UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Estructura de un Programa en C#

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Septiembre de 2017

Competencias

Al final de este módulo, los estudiantes habrán alcanzado las siguientes competencias:

- Describir los tipos de variables que se pueden emplear en programas C#.
- Nombrar variables según las convenciones de nomenclatura estándar de C#.
- Declarar variables utilizando tipos de datos predefinidos.
- Asignar valores a variables.
- Convertir variables existentes de un tipo de dato a otro.
- Crear y utilizar sus propios tipos de datos.

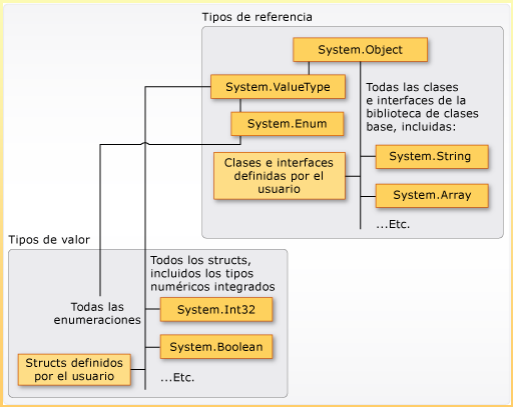
Sistema de tipos comunes (CTS)

- Cada variable tiene un tipo de datos que determina los valores que se pueden almacenar en ella. C# es un lenguaje de especificaciones seguras (type-safe), lo que significa que el compilador de C# garantiza que los valores almacenados en variables son siempre del tipo adecuado.
- El runtime de lenguaje común incluye un sistema de tipos comunes (Common Type System, CTS) que define un conjunto de tipos de datos predefinidos que se pueden utilizar para definir variables.

Tipos de datos C#• FIUAS

Aspectos generales del CTS

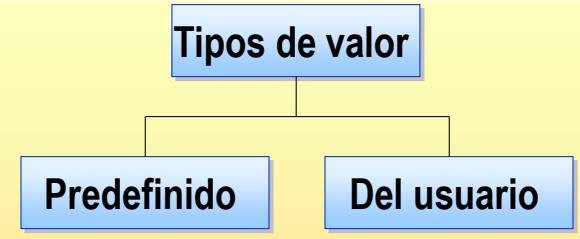
- El CTS admite tanto tipos de valor como de referencia



Comparación de tipos de valor y de referencia

- **Tipos de valor:**
  - Contienen sus datos directamente
  - Cada una tiene su propia copia de datos
  - Las operaciones sobre una no afectan a otra
- **Tipos de referencia:**
  - Almacenan referencias a sus datos (conocidos como objetos)
  - Dos variables de referencia pueden apuntar al mismo objeto
  - Las operaciones sobre una pueden afectar a otra

Comparación de tipos de valor predefinidos y definidos por el usuario



- **Ejemplos de tipos de valor predefinidos:**
  - int
  - float
- **Ejemplos de tipos de valor definidos por el usuario:**
  - enum
  - struct

Contienen valores a NULL

Tipos simples (predefinidos)

Los tipos de valor predefinidos también reciben el nombre de tipos de datos básicos o tipos simples.

Los tipos simples se identifican por medio de palabras reservadas que son alias para tipos de estructura predefinidos.

- Se identifican mediante palabras reservadas:
  - int // Palabra reservada
    - o -
  - **System.Int32**

Tipos de datos C#• FIUAS

Enteros

La siguiente tabla muestra los tamaños e intervalos de los tipos integrales, que constituyen un subconjunto de los tipos simples.

Tipo	Intervalo	Size
sbyte	-128 a 127	Entero de 8 bits con signo
byte	0 a 255	Entero de 8 bits sin signo
char	U+0000 a U+ffff	Carácter Unicode de 16 bits
short	-32.768 a 32.767	Entero de 16 bits con signo
ushort	0 a 65.535	Entero de 16 bits sin signo
int	-2.147.483.648 a 2.147.483.647	Entero de 32 bits con signo
uint	0 a 4.294.967.295	Entero de 32 bits sin signo
long	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	Entero de 64 bits con signo
ulong	0 a 18.446.744.073.709.551.615	Entero de 64 bits sin signo

Comentarios

Si el valor representado por un literal entero supera el intervalo de valores del tipo **ulong**, se producirá un error de compilación.

Fraccionarios (punto flotante)

Tipo	Intervalo aproximado	Precisión
float	$\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$	7 dígitos
double	$\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$	15-16 dígitos

- **Comentarios**

Si el valor representado por un literal supera el intervalo de valores del tipo, se producirá un error de compilación.

Decimal

La palabra clave **decimal** denota un tipo de datos de 128 bits. Comparado con los tipos de punto flotante, el tipo **decimal** tiene una mayor precisión y un intervalo de valores más reducido, lo que le hace adecuado para cálculos financieros y monetarios.

El intervalo de valores aproximado y la precisión para el tipo **decimal** se muestran en la siguiente tabla.

Tipo	Intervalo aproximado	Precisión	Tipo de .NET Framework
decimal	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$	28-29 dígitos significativos	**System.Decimal**

bool

La palabra clave bool es un alias de System.Boolean. Se utiliza para declarar variables que almacenan los valores booleanos true y false.

Literales:

Es posible asignar un valor booleano a una variable de tipo bool, por ejemplo:

- bool MyVar = true;

También se puede asignar una expresión que se evalúa como bool a una variable de tipo bool, por ejemplo:

- bool Alphabetic = (c > 64 && c < 123);

Tipos de datos C#• FIUAS

Tipos de Referencia

Las variables de tipos de referencia, conocidas como objetos, almacenan referencias a los datos reales. Esta sección presenta las palabras clave siguientes utilizadas para declarar tipos de referencia:

- class
- interface
- delegate

Esta sección también presenta los siguientes tipos de referencia integrados:

- object
- string

Tipos de datos básicos definidos en la BCL y su alias en C#

Tipo	Descripción	Bits	Rango de valores	Alias
SByte	*Bytes* con signo	8	-128 – 127	sbyte
Byte	*Bytes* sin signo	8	0 – 255	byte
Int16	Enteros cortos con signo	16	[-32.768, 32.767]	short
UInt16	Enteros cortos sin signo	16	[0, 65.535]	ushort
Int32	Enteros normales	32	[-2.147.483.648, 2.147.483.647]	int
UInt32	Enteros normales sin signo	32	[0, 4.294.967.295]	uint
Int64	Enteros largos	64	[-9.223.372.036.854.775.808, 9.223.372.036.854.775.807]	long
UInt64	Enteros largos sin signo	64	[0-18.446.744.073.709.551.615]	ulong
Single	Reales con 7 dígitos de precisión	32	$[1,5\times10^{-45}$ - $3,4\times10^{38}]$	float
Double	Reales de 15-16 dígitos de precisión	64	$[5,0\times10^{-324}$ - $1,7\times10^{308}]$	double
Decimal	Reales de 28-29 dígitos de precisión	128	$[1,0\times10^{-28}$ - $7,9\times10^{28}]$	decimal
Boolean	Valores lógicos	32	**true**, **false**	bool
Char	Caracteres Unicode	16	['\u0000', '\uFFFF']	char
String	Cadenas de caracteres	Variable	El permitido por la memoria	string
Object	Cualquier objeto	Variable	Cualquier objeto	object

System.int32 i=10; int i=10;

Sufijos que admiten los literales numéricos para indicar al compilador cuál es el tipo que se ha de considerar que tiene:

Sufijo	Tipo del literal entero
ninguno	Primero de: **int, uint, long, ulong**
L ó **l**	Primero de: **long, ulong**
U ó **u**	Primero de: **int, uint**
UL, Ul, uL, ul, LU, Lu, lU ó **lu**	**ulong**

Sufijos de literales enteros

Sufijo	Tipo del literal real
F ó **f**	**float**
ninguno, **D** ó **d**	**double**
M ó **m**	**decimal**

Sufijos de literales reales

Inicialización

El valor que por defecto se da a los campos de tipos básicos consiste en poner a cero todo el área de memoria que ocupen.

Esto se traduce en que los campos de tipos básicos:

- numéricos se inicializan por defecto con el valor 0,
- los de tipo **bool** lo hacen con **false**,
- los de tipo **char** con '\u0000', y
- los de tipo **string** y **object** con **null**.

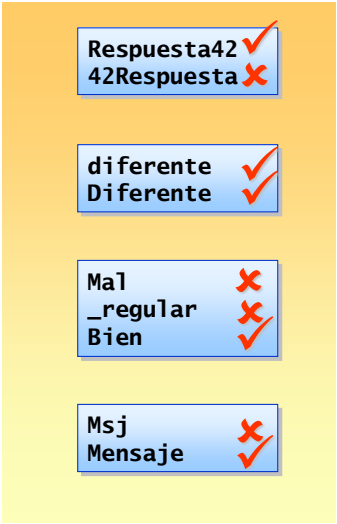
Nombres de variables

Reglas y recomendaciones para nombrar variables.

Para usar una variable hay que elegir primero un nombre apropiado y significativo para ella. Cada variable tiene un nombre al que se conoce también por *identificador de variable*.

Reglas y recomendaciones para nombrar variables

- **Reglas**
  - Use letras, el signo de subrayado y dígitos
- **Recomendaciones**
  - Evite poner todas las letras en mayúsculas
  - Evite empezar con un signo de subrayado
  - Evite el uso de abreviaturas
  - Use PascalCasing para nombres con varias palabras



Palabras clave de C#

- Las palabras clave son identificadores reservados

```
abstract, base, bool, default, if, finally
```

- No utilice palabras clave como nombres de variables
  - Produce errores en tiempo de compilación
- Procure no usar palabras clave cambiando mayúsculas y minúsculas

```
int INT;  // Mal estilo
```

Uso de tipos de datos predefinidos

- Declaración de constantes y variables locales
- Asignación de valores a variables
- Asignación compuesta
- Operadores comunes
- Incremento y decremento
- Precedencia de operadores

Constantes

Las constantes son variables cuyo valor no puede ser alterado:

```
const int SegundosPorHora = 3600;
const float PI = 3.1416F;
```

Declaración de variables locales

- Se suelen declarar por tipo de dato y nombre de variable:

```
int objetoCuenta;
```

- Es posible declarar múltiples variables en una declaración:

```
int objetoCuenta, empleadoNúmero;
```

```
int objetoCuenta,
    empleadoNúmero;
```


Asignación de valores a variables

- Asignar valores a variables ya declaradas:

```
int empleadoNumero;
empleadoNumero = 23;
```

- Inicializar una variable cuando se declara:

```
int empleadoNumero = 23;
```

- También es posible inicializar valores de caracteres:

```
char inicialNombre = 'J';
```

Asignación compuesta

- Es muy habitual sumar un valor a una variable

```
itemCount = itemCount + 40;
```

- Se puede usar una expresión más práctica

```
itemCount += 40;
```

- Esta abreviatura es válida para todos los operadores aritméticos:

```
itemCount -= 24;
```


Operadores comunes

Operadores comunes	Ejemplo
• Operadores de igualdad	== !=
• Operadores relacionales	< > <= >= is
• Operadores condicionales	&& || ?: !
• Operador de incremento	++
• Operador de decremento	- -
• Operadores aritméticos	+ - * / %
• Operadores de asignación	= *= /= %= += -=

Operador condicional

Es el único operador incluido en C# que toma 3 operandos, y se usa así:

<condición> **?** <expresión1> **:** <expresión2>

El significado del operando es el siguiente: se evalúa <condición> Si es cierta se devuelve el resultado de evaluar <expresión1>, y si es falsa se devuelve el resultado de evaluar <condición2>. Un ejemplo de su uso es:

```
b = (a>0)? a : 0;      // Suponemos a y b de tipos enteros
```


Incremento y decremento

- Es muy habitual cambiar un valor en una unidad

```
objetoCuenta += 1;
objetoCuenta -= 1;
```

- Se puede usar una expresión más práctica

```
objetoCuenta++;
objetoCuenta--;
```

- Existen dos formas de esta abreviatura

```
++objetoCuenta;
--objetoCuenta;
```

Precedencia de operadores

- Precedencia y asociatividad de operadores
  - Todos los operadores binarios, salvo los de asignación, son asociativos por la izquierda
  - Los operadores de asignación y el operador condicional son asociativos por la derecha

Tipos de datos C# • FIUAS

Creación de tipos de datos definidos por el usuario

- Enumeraciones
  - Una enumeración crea un conjunto único de valores constantes relacionados.
  - Los enumeradores son útiles cuando una variable sólo puede tener un conjunto de valores concreto.
- Estructuras
  - Una estructura es una agrupación de tipos arbitrarios.

Enumeraciones

- Definición de una enumeración

```
enum Color { Rojo, Verde, Azul }
```

```
Color colorPaleta = Color.Rojo; - O - colorPaleta = (Color)0;
```

```
Console.WriteLine("{0}", colorPaleta); // Muestra Rojo
```

tiene el valor 1 y *Azul* tiene el valor 2.

Estructuras

- Definición de una estructura

```
public struct Empleado
{
    public string pilaNombre;
    public int age;
}
```

- Uso de una estructura

```
Employee empresaEmpleado;
empresaEmpleado.pilaNombre = "Juan";
empresaEmpleado.age = 23;
```

Conversión de tipos de datos

- Conversión implícita de tipos de datos
- Conversión explícita de tipos de datos

•Conversión de Tipos

- Explícita

  Se llevara a cabo solamente si se especifica la conversión. Hay que aclarar que no siempre es posible convertir un tipo a otro.

```
int entero = 456;
short corto = (short) entero;
```

- Implícita

  Ocurren cuando se desea cambiar de un tipo de menor capacidad a uno de mayor capacidad del mismo tipo. Por ejemplo de un short a un int.

```
short corto = 3;
int entero = corto;
```

Conversión de Tipos

Conversión Implícita (I) Conversión Explícita (E)											
Desde/hacia	byte	sbyte	short	ushort	int	uint	long	ulong	float	double	decimal
byte	I	E	I	I	I	I	I	I	I	I	I
sbyte	E	I	I	E	I	E	I	E	I	I	I
short	E	E	I	E	I	E	I	E	I	I	I
ushort	E	E	E	I	I	I	I	I	I	I	I
int	E	E	E	E	I	E	I	E	I	I	I
uint	E	E	E	E	E	I	I	I	I	I	I
long	E	E	E	E	E	E	I	E	I	I	I
ulong	E	E	E	E	E	E	E	I	I	I	I
float	E	E	E	E	E	E	E	E	I	I	E
double	E	E	E	E	E	E	E	E	E	I	E
decimal	E	E	E	E	E	E	E	E	E	E	I

Conversión implícita de tipos de datos

- Conversión de int a long

```
using System;
class Test
{
   static void Main( )
   {
        int intValor = 123;
        long longValor = intValor;
        Console.WriteLine("(long) {0} = {1}", intValor, longValor);
   }
}
```

- Las conversiones implícitas no pueden fallar
  - Se puede perder precisión, pero no magnitud

Conversión explícita de tipos de datos

- Para hacer conversiones explícitas se usa una expresión de cast (molde):

```
using System;
class Test
{
   static void Main( )
   {
        long longValor = Int64.MaxValor;
        int intValor = (int) longValor;
        Console.WriteLine("(int) {0} = {1}", longValor,intValor);
   }
}
```

Tipos de datos C# • FIUAS

Conversión explícita de tipos de datos

- Para hacer conversiones explícitas se usa una expresión de cast (molde):

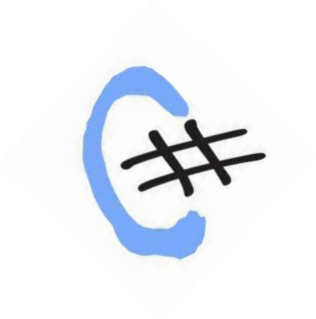
```
using System;
class Test
{
   static void Main( )
   {
        long longValor = Int64.MaxValor;
        int intValor = int.Parse(longValor);
        Console.WriteLine("(int) {0} = {1}", longValor,intValor);
   }
}
```

Conversión explícita de tipos de datos

- Para hacer conversiones explícitas se usa una expresión de cast (molde):

```
using System;
class Test
{
   static void Main( )
   {
        long longValor = Int64.MaxValor;
        int intValor = Convert.ToInt32(longValor);
        Console.WriteLine("(int) {0} = {1}", longValor,intValor);
   }
}
```

Tipos de datos C#• FIUAS



Codificación de Algoritmos Pseudocódigos Secuenciales al Lenguaje.

Actividad 1

1. Codificar el pseudocódigo propuesto.
2. Verificar su correcto funcionamiento, utilizando el plan de prueba elaborado.

Tipos de datos C# • FIUAS

Pseudocódigo

Objetivo: Calcular la Superficie de una Circunferencia
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/septiembre/2016

```
INICIO
        CONST REAL  PI = 3.1416
        ENTERO  Radio, Superficie
       IMPRIMIR "Indique el valor del Radio:"
       LEER  Radio
        Superficie = PI*Radio*Radio
        IMPRIMIR "La superficie de la circunferencia de  radio ",
   Radio, "es igual a ", Superficie
FIN
```

Plan de Prueba o verificación del algoritmo

Verificación

¿Desarrollamos el Algoritmo correctamente?

Validación

¿Desarrollamos el Algoritmo correcto?

Valores de Entrada	Salidas Esperadas	Resultado
Radio =**10**	Superficie= **314.16**	Correcto/Incorrecto

Tipos de datos C# • FIUAS

Actividad 2

Pseudocódigo

Objetivo: Determinar el total que pagara Ana por la compra de Huesos.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/septiembre/2016

```
INICIO
    //Definición de Constantes y Variables
  CONST ENTERO PorcentajeDelgada=15, PorcentajeHueso=17
  REAL TotalPagarHueso, PrecioKiloGruesa, KilosHueso, PrecioKiloDelgada,
    PrecioKiloHueso
  //Lectura de Datos no Conocidos
  IMPRIMIR "Teclee el precio por kilo de carne gruesa:$"
  LEER PrecioKiloGruesa
  IMPRIMIR "Teclee el número de kilos de hueso a comprar:"
  LEER KilosHueso
```

Tipos de datos C# • FIUAS

Pseudocódigo

```
  //Procesamiento de los Datos
  //Determinar el precio por kilo de la carne delgada
  PrecioKiloDelgada = PrecioKiloGruesa * ( ( 100 –
                      PorcentajeDelgada) / 100)
  //Determinar el precio por kilo de Hueso
  PrecioKiloHueso = PrecioKiloDelgada * (PorcentajeHueso / 100)
  //Determinar el total que pagara Ana por la compra de huesos
  TotalPagarHueso = KilosHueso * PrecioKiloHueso
  //Impresión de Resultados
  IMPRIMIR "Total a pagar:$",TotalPagarHueso
FIN
```

Carácter Coma (,)

Actividad 3

Pseudocódigo

```
//Objetivo: Determinar el Total de Producción de Mangos Finos
//Programador: MC. Gerardo Gálvez G.
//Fecha:__ /Septiembre/2015

INICIO
    //Definición de Constantes y Variables
    CONST ENTERO  ProduccionPlantaFina=8500
    ENTERO NumeroArboles,TotalProduccion
    //Lecturas de Datos de no Conocidos (Entrada)
    IMPRIMIR "Teclee el Número de Árboles en la Huerta:"
    LEER NumeroArboles
    //Procesamiento de los Datos
    // (calcular el total de producción de mangos finos
    TotalProduccion=(NumeroArboles - NumeroArboles MOD 2) / 2  * ProduccionPlantaFina
   //Impresión de Resultados
   IMPRIMIR "Total de Mangos Finos Producidos: ", TotalProduccion
FIN
```

Preguntas?

Tipos de datos C#• FIUAS

Actividades

ExtraClases

Objetivo

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