# 1. FAMILIARIZACIÓN CON EL ENTORNO DE TRABAJO

¿Qué ofrece la programación en AutoCAD?
¿Qué necesito para empezar a programar?
¿Qué es C# y .NET?

Autor: Jorge A. Díez Pomares
Laboratorio de C.A.D. UMH

# Índice:

# 1.1. INTRODUCCIÓN

- AutoCAD: Software CAD utilizado ampliamente por Arquitectos, Ingenieros, Diseñadores…
- A diferencia de otros software CAD, permite una amplia personalización del entorno de trabajo gracias a la posibilidad de programar sobre él.
- Lenguajes de programación: Script, AutoLISP, C++, VisualBasic, C#.
- Desde programas sencillos para tareas puntuales y/o repetitivas (Ej: Dibujar catenarias), hasta programas complejos que amplíen sus funcionalidades (Ej: Corrector en desarrollo por el Laboratorio de CAD UMH).

# 1.2. SOFTWARE NECESARIO

- AutoCAD
- .NET Framework 4.0
- Entorno de desarrollo (IDE):
  - MS Visual Studio (Express)
- API de AutoCAD
  - ObjectARX

# 1.2. SOFTWARE NECESARIO

- AutoCAD:
  - Para estudiantes UMH, versión de estudiantes en: http://students.autodesk.com/
  - Versión de prueba 30 días: http://www.autodesk.com/products/autodesk-autocad/free-trial

- .NET Framework:
  - CLR, bibliotecas, lenguajes de programación (C#, VB.NET…)
  - Descarga: http://www.microsoft.com/es-es/download/details.aspx?id=17851

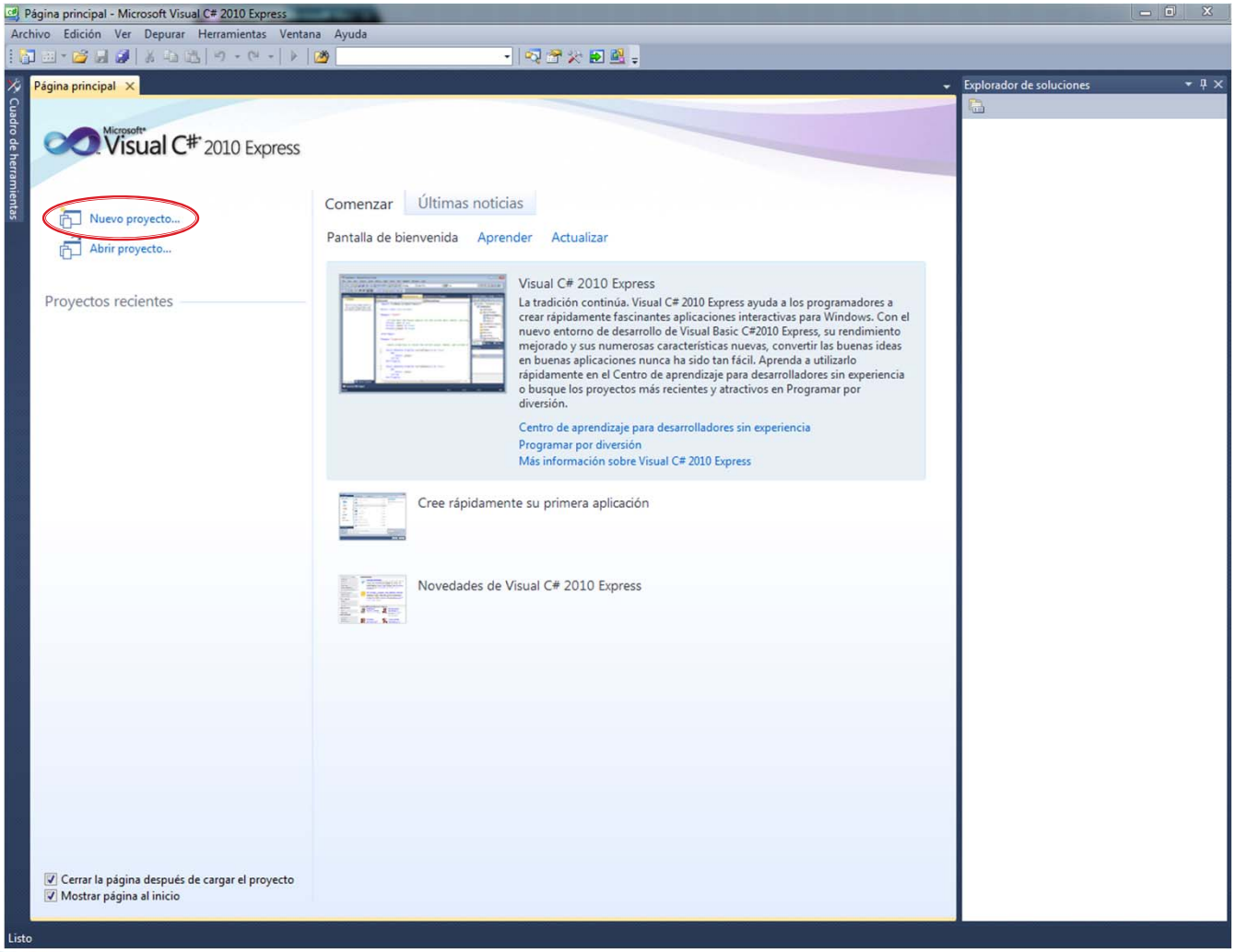
# 1.2. SOFTWARE NECESARIO

- ENTORNO DE DESARROLLO:
  - Microsoft Visual C# Express 2010: http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express
  - Microsoft Visual Studio Express 2012: http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products (No funciona en Windows XP o anterior)
  - Estudiantes UMH, MS Visual Studio en MSDN.

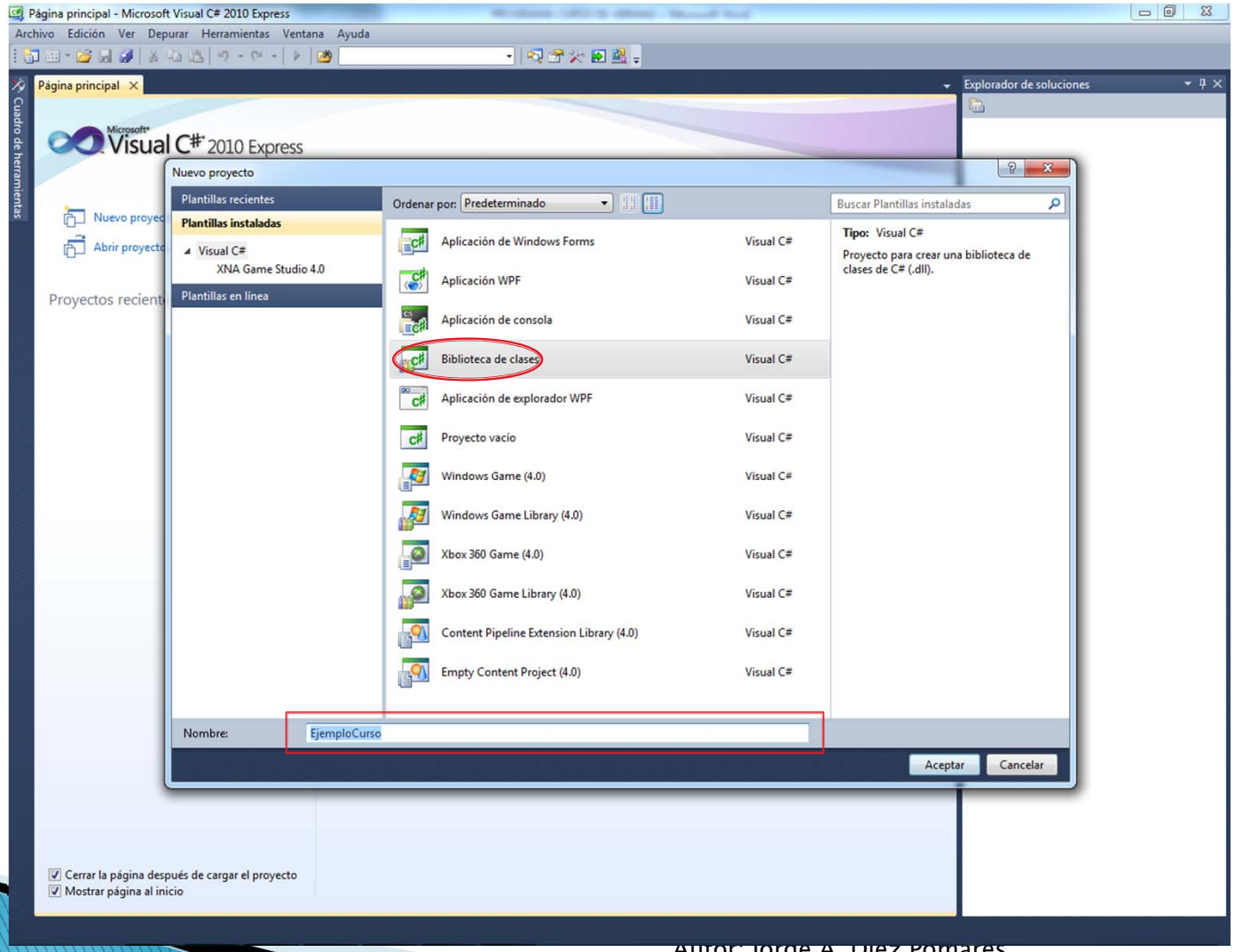
# 1.2. SOFTWARE NECESARIO

- API de AutoCAD: ObjectARX 2012
  - Conjunto de librerías
  - Escritas en C++
  - Incluye librerías que adaptan las funciones en C++ a lenguajes .NET (VB.NET y C#)
  - Implementan funciones de:
    - Gestión de base de datos .dwg
    - Interacción con usuario
    - Álgebra lineal y operaciones geométricas
    - Personalización de interfaz gráfica
    - Representación gráfica
    - Publicación e impresión
  - Documentación que explica las funciones y parámetros (Integrable en MS Visual Studio).
  - Descarga: Autodesk Developer Network

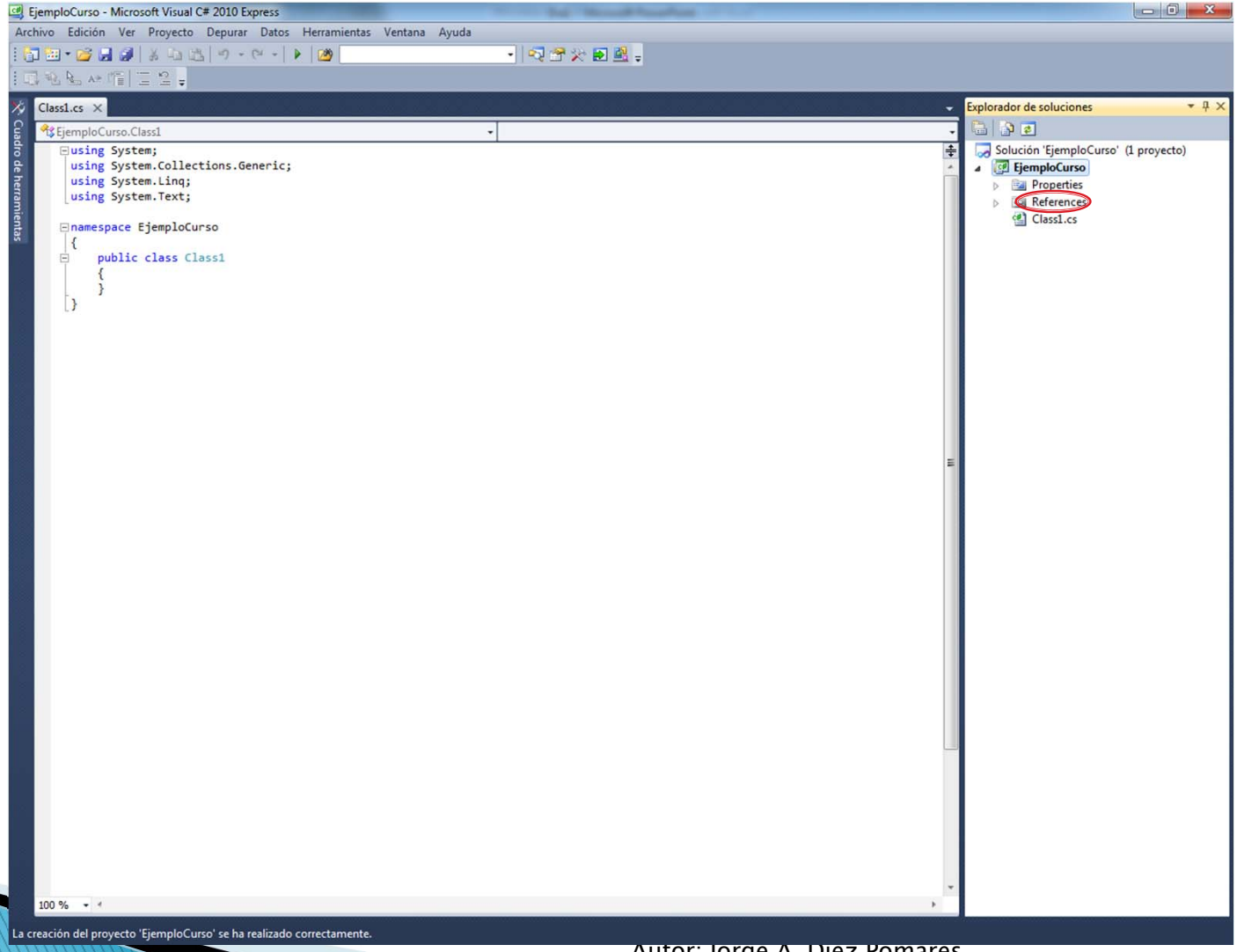
# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010



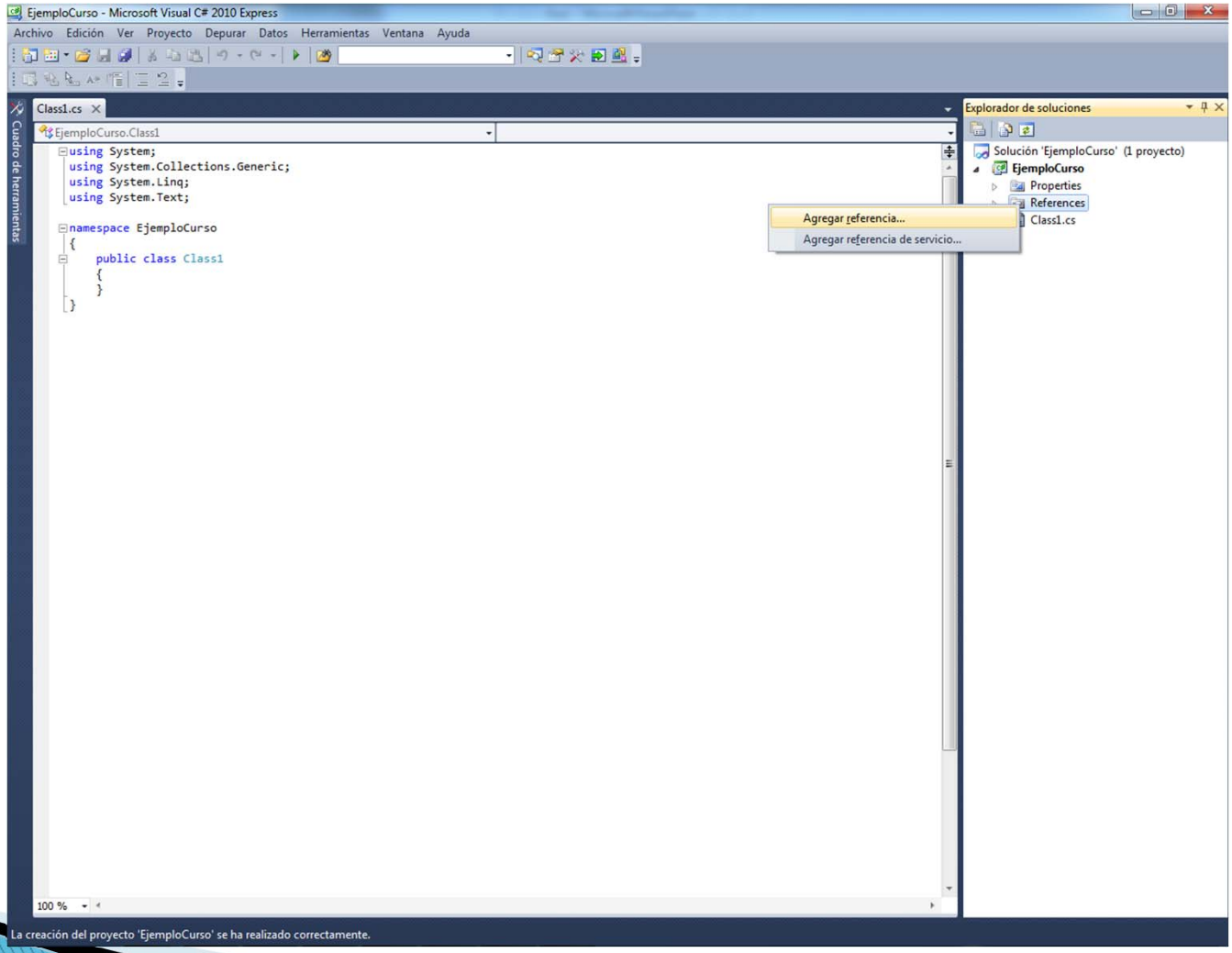
# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010



# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010

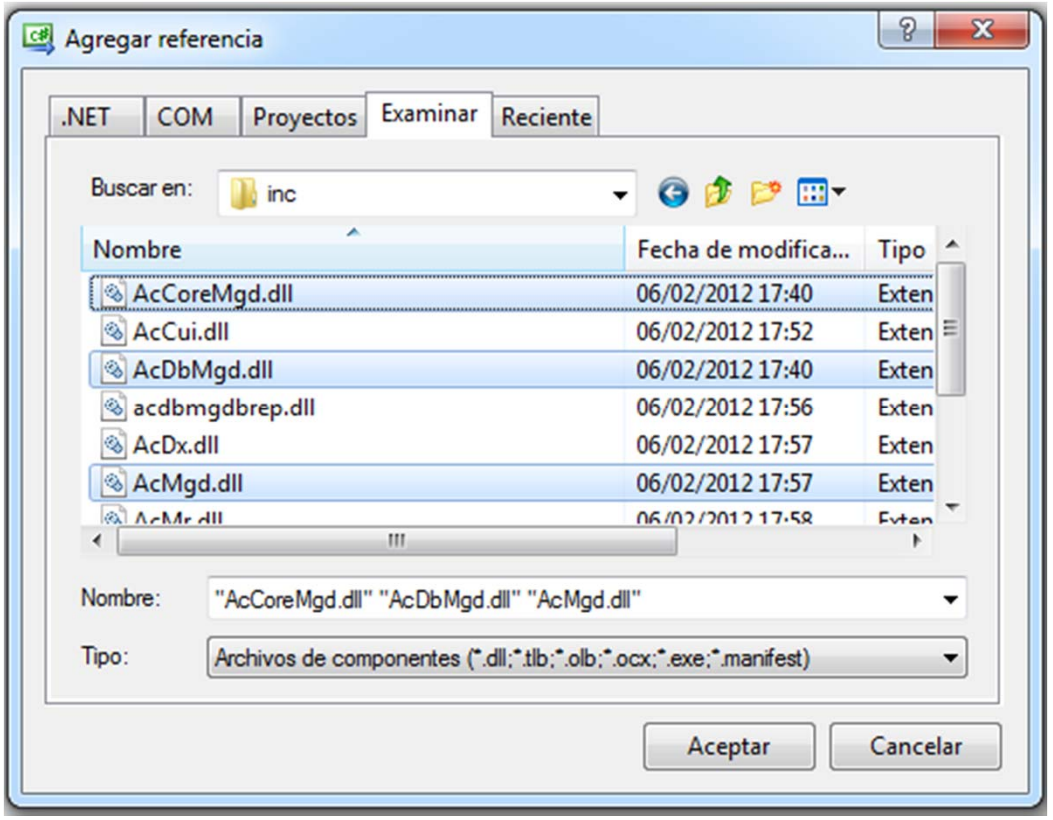


# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010

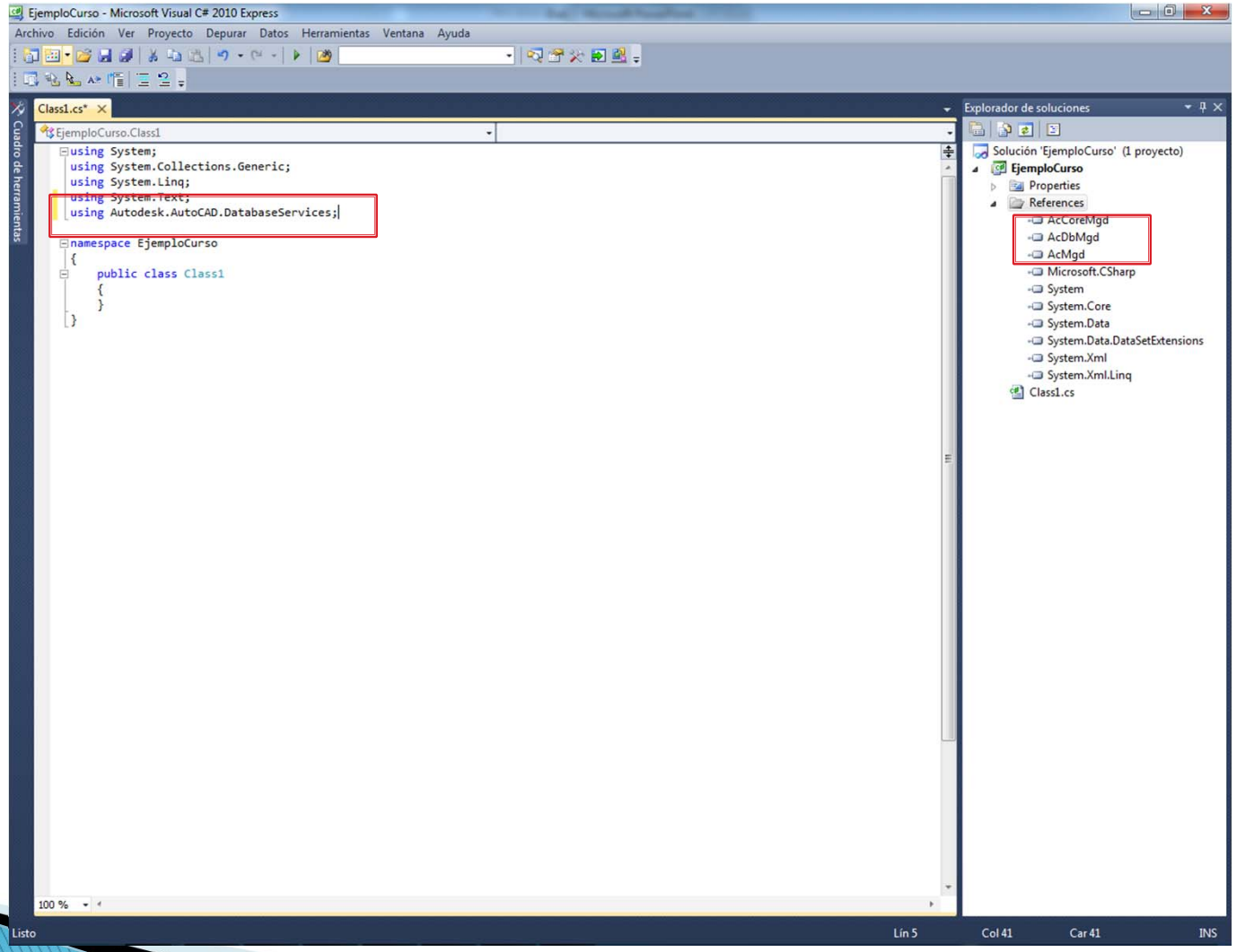


# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010

Las bibliotecas de clases para .NET se encuentran en los directorios “inc” “inc-win32” e “inc-x64” de la carpeta ObjectARX



# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010

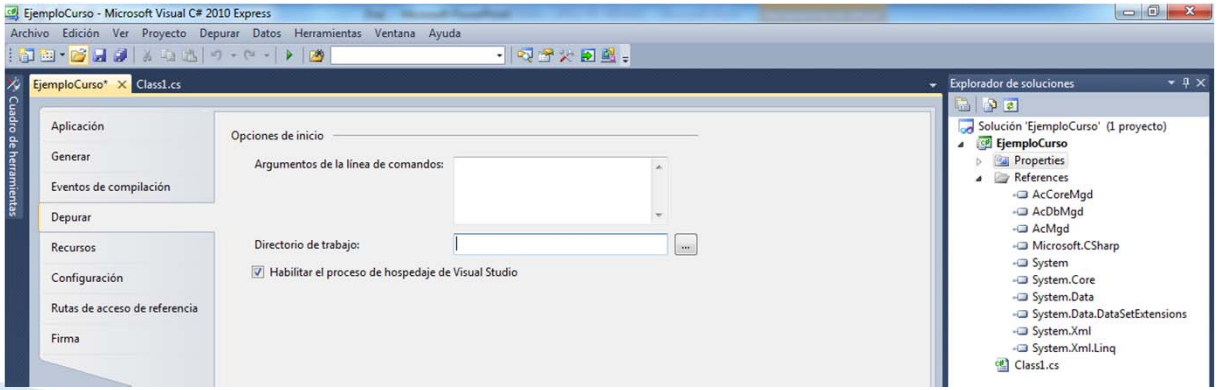


# 1.3. CONFIGURACIÓN DEL PROYECTO EN VISUAL STUDIO 2010

- Iniciar AutoCAD durante la depuración en Visual C# Express 2010:
  - Añadir a la carpeta del proyecto un archivo llamado "NombreDelProyecto.csproj.user"
  - Escribir en él:

```
<?xml version="1.0" encoding="utf-8"?>
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
<StartAction>Program</StartAction> <StartProgram>"RUTA /acad.exe"</StartProgram>
</PropertyGroup> </Project>
```

  - Añadir el directorio de trabajo: C:\RutaAutoCAD\UserDataCache\



# 1.4. INTRODUCCIÓN A C#

## ¿Qué es C#?

- C#: Lenguaje de programación orientado a objetos, desarrollado por Microsoft
- Parte de la plataforma .NET
- Basado en C/C++, similar a Java incluyendo mejoras de otros lenguajes
- Código fuente se compila a lenguaje intermedio común (CIL) interpretado. Permite integrar varios lenguajes en la misma aplicación (VB.NET, C#, J#...)
- Simple, moderno y de propósito general. Desde aplicaciones sencillas hasta sistemas operativos y programas distribuidos.

# 1.4.1. INSTRUCCIONES Y LÉXICO ELEMENTAL

- Léxico basado en C/C++
- Declaración de variables: *tipo nombre;*
- Operadores básicos:

| Categoría | Operadores |
|---|---|
| Aritmético | + - * / % |
| Lógico y a nivel de bits | ^ ! ~ && \|\| |
| Concatenación | + |
| Incremento, decremento | ++ -- |
| Desplazamiento | << >> |
| Relacional | == != < > <= >= |
| Asignación | = ^= <<= >>= |
| Acceso a miembro | . |
| Indexación | [ ] |
| Conversión | ( ) |
| Condicional | ? : |
| Creación de objeto | new |
| Información de tipo | as is sizeof typeof |

# 1.4.1. INSTRUCCIONES Y LÉXICO ELEMENTAL

- Instrucciones de control de flujo de programa: permiten tomar decisiones y/o ejecutar repetidas veces el mismo código.
  - Condicionales: Se ejecuta cierto código si se cumple una condición
    - *If(condición) {código}:* Se ejecuta el código si se cumple la condición.
    - *If(condición) {código} else {código'}* :Si se cumple condición se ejecuta código, si no se ejecuta código'
    - *If(condición1) {código 1} else if(condición 2) {código 2} … else if(condición n) {código n}* :Si se cumple condición 1 se ejecuta código 1, si se cumple condición 2 se ejecuta código 2,etc. (¡Ojo! Sólo se ejecuta el código de la primera condición que se cumple).

# 1.4.1. INSTRUCCIONES Y LÉXICO ELEMENTAL

- Instrucciones de control de flujo de programa: permiten tomar decisiones y/o ejecutar repetidas veces el mismo código.
  - Selección de casos: Útil para crear menús de selección.
    - *Switch (variable) {case "caso1": …break};* *Los casos pueden estar definidos tanto por enteros (1,2,3..) como por caracteres ('a','b', 'c'…) como por cadenas de texto ("Caso1","Caso2"…)*

**Ejemplo:**

```
switch (Opcion)
{ case "Opcion1":
...
 break;
case  "Opcion2":
...
break;
default:
 ...
break;
}
```

# 1.4.1. INSTRUCCIONES Y LÉXICO ELEMENTAL

- Instrucciones de control de flujo de programa: permiten tomar decisiones y/o ejecutar repetidas veces el mismo código.
  - Bucles: Ejecución repetida de un determinado fragmento de código mientras se cumpla cierta condición. Útil para aplicar el mismo algoritmo a distintos datos.
    - *while(condición){código}:* Se ejecuta código mientras se cumpla la condición
    - *do{código} while(condición);* : Ídem que el anterior pero la condición se comprueba al final, por lo que siempre se ejecuta al menos una vez.
    - *For(int contador=inicial ; condición; variación de valor de contador) {código}*: Similar al bucle while, pero este bucle permite actualizar el valor de la variable contador al final de éste. Muy utilizado para recorrer listas de elementos (Array, List…) cuyo tamaño vaya cambiando.
      - Ejemplo: *for(int i=0; i<=array.Count; i++){ código }*
    - *Foreach( tipo nombre in listaobjetos) {código}*: Recorre cada uno de los objetos de la lista "listaobjetos" y los asigna a la variable "nombre". Permite trabajar rápidamente con todos y cada uno los elementos de una lista cuyo tamaño y orden no va a ser alterado.

# 1.4.2. TIPOS DE VARIABLES: ALFANUMÉRICAS, ARRAYS Y LISTAS

- Literal: Valor fijo expresado explícitamente en código, una vez compilado el programa su valor es fijo. Para cambiarlo es necesario modificar el código y recompilarlo.
  - Ejemplo: 5, 'a', "Hola mundo"
- Constante: Valor fijo al que nos referimos a través de un nombre alfanumérico.
  - *const tipo nombre = Valor*; :Cuando en nuestro programa escribamos el nombre de la constante, en compilación se sustituirá por su valor.
  - *readonly tipo nombre;* :Puede asignársele el valor después de declararse, pero sólo una vez.
- Variable: Espacio de memoria con el tamaño adecuado al tipo, al cual nos referimos por un nombre alfanumérico. El valor que toma puede variar durante la ejecución del programa.
  - *tipo nombre;*

# 1.4.2. TIPOS DE VARIABLES: ALFANUMÉRICAS, ARRAYS Y LISTAS

- Tipos principales de de Variables:
  - *bool*: Representa un único bit que puede tomar el valor 1 ó 0.
  - *byte*: Representa un conjunto de 8 bits, desde 00000000 a 11111111.
  - *int*: Representa un número entero de 32 bits.
  - *long*: Número entero de 64 bits (mayor rango de valores que int pero ocupa más memoria).
  - *short*: Número entero de 16 bits.
  - *float*: Número en coma flotante (real) de 32 bits.
  - *double*: Número en coma flotante de 64 bits (mayor precisión decimal).
  - *char*: Carácter Unicode de 16 bits (A diferencia de la mayoría de lenguajes, puede representar la ñ).
  - *string*: Cadena de caracteres Unicode de 16 bits, permite representar textos.
  - Las variables que representan números enteros pueden representar únicamente enteros positivos si en el tipo se añade el prefijo “u”: uint, ushort, ulong.

# 1.4.2. TIPOS DE VARIABLES: ALFANUMÉRICAS, ARRAYS Y LISTAS

- Conversión entre tipos de variables (casting):
  - Implícita: Ciertas variables pueden convertirse implícitamente simplemente realizando la asignación que se desea. No hay pérdida de información.
    - a=b donde a es de tipo entero y b es de tipo short.
  - Explícita: Se especifica el tipo al que se quiere convertir escribiendo este entre paréntesis. Puede haber pérdida de información.
    - *tipo1 a= (tipo1)b*; donde b es una variable de tipo2.

Conversiones de tipo de datos (Fuente Wikipedia): A (impícita), E (explícita), I (incompatible)

| | byte | sbyte | short | ushort | int | uint | long | ulong | float | double | decimal | char | bool |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| byte | | E | A | A | A | A | A | A | E | E | E | E | I |
| sbyte | E | | A | E | A | E | A | A | E | E | E | E | I |
| short | E | E | | E | A | A | A | A | E | E | E | E | I |
| ushort | E | E | E | | A | A | A | A | E | E | E | E | I |
| int | E | E | E | E | | E | A | A | E | E | E | E | I |
| uint | E | E | E | E | E | | A | A | E | E | E | E | I |
| long | E | E | E | E | E | E | | E | E | E | E | E | I |
| ulong | E | E | E | E | E | E | E | | E | E | E | E | I |
| float | E | E | E | E | E | E | E | E | | A | E | I | I |
| double | E | E | E | E | E | E | E | E | E | | E | I | I |
| decimal | E | E | E | E | E | E | E | E | E | E | | I | I |
| char | E | E | E | A | A | A | A | A | A | A | A | | I |
| bool | I | I | I | I | I | I | I | I | I | I | I | I | |

# 1.4.2. TIPOS DE VARIABLES: ALFANUMÉRICAS, ARRAYS Y LISTAS

- Arrays: Lista de valores de cierto tipo, a los que se puede acceder mediante un índice, que comienza por 0. Se utilizan cuando se necesita un conjunto de valores de dimensión conocida.
  - Declaración:
    - tipo[] array = new tipo[dimensión];
    - tipo[] array = new tipo{ valor1, valor2, valor3};
    - tipo[,] array = new tipo[dimension1,dimension2];
    - tipo[,] array= new tipo{ {valor11,valor12}, {valor21,valor22}};
  - Acceso a elementos:
    - array[0]; : Elemento 0 de array
    - array [1,2]; :Elemento (1,2) de array

# 1.4.2. TIPOS DE VARIABLES: ALFANUMÉRICAS, ARRAYS Y LISTAS

- Listas: Lista de valores de cierto tipo, a los que se puede acceder mediante un índice, que comienza por 0. A diferencia de los array, las listas varían dinámicamente su tamaño.
  - Declaración:
    - List<tipo> lista=new List<tipo>();
  - Adición de elementos:
    - Lista.Add(tipo)
  - Acceso a elementos:
    - Lista[0];

# 1.4.3. MANEJO BÁSICO DE FICHEROS

- Ficheros: Almacenamiento no volátil de información.
- .NET incluye librerías de gestión de ficheros.
  - System.IO.FileInfo
  - System.IO.DirectoryInfo
  - System.IO.DriveInfo
  - System.IO.Directory
  - System.IO.File
- Creación:
  - System.IO.FileStream fs = System.IO.File.Create(Ruta);
- Escritura:
  - System.IO.StreamWriter file = new System.IO.StreamWriter(ruta, true);
  - file.WriteLine("Fourth line");
- Lectura:
  - System.IO.StreamReader file = new System.IO.StreamReader(ruta);
  - file. ReadLine();
- Es importante cerrar el fichero para liberar su uso a otras aplicaciones. Métodos: file.Close() y file.Dispose()
- Más información en la guía de programación C# de Microsoft.

# CASO PRÁCTICO

- Escribir un programa en C# que implemente:
  - Un menú con 3 opciones:
    - 1.Escribir las cadenas que introduzca el usuario por consola de comandos.
    - 2.Leer enteros de un fichero dado y mostrarlo por pantalla, cada uno en una línea distinta.
    - 3.Salir.
  - En la opción 1, el usuario debe introducir primero la ruta y nombre del fichero a crear.
  - En la opción 2, el usuario especificará la ruta donde se encuentra el fichero a leer.
  - Si el fichero ya existe, se le pedirá al usuario que introduzca otro nombre.
  - Si el fichero a leer no existe se le pedirá al usuario de nuevo el nombre del fichero o se le dará la opción de salir, si pulsa Q.