Carrera: Desarrollo de software Semestre 5

Unidad didáctica:
Programación net I

Unidad 1. Introducción a .net

Ciudad de México, octubre de 2025

Clave:
15143526

Universidad Abierta y a Distancia de México

Índice

Presentación de la unidad

Bienvenido(a) a la unidad didáctica de Programación .NET I. En esta primera unidad conocerás latecnología .NET, específicamente trabajarás en el compilador C# .NET, conocerás su entornode desarrollo y configuración, los datos que soporta, los operadores aritméticos y lógicos utilizados, además de los tipos de datos a utilizar. De igual forma, se mostrará cómo se estructura un programa en este lenguaje de programación, qué palabras reservadas se utilizan y cómo se declaran objetos y constructores bajo este lenguaje.

Para revisar una breve semblanza del surgimiento de .NET y el alcance que tiene para el desarrollo de software, revisa los documentos U1. Importancia de .NET (Hernández, 2007) y U1. Componentes .NET en la sección *Materiales de apoyo.*

Logros de la unidad

- Distinguir las características de .NET.
- Identificar la organización de un programa.
- Crear programas básicos en .NET.

Competencia específica

- Analiza las características que ofrece la plataforma .NET para resolver problemas informáticos y representar su solución mediante herramientas de software.

1.1 Características de LA PLATAFORMA .NET

Las diversas herramientas de desarrollo de software presentan diversos problemas por ejemplo algunas inconsistencias del sistema operativo Windows, interoperabilidad en el lenguaje, etcétera, “el principal objetivo de Microsoft con Visual Studio. NET fue crear una arquitectura que eliminara los principales defectos de las herramientas de desarrollo de software actuales […] .NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones” (Hernández, 2007, p.47). Microsoft Visual .NET es considerado como un Entorno de Desarrollo Integrado (IDE), que contiene un conjunto variado de herramientas necesarias para desarrollar diversos tipos de aplicaciones. Trabaja con compiladores o lenguaje sencillos de aprender, uno de ellos es elC#, el cuál toma características de otros lenguajes, tales como C, C++ y Java. (Hernández,2007).

1.1.1. Entorno de desarrollo y configuración

Es importante que instales la aplicación Visual Studio .NET debido a que, en ella podrás realizar ejercicios para complementar tu conocimiento; de igual forma es importante que estudies los temas de los libros que se te citan, los leas y comprendas, ya que éstos son los que te orientarán para comenzar a realizar aplicaciones sencillas que te ayudarán a un mejor aprendizaje.

Una de las partes esenciales es realizar las actividades presentadas en esta materia, por lo que es importante que instales la aplicación Visual Studio .NET más actual en versión express directamente de Microsoft en línea.

A mitad de la pantalla te mostrará varias opciones, de la cual deberás elegir: “Ediciones Express” y, posteriormente; la opción “Express para escritorio” dando click en el botón “Descarga” con lo que inmediatamente iniciará la descarga de un archivo ejecutable que te guiará a través de la descarga e instalación del programa.

Durante el proceso de instalación se verificará que se cuente con los requisitos mínimos y programas necesarios, tales como: Windows 7 SP1, procesador de 1.6 Ghz, 1 Gb de Ram, etc., para la correcta ejecución de Visual Studio por lo que puede tomar un tiempo considerable completar el proceso. La lista completa de requerimientos la puedes verificar en el sitio de Visual Studio de Microsoft (2022)

Si tienes dudas en cuanto a la instalación, te puedes apoyar en cualquiera de los manuales de instalación que se muestran en la web ***y con tu figura académica***.

> *Microsoft es uno de los proveedores de aplicaciones reconocidos a nivel mundial, si bien es cierto que la mayoría de ellos no son de libre uso a menos que se cuente con una licencia, es importante conocer que Microsoft cuenta con una serie de productos denominados 'express'. Visual Studio ofrece un conjunto de herramientas gratuitas paraponer sus ideas en acción. Con estas aplicaciones es posible crear grandes aplicaciones para el teléfono, web, móviles y de escritorio, todo el tiempo creando una familiarización con el IDE de Visual Studio. Por tal motivo es importante que ingreses al link para descargar la aplicación que utilizarás en este curso.*
> *https://visualstudio.microsoft.com/es/vs/express/*

Visual Studio .NET es considerado como un entorno de programación dotado de herramientas funcionales necesarias para crear proyectos en C#, de la magnitud que se desee. Es importante considerar que incluso el trabajar en este lenguaje permite al desarrolladorcombinar o mezclar módulos de diferentes lenguajes.

Para revisar las principales características de Visual Studio revisa el documento U1. Visual Studio .NET en la sección *Material de apoyo*.

1.1.2. Tipos de datos soportados en .NET

Es esta sección, revisarás los tipos de datos soportados en C# de VS .NET, además analizarás datos importantes en cuanto a la sintaxis utilizada y cómo se aplican en el entorno de programación.

Antes de ingresar de lleno a los tipos de datos soportados en .NET, es importante conocer sobre ciertas reglas sintácticas que maneja .NET y sobre los identificadores del lenguaje .NET.

Reglas sintácticas del lenguaje C#:

- Cuando se termina una sentencia, ésta debe incluir un punto y coma al final.
- **Distingue** entre **mayúsculas** y **minúsculas**, por lo que, si se escribe una variable denominada *hola* y luego se le referencia como *Hola*, el lenguaje las considera como variables diferentes.
- Las variables son como baldes contenedores de cierta información o datos, con un cierto tipo definido; para poder hacer uso de una variable es necesario declararla; una vez declarada, se le podrá asignar la información correspondiente, además de que puede variar según los procedimientos que vaya diseñando el programador.

Con base en los autores Sharp John & Jagger Jon (2002, pp. 30), se explica información sobre el tema de las variables y la forma en que se declaran, **pon** atención en este tema, pues, para obtener resultados certeros en un programa, es necesario contar con variables adecuadas que almacenen dicho resultado.

Consulta también, Ceballos F., (2007, pp. 31 - 43) para un conocimiento más amplio; en las páginas indicadas, se desarrolla el tema de identificadores y variables.

Puedes consultar también la librería MSDN de Microsoft (2007b), en la cual podrás localizar información sobre las constantes y variables:

Los identificadores en el lenguaje C#

Al igual que otros lenguajes, C# tiene ciertas consideraciones en cuanto a sus identificadores, los cuales sirven para designar los elementos que conforman un programa, tales como una clase, una variable, un método etc. En Ceballos F., (2007, pp. 41-43) **lee** las páginas

indicadas, ahí se mencionan los identificadores, declaración y características de cada uno de ellos.

Una vez que se conoce cómo deben utilizarse los identificadores en el lenguaje, entraremos de lleno al tema de tipos de datos; sobre los cuales es importante conocer los tipos de datos primitivos, ya que son utilizados en la plataforma .NET. Un aspecto importante para mencionar es que, en cada uno de los compiladores de la plataforma, los tipos de datos soportan diferente tamaño o peso, por lo que nos enfocaremos ahora sólo en los tipos de datos primitivos .NET para C#.

Los tipos de datos, de forma general, constituyen el tipo de información que almacenará cualquier variable; éstos indican que tipo de instrucciones podrán ejecutarse sobre la variable; por mencionar algunos:

- Datos de tipo entero.
- Datos de tipo cadena.
- Datos de tipo carácter.
- Datos de tipo punto flotante, entre otros.

Cabe mencionar que, según el tipo de dato antepuesto a la variable, será el resultado que ésta arroje al hacer una operación.

Los tipos de datos en C# se clasifican en:

- Tipos valor
- Tipos referencia.

C# posee cierto número de datos incorporados denominados tipos de datos primitivos. En Ceballos F. (2007, pp.38-39) se hace referencia a los tipos de datos y su estructura en C#, por lo que es necesario que consultes este documento en la sección Material de apoyo.

Revisa a Ramírez F. (2007, pp. 62-63). Este autor hace una referencia a los tipos de datos utilizados en cada una de las plataformas soportadas por .NET.

Se te siguiere no sólo leas los términos que se te presentan: tipos de datos, identificadores, variables, operadores y conversión de tipos de datos, si no que **revises** los temas de los libros referenciados, ya que te ayudarán abordar a fondo cada uno de los temas.

1.1.2. Operadores aritméticos, lógicos y de comparación

C# maneja, al igual que todos los demás lenguajes de programación, operadores, los cuales son útiles para realizar las distintas operaciones que te posibilitan resolver algún problema; no sólo se trata de operadores aritméticos útiles para realizar operaciones matemáticas, sino de operadores relacionales y lógicos, útiles para las distintas tomas de decisiones.

Los operadores aritméticos son lo que se conocen y utilizan cotidianamente, es decir el operador suma +, resta -, multiplicación * y división /. Es importante conocer que estos operadores se aplican sólo a los tipos de datos *int, float, long, short y double*, pero no se pueden aplicar a tipos de datos tales como *string o bool* ya que no son numéricos.

Revisa a Ceballos F. (2007, pp. 43 – 47), pues presenta una detallada referencia a los operadores y a sus tipos utilizados en .NET, lo cual te ayudará para conocer la correcta evaluación de valores al realizar tus programas.

Ramírez F. (2007, pp. 76 - 83) muestra de forma detallada los operadores y sus tipos, finalizando con un ejercicio. **Revisa** el texto y, posterior a ello, **realiza** el ejercicio para una mejor comprensión del tema.

1.1.3. Conversión de tipos de datos

Otro aspecto importante para mencionar es que podemos hacer conversiones entre tipos de datos; esto, porque puede haber ocasiones en que una variable la utilicemos para realizar una operación en la que necesitamos que el resultado sea mostrado como entero; pero, quizá en ése mismo programa, pero en diferente lugar, necesitamos que esa variable sea mostrada como flotante, para lo que necesitamos hacer en ese momento una conversión de ese tipo de datos.

En Ceballos F (2007, pp. 39 – 40) se hace referencia a cómo convertir un cierto tipo de dato a otro, según la necesidad del usuario, **lee** detenidamente el texto para una mejor comprensión. Posterior a ello, **analiza** los ejercicios que se presentan en los libros citados, referentes a los temas, además de diversos ejercicios que puedes encontrar en la liga directa de la librería MSDN (2015a).

Para concluir, es importante comentar que no sólo te debes de quedar con lo que aquí se te muestra, si no **adentrarte** e **investigar** más a fondo sobre cada uno de los temas que se mencionan; realizando y analizando cada uno de los ejemplos mostrados.

En el estudio correspondiente a las características del lenguaje .NET, es importante no sólo tomar en cuenta aspectos, tales como entorno de desarrollo, configuración de la aplicación y uso de esta, si no otros aspectos tales como sintaxis utilizada, los tipos de datos, identificadores y variables, que son necesarios para representar declaraciones correctas en la programación C#.

1.2. Organización de un programa

En este apartado conocerás cómo se organiza un programa en C #.NET, comenzarás con la estructuración del programa; ya que, si bien se han visto las características de la programación, no se ha visto cómo se implementa en un programa estructurado.

Otra de las secciones de suma importancia para el desarrollo de aplicaciones en este compilador, son las palabras reservadas que soporta y utiliza; cabe mencionar que son aproximadamente sesenta palabras reservadas, por lo que se enumerarán y ejemplificarán algunas de las más utilizadas; para conocer el resto es necesario tomar en cuenta lo que se propone en la sección de *Para saber más.*

Finalmente se mostrará como declarar objetos y constructores, esto debido a que C#, al igual que Java, es un lenguaje orientado a objetos, por lo que, si se comprendió la base ya vista en Java, se entenderá sin complicaciones la base que utiliza C# para estas declaraciones.

1.2.1. Estructura general de un programa EN C#

Iniciemos con una de las partes primordiales de esta primera unidad, si bien es de suma importancia conocer los tipos de datos, identificadores, operadores entre algunos otros temas sobre C# .NET, ninguno de ellos se comprendería si no se conoce cómo es la **estructura básica de este compilador**.

Para iniciar una nueva aplicación en C# te invito a que visites la siguiente liga de Microsoft Developer Network (MSDN), es una página de Microsoft enfocada totalmente a los diferentes programas desarrollados por esta compañía, la cual se actualiza continuamente; en ella, se te explica paso a paso como iniciar una nueva aplicación en C# .Net, **verifica** la *aplicación consola*.

Este tipo de proyecto se utiliza para crear aplicaciones y utilidades de línea de comandos. El resultado y la entrada del programa tienen lugar a través de una ventana de terminal basada en texto, esto te ayudará a entender mejor la programación básica de C#, que es la que se pretende lograr en esta Unidad 1 *aplicación consola*, disponible en (MSDN,2007a)

La siguiente imagen muestra un ejemplo del esqueleto en C# .NET en consola, la cual se presenta al abrir un nuevo proyecto en la aplicación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace EstructuraBasica
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Imagen 1. Esqueleto C# .NET

Observa detenidamente: dentro del *namespace,* está definida una clase, la cual contiene un método estático llamado *Main*, este método es esencial, ya que es el punto de entrada al programa, sin él, el programa no ejecuta o arranca. En el *Main* se pueden ingresar mensajes en pantalla, declaración de variables, realizar operaciones directas o bien, mandar llamar a ciertos métodos definidos en una clase.

En la imagen *Código en consola de programa Bienvenido* se muestra un ejemplo de aplicación de tipo consola, se muestra un mensaje de bienvenido a tu curso C# .NET, posteriormente pide ciertos datos de tipo *int y string* y los muestra concatenados.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HolaBienvenido
{
    class Hola
    {
        static void Main(string[] args)
        {
            int Edad;
            string nombre;
            Console.WriteLine("Bienvenido a tu curso de .NET I");

            Console.WriteLine("Captura tu edad: ");
            Edad = int.Parse(Console.ReadLine());

            Console.WriteLine("Captura tu nombre: ");
            nombre = (Console.ReadLine());

            Console.WriteLine("Tu nombre es : " + nombre + "  y tu edad es : " + Edad);
        }
    }
}
```

Imagen 2. Código en consola de programa Bienvenido

1.2.2. Palabras reservadas

En este apartado se conocerán cuáles son las palabras reservadas del lenguaje C#, de las que es importante conocer: su finalidad y que no pueden ser utilizadas para asignación de valores como lo hacemos comúnmente con las variables, ya que a éstas no se les puede asignar un tipo de datos.

Las palabras reservadas o clave de C#, no son más que identificadores predefinidos reservados que tienen un significado especial para el compilador. No se pueden utilizar como identificadores en un programa a menos que incluyan el carácter @ como prefijo.

Las palabras reservadas constituyen un elemento fundamental en el desarrollo de un programa en C# y son necesarias para codificar las diversas instrucciones de tu código tales como: la declaración de variables, definir tipos de métodos, tipos de datos, creación de clases, creación de objetos, etc., por ello, es necesario que visites el sitio de MSDN (2015b), para revisar cuáles son estas palabras y desarrolles algunos ejemplos que se recomiendan.

A continuación, se muestra un listado de las palabras reservadas disponibles en C# y que puedes y en su caso debes utilizar para la codificación de tu programa:"

abstract	as	base	bool
break	byte	case	catch
char	checked	clase	const
continue	decimal	default	delegado
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	in (modificador genérico)	Valor int.	interfaz
internal	es	bloquear	long
namespace	new	null	Objeto.
':?'.	out	out (modificador genérico)	override
params	private	protected	public
readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc
static	string	struct	switch
this	throw	true	try
typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual
void	volatile	while	

Tabla 1. Palabras reservadas
https://msdn.microsoft.com/es-es/library/x53a06bb.aspx

1.2.3. Declaración de objetos y constructores

Inicialmente al escuchar la palabra objeto no se nos viene nada a la mente, necesitamos conocer por lo menos alguna característica que pueda identificar a lo que nos referimos (si decimos un objeto que tiene llantas, volante, puertas, máquina entre otros, muy probablemente nos refiramos a un auto o camioneta o tráiler, de hecho, éstos tienen diferentescaracterísticas entre sí). Otro ejemplo de características de objeto es: que es redonda, rebota,de color blanco, pequeño y se utiliza para jugar golf, quizá se trata de una pelota de golf, la cual para poder ser identificada fue necesario conocer ciertas características de este objeto.

- Por tal motivo, al referirnos a un objeto en programación, necesitamos saber cuáles serán suscaracterísticas y lo más importante, su utilidad.

*Ceballos, F. (2007 pp. 82) apunta que "*Un objeto es una determinada clase que se crea al momento en que se invoca al operador **new** para dicha clase".

Por lo tanto: **lee** todo el tema referente a objetos, clases y atributos, además de que **analiza** y **realiza** los ejemplos que ahí se muestran para una mejor comprensión, en *Ceballos, F. (2007* pp. 82 - 85).

- Los **constructores** son métodos de una clase, éstos se llegan a ejecutar cuando se crea unobjeto de cualquier tipo. Es de suma importancia conocer que los constructores deben de tener el mismo nombre que la clase y determinar los datos que se van a inicializar.

También lo podemos definir como: mensajes que recibe la clase para crear e inicializa un objeto para que tenga valores correctos.

El constructor es invocado desde la clase *Main*, de ahí se pueden enviar parámetros a la clase y con ellos generar o procesar la información, la cual es impresa nuevamente desde la clase *Main*. Es decir, va y regresa a la clase *Main*. Ceballos, F. (2007. pp. 94) define a "*Un constructor es un método especial de una clase que se llama automáticamente siempre que se crea un objeto de esa clase. Su función es iniciar el objeto".*

Lee detenidamente el tema completo de constructores y destructores, el cuál detalla de forma clara Ceballos, F. (2007, pp. 94-97).

En la siguiente imagen (*Ejemplo de constructor de clase sin parámetros*) se muestra un ejemplo en el que se tiene una clase llamada *Autos* definida con un constructor simple, el cuál es invocado con el operador new:

```
public class Autos
{
    //Constructor sin parámetros
    public Autos() //Debe llevar el mismo nombre que la clase
    {
        System.Console.WriteLine("El constructor se invoca al crear el objeto");
    }

}

class Program
{
    static void Main(string[] args)
    {
        //Se crea el objeto y se invoca
        //al constructor de la clase
        Autos objAutos = new Autos();
    }
}
```

Imagen 3. Ejemplo de constructor de clase sin parámetros.

Por otro lado, en la siguiente imagen (Ejemplo de constructor de clase con parámetros) se muestra un ejemplo de un constructor de clase que recibe un parámetro de tipo entero **num** el cual debe especificarse al momento de crear el objeto:

```
public class Multiplica //Declaración de la clase
 {
     //Constructor que recibe un parámetro tipo entero.
     public Multiplica(int num);
     {
         Int result = num * 10;
         Console.WriteLine("El resultado es:" + result);

     }
 }

 class ProgramConstIni
 {

     static void Main(string[] args)
     {

             //En la siguiente  línea se invoca al constructor de la clase Multiplica
             //automáticamente al crear el objeto "x" mediante la sentencia "New"

             Multiplica x = new Multiplica(24); //Se envía como parámetro el numero = 24
     }
 }
```

Imagen 4. Ejemplo de constructor de clase con parámetros.

Ahora, **consulta** a Sharp J. & Jagger J. (2002, pp. 30-34) quien menciona como iniciar un primer programa en C# en consola.

Ceballos, F. (pp. 90 - 97) por su parte, muestra ejemplos adecuados y fáciles de digerir referentes a objetos y constructores, también **consulta** dicho texto.

Finalmente, no dejes de **visitar** la librería MSDN de Microsoft, ahí encontrarás ejemplos que van desde el esqueleto de un programa hasta orientados a la declaración de objetos y constructores que te ayudarán a entender mejor cada tema visto en este apartado.

Es importante tener en cuenta cómo se declaran los objetos y cómo se declaran los constructores, pero más importante cómo se utilizan, por lo cual te invito a que no sólo te quedes con lo que se te explica en esta unidad, si no que indagues en la utilidad de éstos, tan interesantes, procesos de programación.

Durante el trascurso de los temas vistos en esta unidad, revisaste características importantes del lenguaje .NET, tomando en cuenta principalmente que en él se puede desarrollar, aplicaciones para usos exclusivos los cuales son utilizados en la actualidad, tales como desarrollos móviles o web, además de que es de suma importancia instalar la aplicación porque de ello depende que puedas realizar las diversas actividades que se presentan a lo largo de la unidad.

Cierre de la unidad

Durante el desarrollo de esta unidad has conocido una nueva herramienta de programación, retomando algunos términos que, si bien ya se habían estudiado en otras unidades didácticas, es necesario volver a recordar.

Con el estudio de esta primera unidad tienes las bases necesarias para comenzar a trabajar en las siguientes unidades, debido a que con los temas vistos y las actividades que se te presentan comenzarás a familiarizarte con la plataforma, siendo esto punto de partida para lograr buenos y útiles desarrollos.

Te invito a que, si tienes dudas o no comprendiste del todo alguno de los temas presentados, retomes nuevamente lectura de dichos temas, esto te ayudará a comprender de forma más clara y ordenada las ideas que se plasman y que se desean entiendas y apliques en programación.

Para saber más…

Si deseas saber más acerca de la instalación o configuración de Visual Basic .NET o sobre algunos de los temas aquí descritos, consulta en la web en los siguientes sitios electrónicos.

Fundamentos de C# https://csharp.com.es/fundamentos-del-lenguaje-c/

Guía de Programación en C# https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/

Charles Petzold (2006), Manual *de referencia .NET Book Zero* http://ftp.tekwind.co.jp/pub/asustw/nb/Z93E/s2162_z93_hw.pdf

Para consultar otro lenguaje en .NET, revisa el documento: Elementos de Visual Basic .NET(Rosario y Hernández, 2007)

Fuentes de consulta

Ceballos, F (2007). *Enciclopedia de Microsoft Visual C#,* (2da. Ed.). España: Ed. Alfaomega Ra-MA.

Hernández López, S. (2007). Mejora e implementación del sistema de control del posgrado en el Instituto Mexicano del Petróleo vía Cliente/Servidor en Plataforma .NET. México: UNAM. http://132.248.9.195/ptd2009/mayo/0642804/Index.html

MSDN Microsoft Developer Network (2007a). *Cómo: Crear una aplicación de consolade C#.* https://msdn.microsoft.com/es-es/library/0wc2kk78(v=vs.90).aspx

MSDN Microsoft Developer Network (2007b). *Constantes y variables (Visual C# Express).* https://docs.microsoft.com/es-es/previous-versions/visualstudio/visual-studio-2008/wew5ytx4(v=vs.90)?redirectedfrom=MSDN

MSDN Microsoft Developer Network (2023a). *Conversiones de tipos. (Guía de programación de C#).* https://msdn.microsoft.com/es- es/library/ms173105.aspx

MSDN Microsoft (2012). *Guía de programación en C#.* https://www.lawebdelprogramador.com/pdf/8631-C-Guia-Total-del- Programador.html

MSDN Microsoft Developer Network (2023b). *Palabras clave de C#.* https://docs.microsoft.com/es-es/dotnet/csharp/language- reference/keywords/index

Ramírez, F (2007). *Introducción a la programación. Algoritmos y su implementación enVB .NET, C# .NET, Java y C++,* (2da. Ed.). España: Alfaomega.

Rosario García, L., y Hernández Martínez, H. (2007). *Sistema administrador de centrosde cómputo para nivel bachillerato sobre plataforma .NET*. México: UNAM. P. 44- 47. http://132.248.9.195/pd2007/0615306/Index.html

Shart, J & Jagger, J (2002). *Microsoft VISUAL C#. NET, APRENDA YA*. España: McGraw Hill.

Visual Studio Microsoft (2022). *Requisitos del sistema para Visual Studio 2022. Visual Studio Express 2022 para escritorio de Windows.*

https://visualstudio.microsoft.com/es/vs/older-downloads/