el valor booleano sea True

La instrucción do



Sintaxis C#:

inicialización
do {

<instrucciones>

} while(<condición>**);**

```
Definición de Variable de control
. . .
Valor Inicial de Variable
REPETIR
    instrucciones (acciones a realizar)
    Incremento / decremento de variable
MIENTRAS(Condición ó expresión-booleana)
Otras acciones fuera del ciclo
. . .
```

Ejemplo

```
int Indice = 0;
do{
    Console.WriteLine(Indice);
    Indice++;
}while (Indice < 10);
```

0 1 2 3 4 5 6 7 8 9

Flujo de ejecución

Una instrucción **do** se ejecuta de la siguiente manera:

1. El control pasa a la instrucción incrustada.
2. Al llegar al final de la instrucción incrustada, se evalúa la expresión booleana.
3. Si la expresión booleana se cumple (**true**), el control pasa al inicio de la instrucción **do**.
4. Si la expresión booleana no se cumple (**false**), el control pasa al final de la instrucción **do**.

Actividades
ExtraClases

Objetivo:

El alumno realizar los siguientes puntos por cada problema:

1. Análisis del problema
2. PSeudocódigo
3. Plan de prueba y verificación
4. Codificación en el lenguaje de programación

63

Estructuras C# • FIUAS

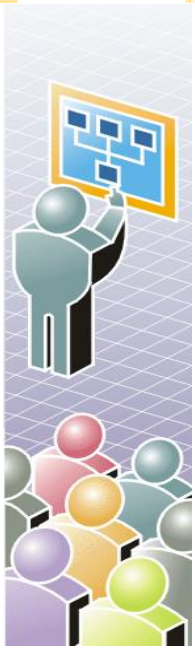
Actividades de Aprendizaje

Cuando la estructura **do-while** es más conveniente, cuando se requiere que un conjunto de instrucciones se ejecuten al menos una vez, antes de decidir se ejecutaran de nuevo.

Actividades: Desarrollo de algoritmos para:

1. Elabore un algoritmo que solicite la lectura de una calificación, y valide que el valor proporcionado este entre 0-10.
2. Realizar un algoritmo que solicite N números. Posteriormente debe imprimir cuantos números de los introducidos fueron positivos y cuantos números fueron negativos.
3. Elabore un algoritmos que encuentre todos los enteros X entre 0 y M, los cuales satisfagan:
    1. X MOD 7 = 1,
    2. X MOD 11=6 y
    3. X MOD 13 = 5.

   Para M leída desde el teclado.
4. Elaborar un algoritmo que simule las operaciones que realiza un cajero en una caja de un supermercado solicitando la cantidad y precio de cada articulo, y determine e imprima el total que el cliente debe pagar.



Estructuras C# • FIUAS

La instrucción for

Sintaxis:

```
for ( inicialización ; condición ; actualización )
{
    instrucción-incrustada
}
```

Como en las demás instrucciones iterativas, la condición en un bloque **for** debe ser una expresión booleana que funciona como condición para la continuación, no para la terminación.

La instrucción for

- La información de actualización está al principio del bucle

```
for (int Indice = 0; Indice < 10; Indice++)
{
    Console.WriteLine(Indice);
}
```

0 1 2 3 4 5 6 7 8 9

Estructuras C# • FIUAS

Explique que pasa?

```
for (;;) {

  Console.WriteLine("Ayuda ");
   ...
}
```

Declaración de variables

Una sutil diferencia entre las instrucciones **while** y **for** es que una variable declarada en el código de inicialización de una instrucción **for** sólo tiene validez dentro de ese bloque **for**.

Por ejemplo:

el siguiente código generará un error en tiempo de compilación:

```
for (int Indice = 0; Indice < 10; Indice++)
{
  Console.WriteLine(Indice);
}
Console.WriteLine(Indice); // Error: i está fuera de ámbito
```


Ejemplos

Error en tiempo de compilación:

```
int Indice;
for (int Indice = 0; Indice < 10; Indice++)
/ Error: Indice ya está en ámbito
```

Por el contrario, el siguiente código sí está permitido

```
:
for (int Indice = 0; Indice < 10; Indice++)
...

for (int Indice = 0; Indice < 20; Indice++)
...
```

Por otra parte, es posible inicializar dos o más variables en el código de inicialización de una instrucción **for**:

```
for (int Indice1= 0, Indice2 =10; ... ; ...)
```

Ejemplo

Sin embargo, las variables tienen que ser del mismo tipo. Por lo tanto, el siguiente código no está permitido:

```
for (int Indice1=0, long Indice2=0; Indice1 < 10; Indice1++)
...
```

También se pueden usar dos o más expresiones separadas por una o más comas en el código de inicialización de una instrucción **for**:

```
for (int Indice1=0, Indice2=0; ... ; Indice1++, Indice2++)
```

Estructuras C# • FIUAS

Actividades

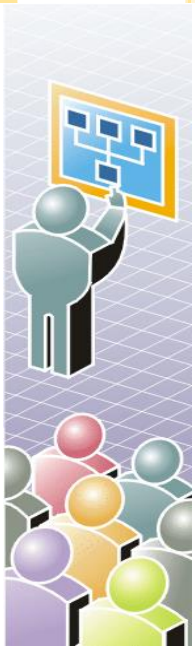
ExtraClases

Objetivo:

El alumno realizar los siguientes puntos por cada problema:

1. Análisis del problema
2. PSeudocódigo
3. Plan de prueba y verificación
4. Codificación en el lenguaje de programación

Actividades: Desarrollo de algoritmos para:



1. Elabore un algoritmo que calcule la siguiente serie: y = 1+ 1/i , donde i toma valores desde 20 hasta -5. Ejemplo:

```
y=
   1+1/20 +
   1+1/19  +
   1+1/18  +
    . . .
   1+1/1
```

2. Elaborar un algoritmo que permita evaluar la función $5X^2+3X+8=538$, cuando se conoce que la solución esta entre el rango de -3 … 20.

Estructuras C# • FIUAS

Actividad

1. Un banco establece que la clave secreta para acceso a sus cajeros automáticos debe ser un numero de cuatro dígitos, tales que ninguno de ellos se repita y que la suma de los dos dígitos intermedios sea par. Elabore un algoritmo, que permita ingresar un valor entero y si se trata de una clave valida imprimir el segundo digito menor.

La instrucción foreach

Es habitual hacer iteraciones para todos los elementos de una lista o colección.

Las colecciones son entidades software cuyo propósito es reunir otras entidades software, del mismo modo que un libro de cuentas se puede considerar una colección de cuentas bancarias o que una casa puede ser una colección de habitaciones.

Sintaxis:

```
foreach (<tipoElemento> <elemento> in <colección>)
{
    <instrucciones>
}
```

Estructuras C# • FIUAS

Actividad

Para el Alumno

Descripción:

El alumno realizar un problema que solicite al usuario su nombre y determine e imprima cuantas vocales tiene.

1. Codificarlo en el lenguaje de programación C#

La instrucción foreach

- Elige el tipo y el nombre de la variable de iteración
- Ejecuta instrucciones incrustadas para cada elemento de la clase collection

```
ArrayList Numeros = new ArrayList( );
for (int Indice=0; Indice < 10; Indice++ ) {
    Numeros.Add(Indice);
}

foreach (int Numero in Numeros)
{
    Console.WriteLine(Numero);
}
```

```
0 1 2 3 4 5 6 7 8 9
```


Uso de instrucciones de salto

se usan para transferir el control incondicionalmente a otra instrucción.

- La instrucción goto
- Las instrucciones break y continue

La instrucción goto

- Transfiere el flujo de control a una instrucción con etiqueta
- Pueden dar lugar fácilmente a código "spaghetti" de difícil interpretación

```
if (Numero % 2 == 0)
{
  goto Par;
}
Console.WriteLine("impar");
goto Fin;
Par:
Console.WriteLine("par");
Fin:
---
```


Las instrucciones break and continue

- La instrucción break salta fuera de una iteración
- La instrucción continue salta a la siguiente iteración

```
int Indice = 0;
while (true) {
    Console.WriteLine(Indice);
    Indice++;
    if (Indice < 10)
    {
        continue;
    }
    Console.Write("FIN");
    break;
}
```

Preguntas?