

A decorative graphic on the right side of the page. It features three blue circles of different sizes, each composed of concentric circles in varying shades of blue. Two thin blue lines intersect at a point near the top center, forming a large 'V' shape that frames the circles. The circles are positioned in the upper right, middle right, and lower right areas of the page.

Breve Tutorial de Visual Studio

Modelos Y Algoritmos para videojuegos I

El propósito de este tutorial es el de explicar el uso de las características básicas del IDE Visual Studio que necesitaremos para realizar las primeras practicas.

Pablo Sebastián Rojas Fredini & Emmanuel Rojas Fredini

En este tutorial solo veremos algunas características básicas de Visual Studio.

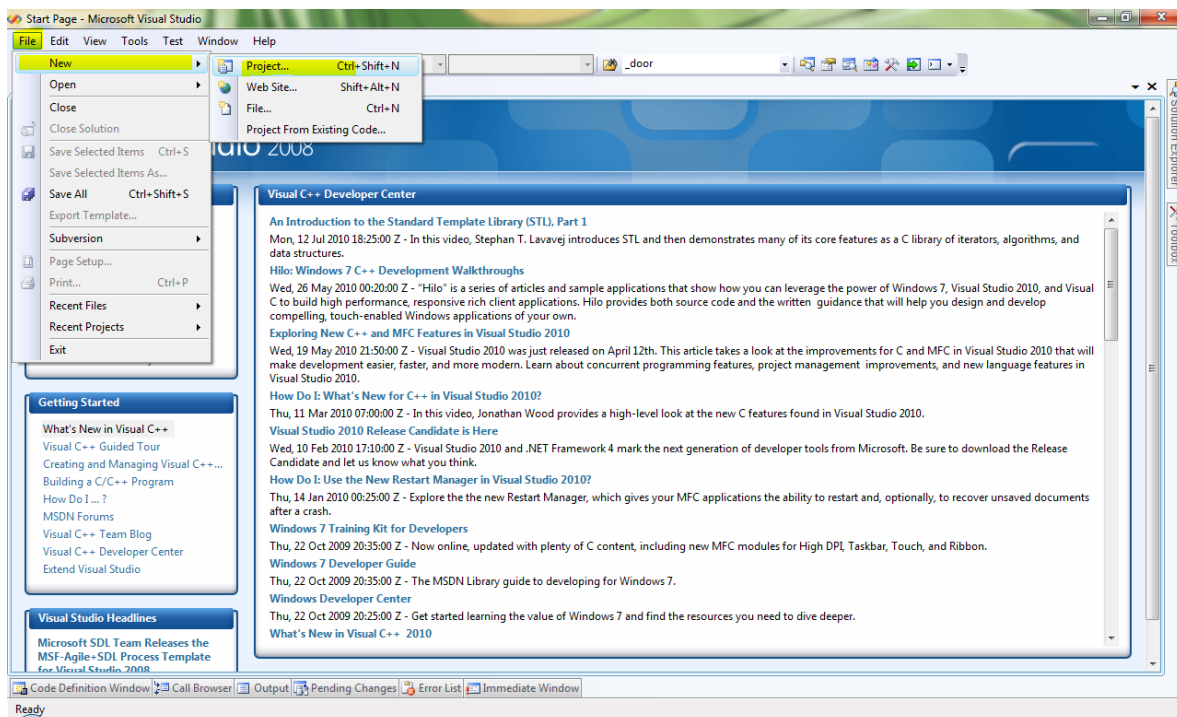
Las capturas del tutorial fueron tomadas de Visual Studio 2008 Profesional en ingles, por esto puede que muchos noten alguna pequeña diferencia con su instalación. El caso más obvio es el idioma, pero además pueden notar que si tiene una versión Express algún menú sea de otro color. No obstante todo el resto de los menús son idénticos.

Los Primeros Pasos:

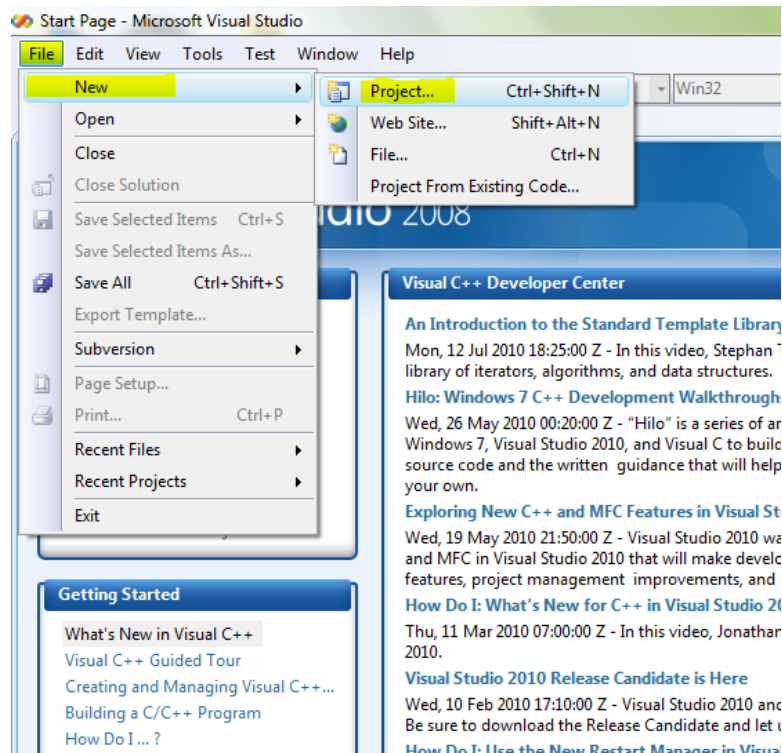
Para crear un proyecto en Visual Studio debemos ir a:



Esto abrirá una ventana de dialogo donde nos pedirá el tipo de proyecto (llamado plantilla de proyecto), esto lo que hará será crearnos los archivos (.cpp y .h) básicos y configurarnos el proyecto en general para ese tipo de proyecto.



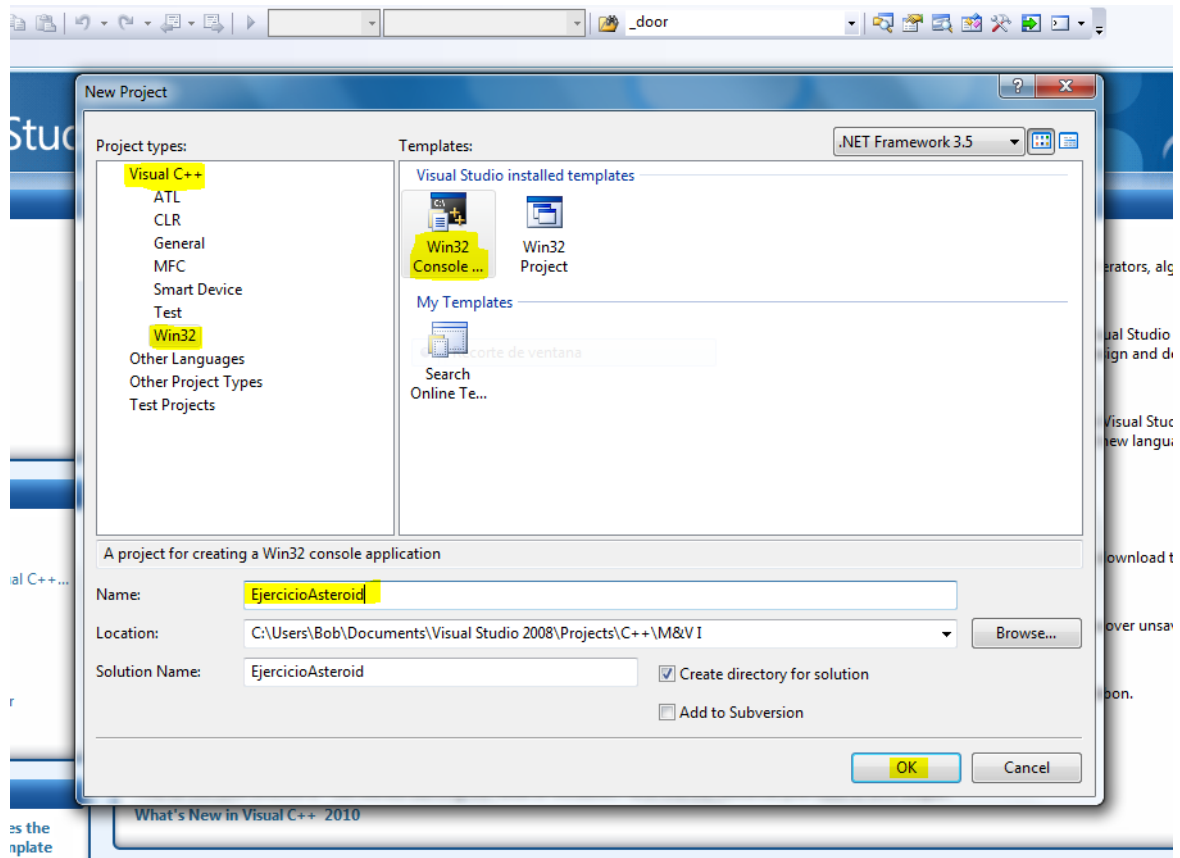
Nuevo Proyecto



Zoom a Nuevo Proyecto

Entonces dentro del dialogo de creación de plantilla seguiremos los siguientes pasos:

1. En la ventana izquierda de selección de tipo de plantilla elegiremos dentro de **Visual C++** (aquí están todas las plantillas de C++) la opción **Win32**.
2. Luego en la ventana derecha elegiremos **Consola Win32**, ya que por el momento no usaremos ningún entorno grafico .Luego en la caja de texto de la parte inferior debemos ingresar el nombre del proyecto. En nuestro caso EjercicioAsteroid.
3. Aceptamos la selección con el Botón **OK**.



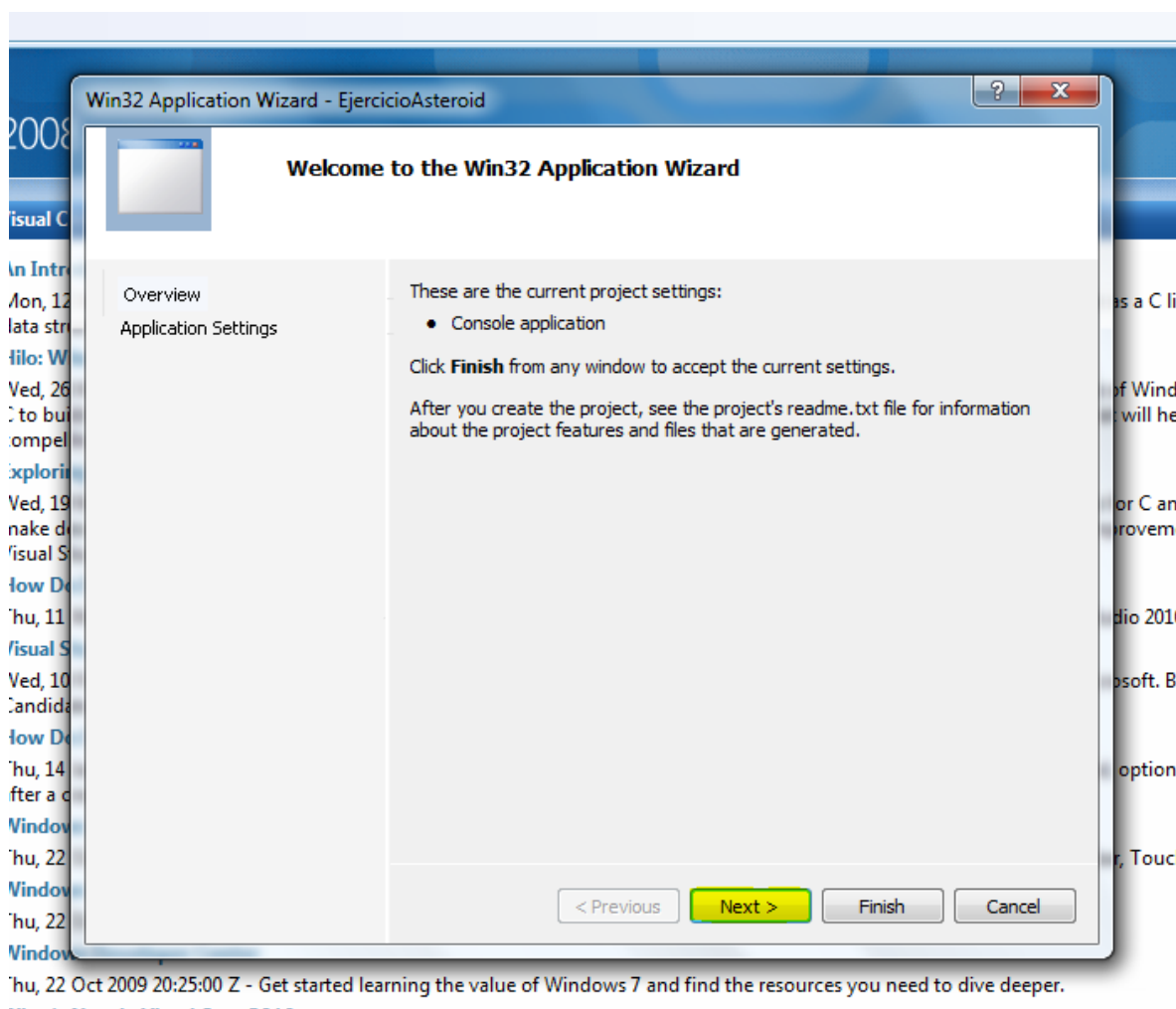
Nuevo Proyecto de Consola

Opcionalmente podemos:

1. Elegir la ubicación donde se creará el proyecto. Por defecto esto es en la carpeta Visual Studio 2008-> Proyectos dentro de Mis Documentos.
2. Seleccionar otro nombre de Solución. Una solución es un grupo de proyectos, nosotros no utilizaremos soluciones con más de 1 proyecto ya que no nos serían de ninguna utilidad, pero para un proyecto grande puede ser más ordenado.

Luego seguiremos con el dialogo de creación de proyecto. En esta ventana solo nos dirá información de nuestra selección.

A continuación seleccionaremos en Siguiente para pasar a la próxima etapa de configuración.

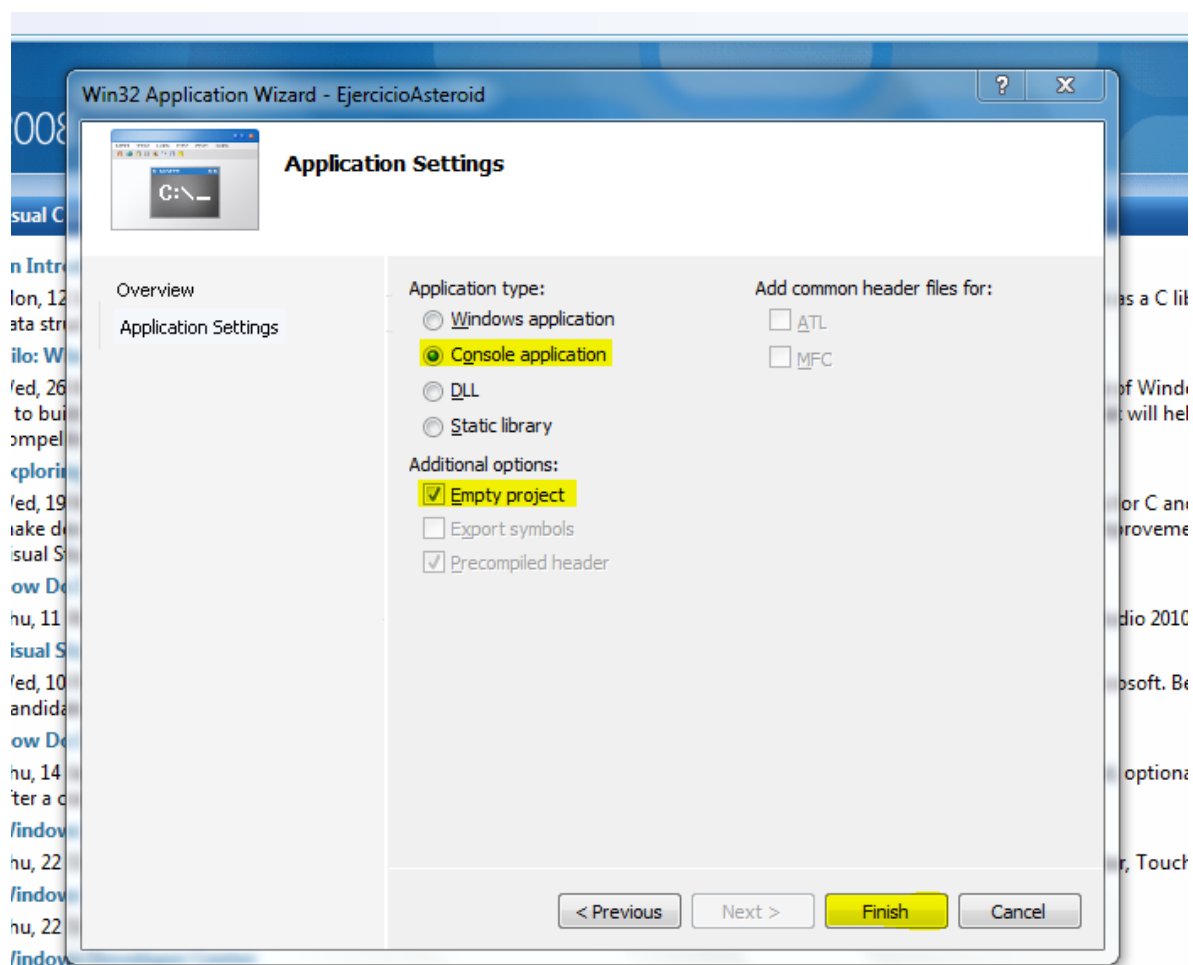


Nota: Si seleccionamos Finalizar nos creara el proyecto con una configuración por defecto, pero nosotros queremos hacer unos cambios antes.

En este paso configuraremos que tipo de proyecto de consola queremos.

Navegaremos dentro de la sección Opciones Adicionales: **Proyecto Vacío**.

Siguiente finalizaremos la creación con el botón Finalizar de la sección inferior del dialogo.

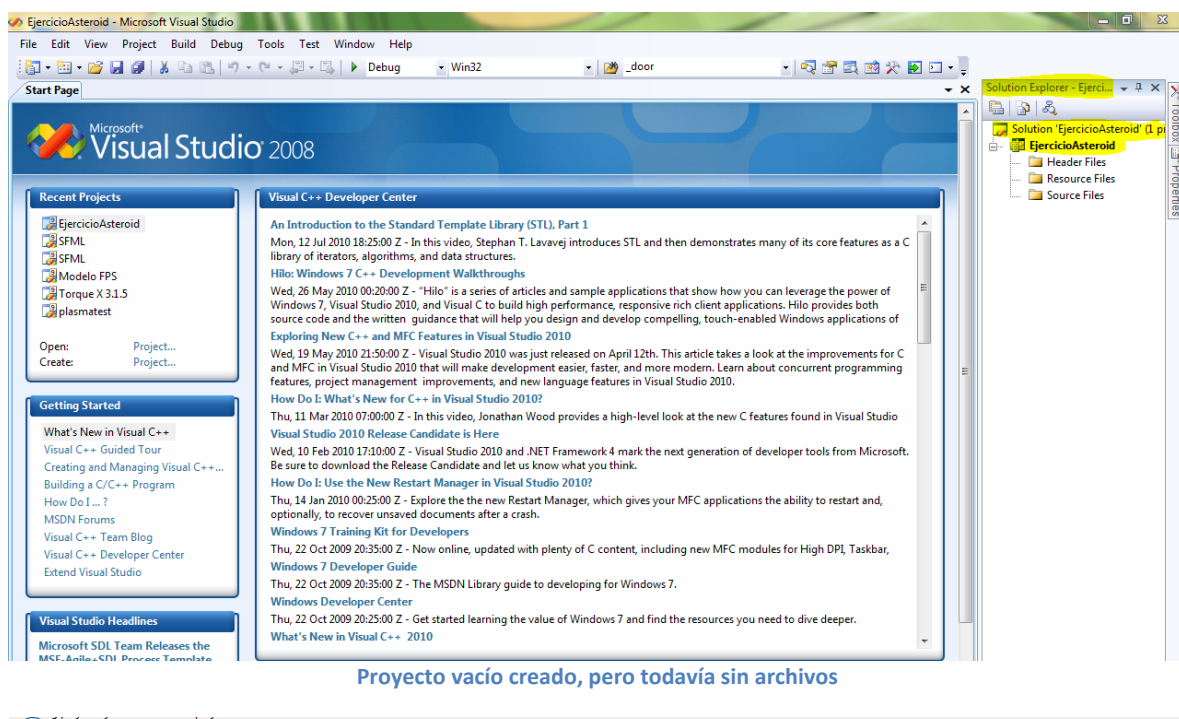


hu, 22 Oct 2009 20:25:00 Z - Get started learning the value of Windows 7 and find the resources you need to dive deeper.

A la derecha de la captura vemos Explorador de Solución, en este podemos ver con una estructura de árbol (similar a la del Explorador de Windows) la estructura de nuestra solución. En nuestro caso solo hay un proyecto, EjercicioAsteroid, y dentro de este hay 3 carpetas:

1. **Headers Files:** Aquí pondremos nuestros .h con las definiciones de clases y funciones.
2. **Resource Files:** Aquí pondremos los recursos que utilizaremos en nuestros proyectos (imágenes, iconos, etc.). Por el momento no pondremos ninguno.
3. **Source Files:** Aquí pondremos nuestros .cpp con las implementaciones de clases, funciones y si tenemos variables globales lo haremos aquí también.

En nuestro caso como elegimos un proyecto vacío vemos en la parte derecha de la captura que no hay ningún archivo en el explorador de solución.

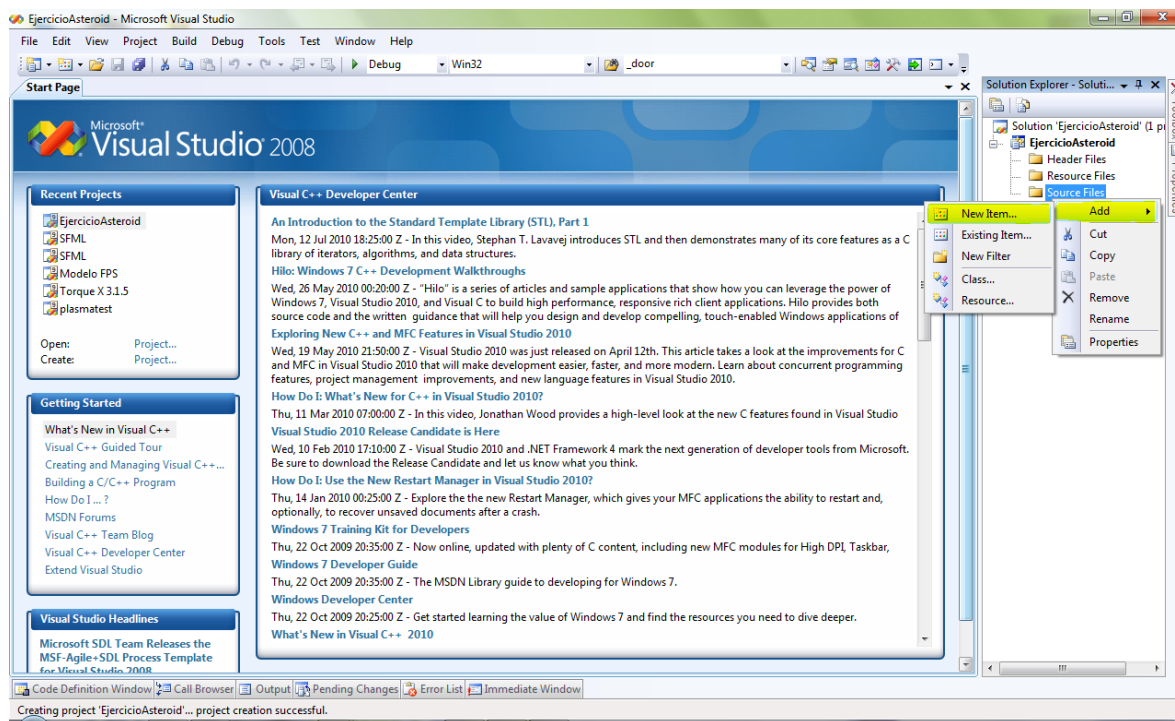


Todavía no tenemos ningún .h ni .cpp en el proyecto, debemos agregar uno.

Antes que nada agregaremos un .cpp para nuestra función de entrada al programa, es decir la función **Main**. Seleccionaremos:



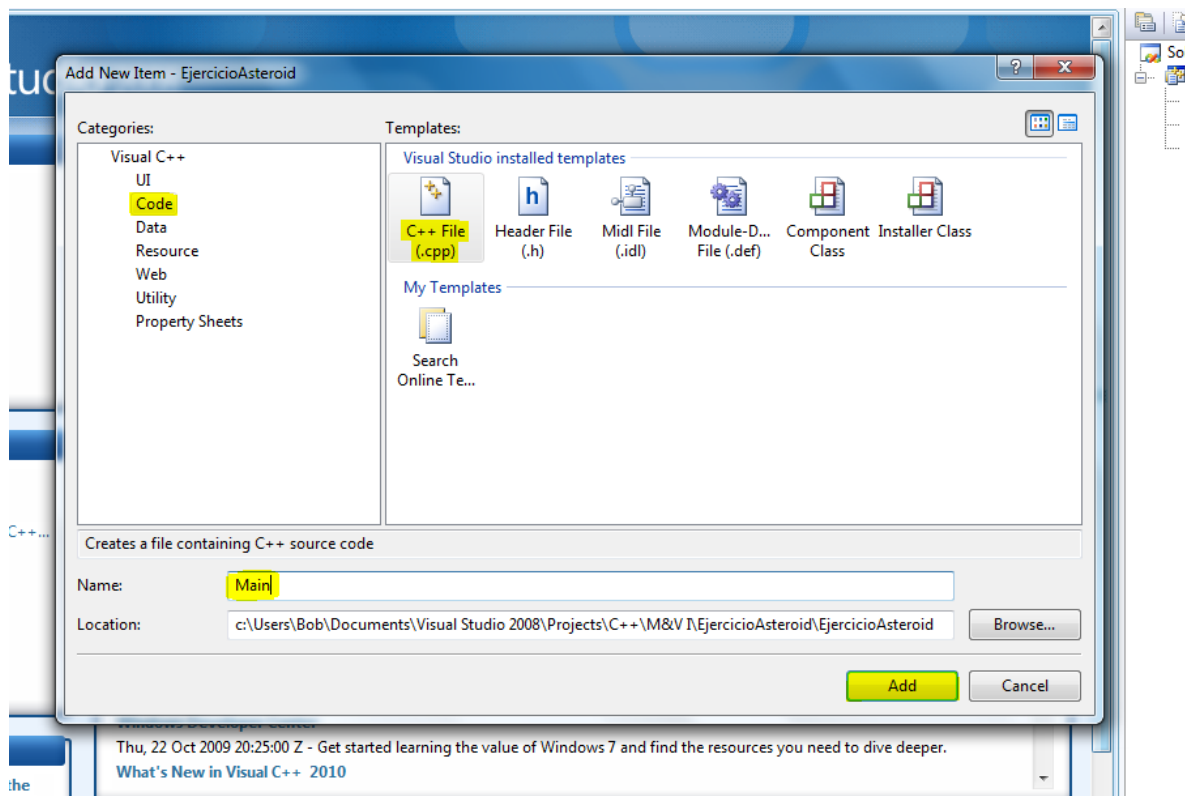
Esto nos abrirá un dialogo para seleccionar el tipo de archivo.



Creando un archivo nuevo

Entonces dentro del dialogo de selección de tipo de archivo:

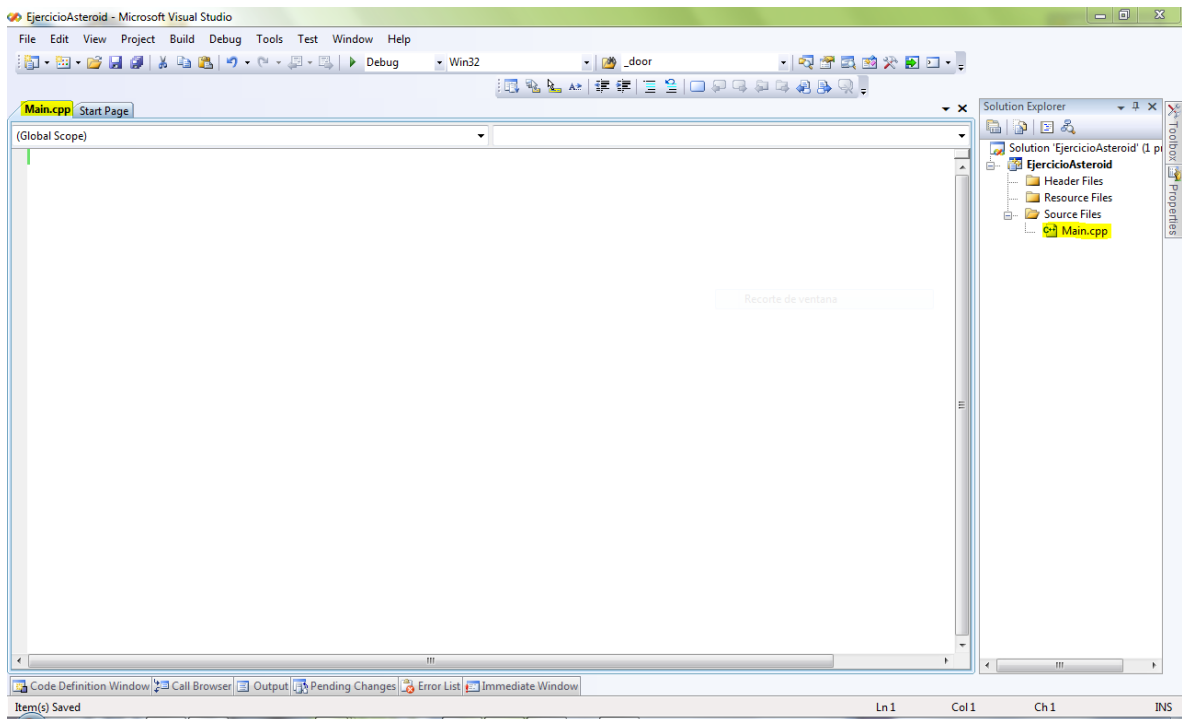
1. En la ventana izquierda de selección de tipo de archivo elegiremos dentro de **Visual C++**, la opción **Code**.
2. Luego en la ventana derecha elegiremos **Archivo C++ (.cpp)**.
3. Luego en la caja de texto de la parte inferior debemos ingresar el **nombre del archivo nuevo**. En nuestro caso Main.
4. Aceptamos la selección con el Botón **Agregar**.



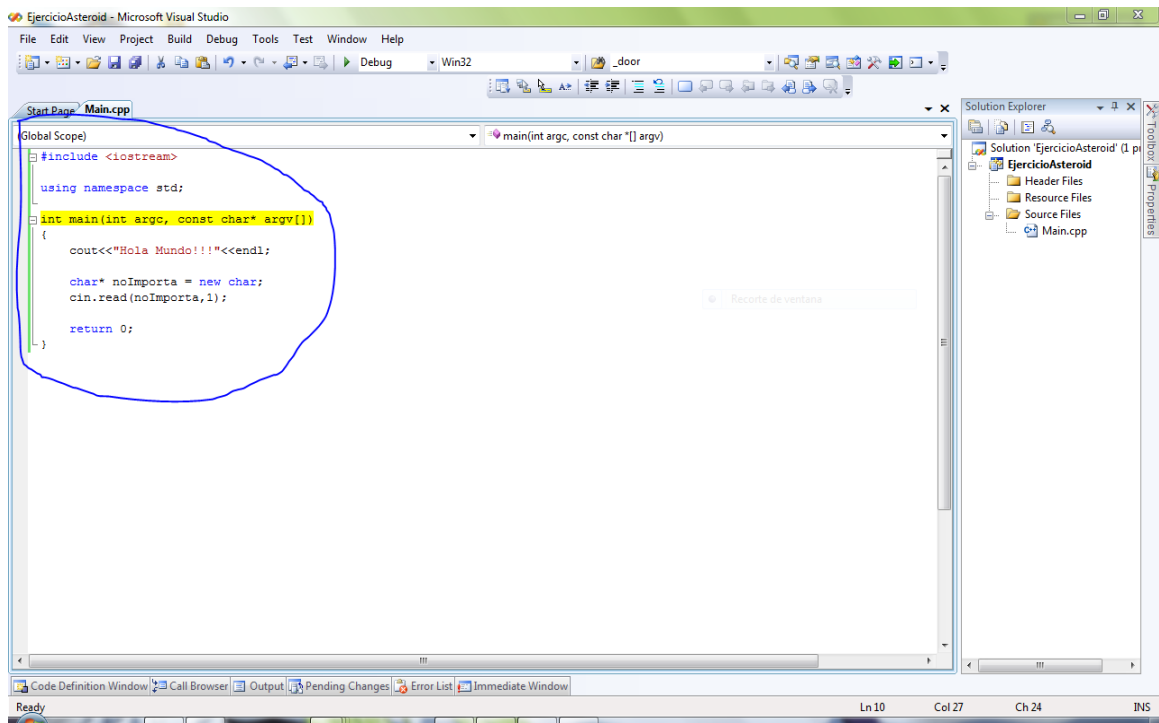
Dialogo de selección de tipo de archivo

En la captura vemos que tendremos el tab de Main.cpp abierto y vacío.

Entonces escribiremos en este la implementación de la función Main con la definición que ya conocemos:



El archivo Main vacío



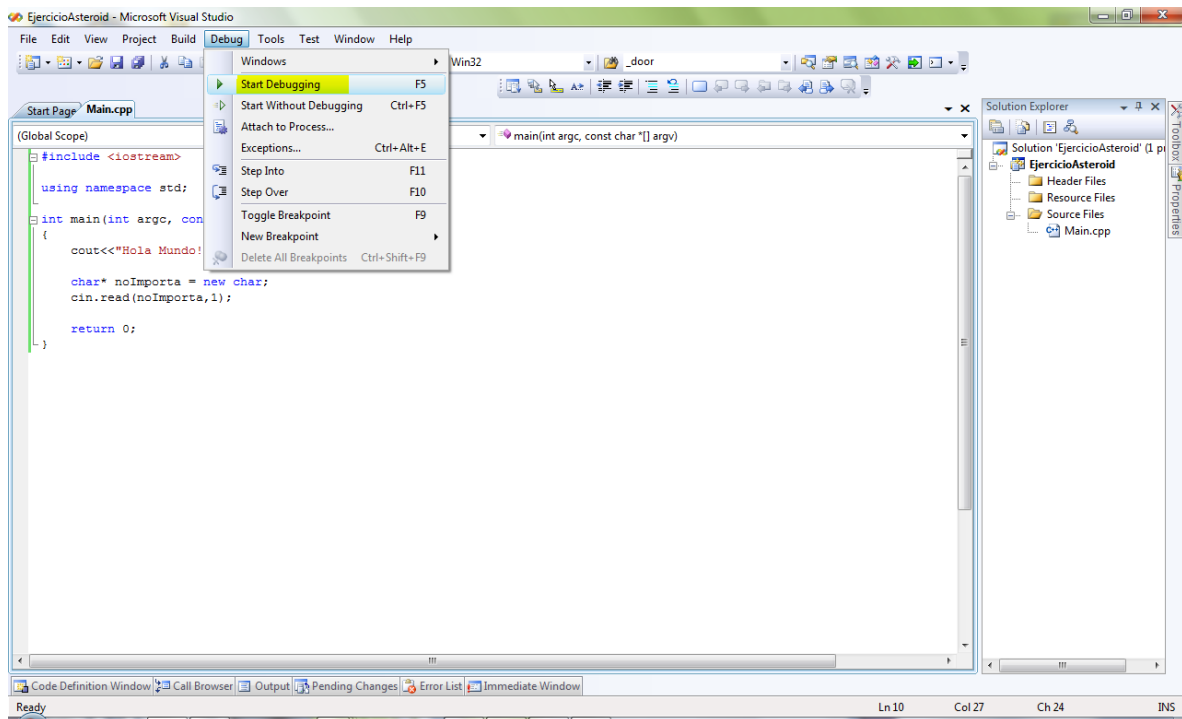
Implementación de nuestro Main

Probando el código:

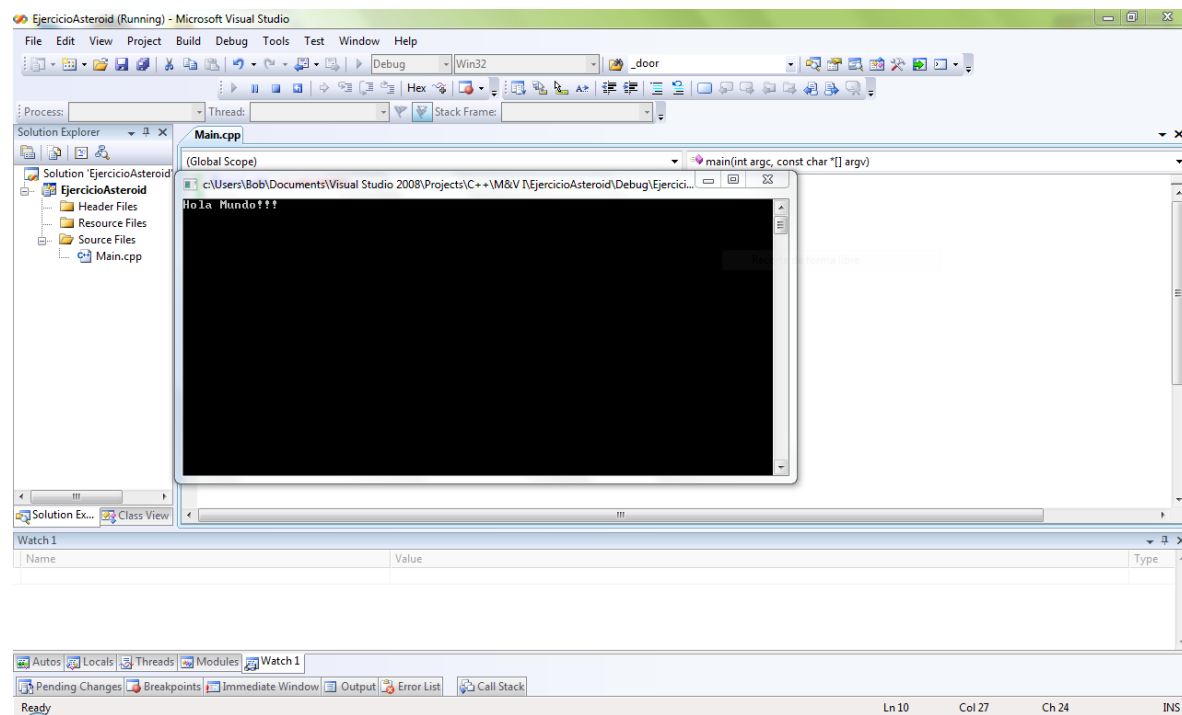
Una práctica fundamental en la programación es la de ir probando el código a medida que avanzamos. Si escribimos mucho sin probarlo lo más probable es que terminemos en un callejón sin salida, con 300 errores (y esto no es necesariamente una exageración) o el programa corriendo y explotando de una forma mística. Para evitarnos todos esos dolores de cabeza y tics nerviosos que pueden dejarnos de forma permanente lo más sano es ir probando el código.

Para hacer eso lo que debemos hacer es:





Una vez que haya sido ejecutado el .exe veremos el mensaje de hola mundo.

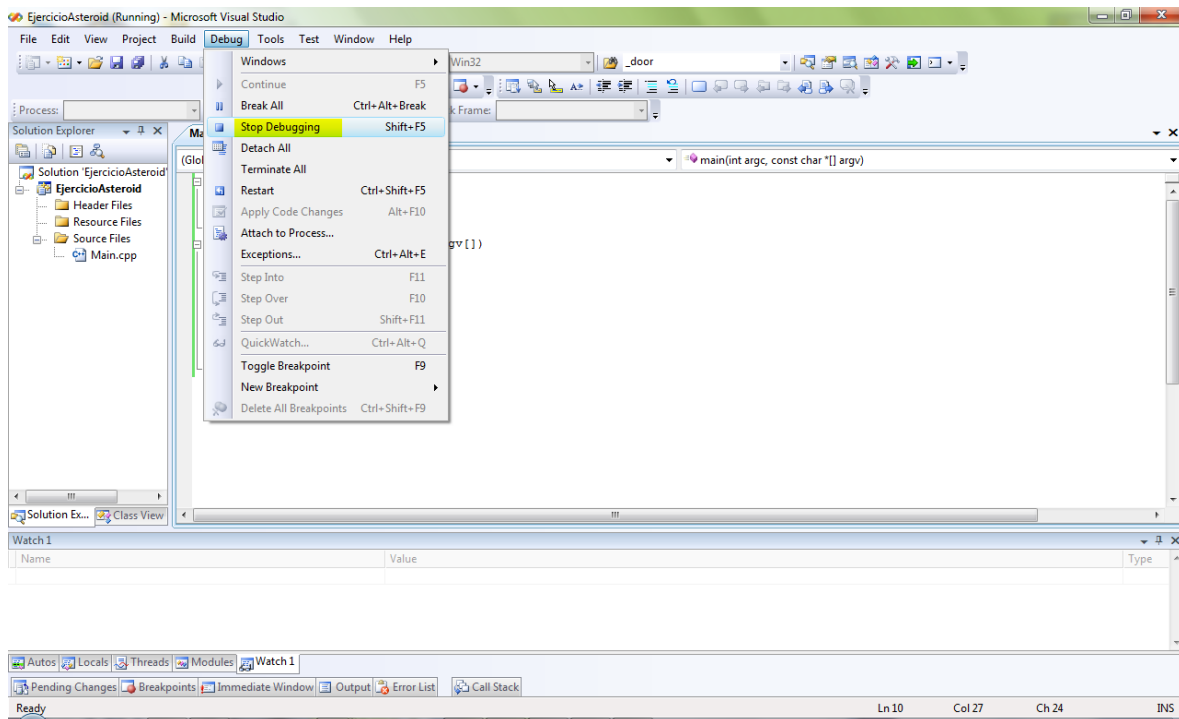


Hola Mundo!!!

Para terminar la aplicación, en nuestro caso particular programamos para que termine una vez que el usuario presione una tecla.

Si queremos terminar de forma arbitraria o terminar un ciclo infinito u otro estado erróneo, lo hacemos desde:

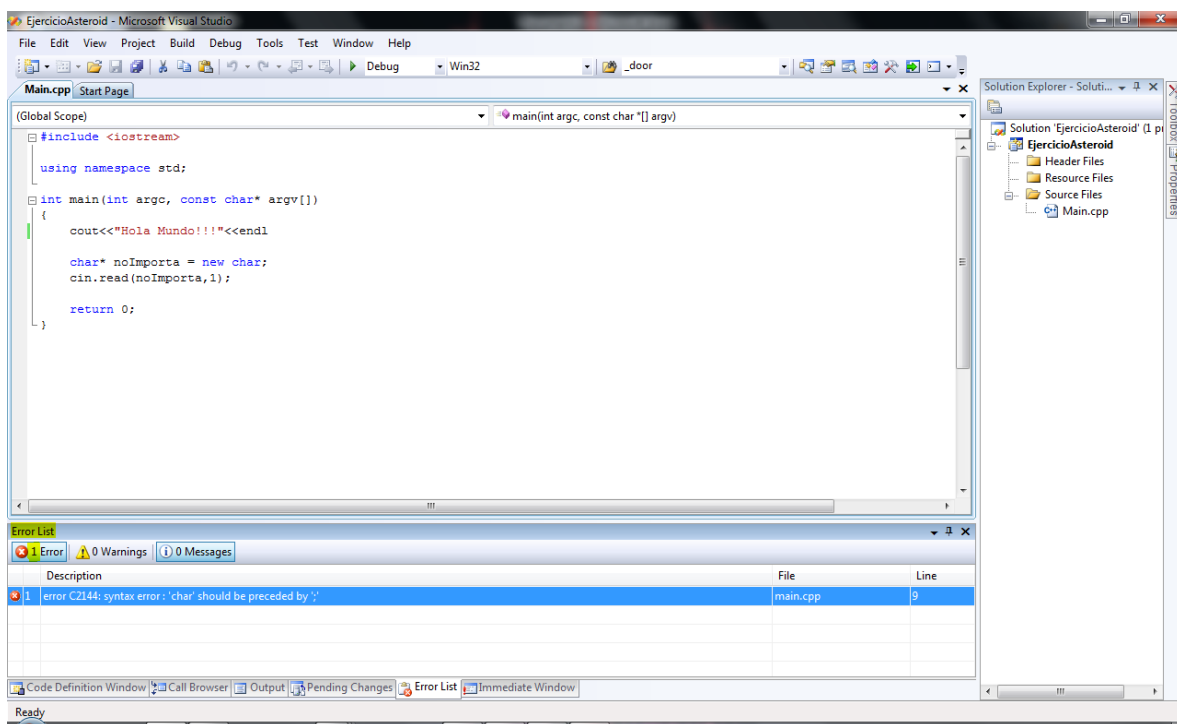
Depurar > Detener Depuración



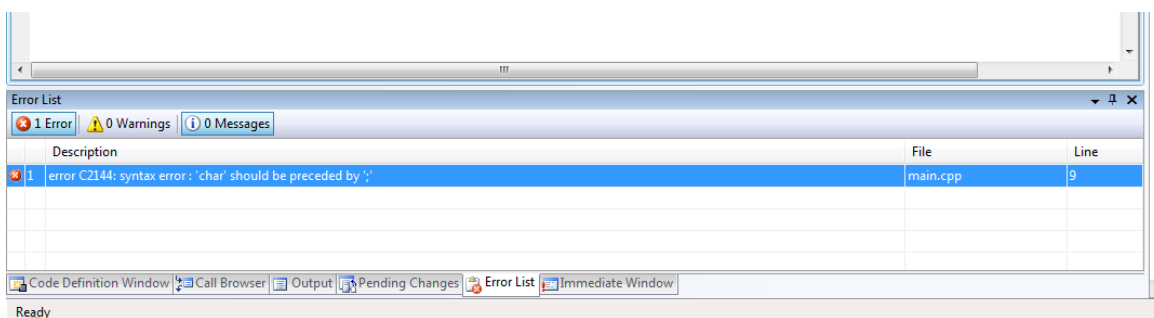
Si durante la compilación o el linkeado se produjeron errores, podremos ver un detalle de los mismos en la ventana de Lista de Errores, en general estará en la sección inferior de la pantalla.

Ahí nos dirá como vemos en la captura la cantidad de errores, los cuales no necesariamente es representativa del problema, es decir un pequeño detalle como no poner un ; al final de una definición de clase puede producir una retahíla de errores.

Además nos dirá en que archivo se produjo y en qué línea de código. De forma más sucinta siempre podemos hacerle doble click y nos llevara automáticamente a donde se produjo el error, esto siempre y cuando sea un error en la compilación, ya que un error en el linkeo puede ser del tipo: "No se encuentra la librería XXX.lib" (aunque en ingles claro) o un error similar donde línea y archivo no tienen sentido.



Vemos en la captura que el error se produjo porque al final de la línea de **cout** no pusimos ;



El detalle nos dará una pista de donde está el problema, aunque en general puede ser muy vaga.

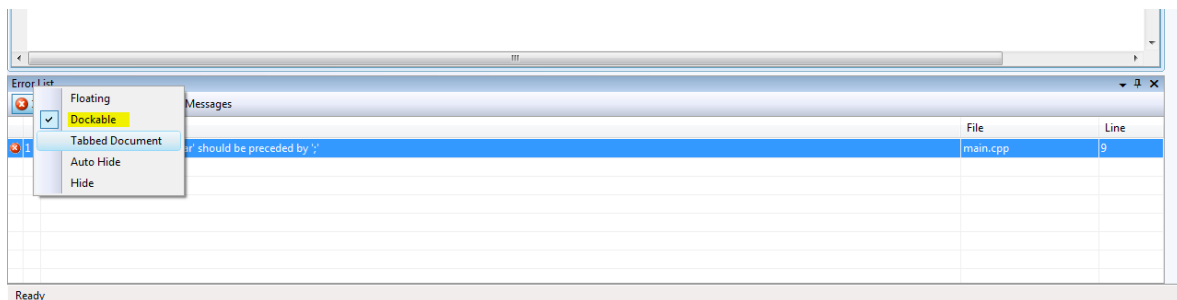
Manejo de las ventanas:

Visual Studio tiene un sistema de interfaz grafica bastante personalizable. Cualquier ventana (como Explorador de Soluciones, Lista de Errores, Output, etc.) puede estar en 3 estados:

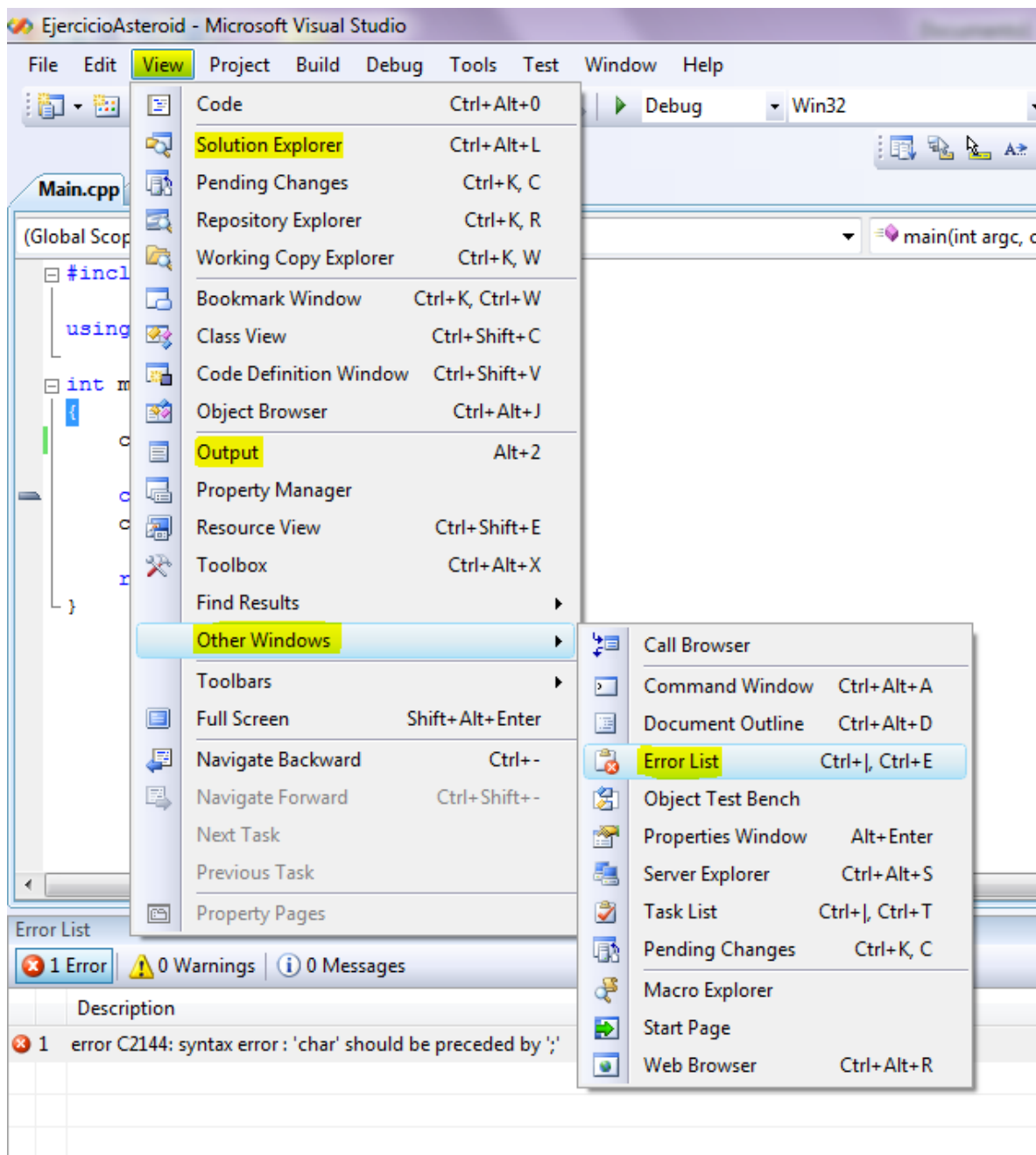
1. Flotante
2. Dockable
 - a. Siempre visible
 - b. Oculta automáticamente
3. En tab
4. Oculta

Si esta como flotante simplemente podremos ponerla en cualquier lado sin estar fijada a ningún costado. Si esta en dockable (en general lo más normal) se la puede fijar a cualquier costado: la parte inferior, superior, izquierda o derecha. En este estado puede decirse a la ventana que se auto-oculte cuando no se está usando. En tab ubica la ventana en la parte superior de tabs de la misma forma que se ubican los archivos .cpp , .h y demás. Y de estar oculta, como es de esperarse, simplemente no se verá y si se la desea se la deberá re activar del menú Ver.

Para cambiar el estado de una ventana solo se le debe hacer click derecho sobre la barra de nombre de la ventana y elegir la opción deseada. En la captura siguiente vemos el menú de selección del estado de la ventana:



Si cerramos o simplemente ya estaba en estado oculta una ventana podemos volver a abrirla desde el menú **Ver** como muestra la captura siguiente:



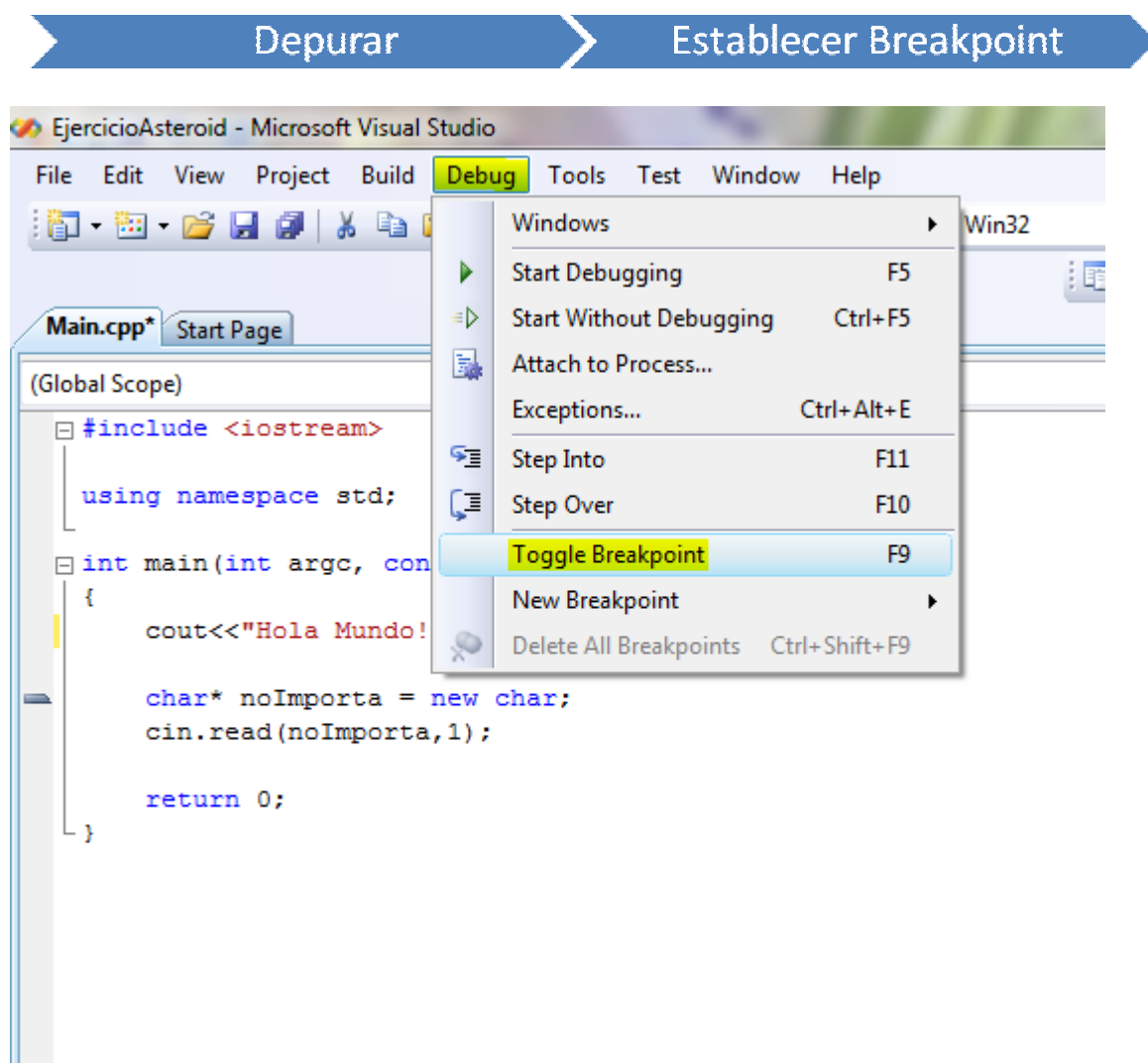
En la captura se ve remarcadas las ventanas de:

- **Explorador de Solución:** Ya se hablo de este
- **Output:** da información sobre los procesos de compilación y linkeo. Si tenemos un problema en la compilación esta ventana al igual que la Lista de Errores nos puede servir. Además también es posible con una llamada a una función de Visual Studio hacer salida a esta ventana.
- **Lista de Errores:** ya se hablo de esta, pero se podría decir que en general es la que mas usaremos.

Breakpoints:

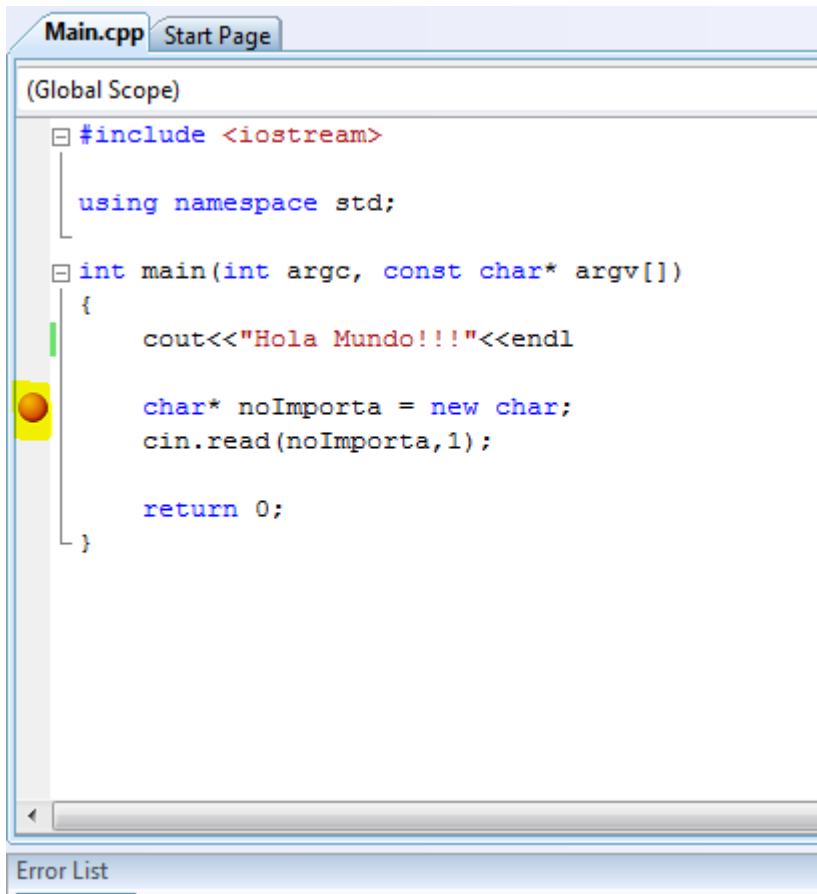
Para probar porque un programa no anda o porque anda (a veces nos sorprenden para bien :D) una técnica de depuración consiste en usar break points. Estos son puntos ubicamos en líneas específicas de código, y al llegar la ejecución del programa a esa línea se para y nos permitirá seguir el programa durante su ejecución línea por línea. Durante esto podremos ver los valores de las variables en ese instante y los estados de los objetos.

Para insertar un break point debemos:



Alternativamente se puede activar presionando F9

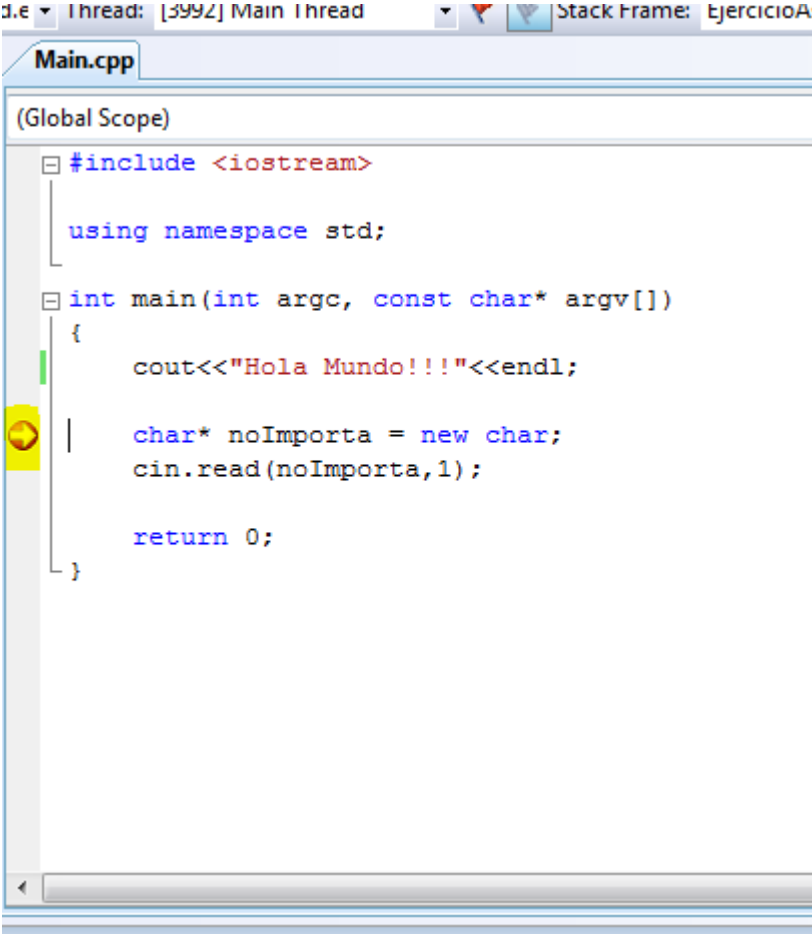
Una vez que se halla insertado el breakpoint veremos un círculo rojo en el lado izquierdo del editor, en la línea que teníamos el cursor cuando presionamos el comando.



De esta forma cuando la ejecución llegue a este punto la ejecución se detendrá y podremos ver estados de las variables y seguir la ejecución línea a línea, para de esta forma detectar errores.

Si corremos el programa desde Depurar -> Iniciar Depuración, como vimos anteriormente, veremos que el programa se para en esa línea de código. Vemos que en el círculo rojo hay una flecha, esta indica en que línea esta parada la ejecución.

En este estado con el mouse podemos posicionarlo sobre nombres de variables y podremos ver qué valor tienen. Por ejemplo pruebe a posicionar el mouse sobre la variable **argc**, de esta forma aparecera una ventana que dira su valor(seguramente 1).



The screenshot shows a debugger window with the following details:

- Thread: [3992] Main Thread
- Stack Frame: EjercicioA
- File: Main.cpp
- Scope: (Global Scope)
- Code:

```
#include <iostream>

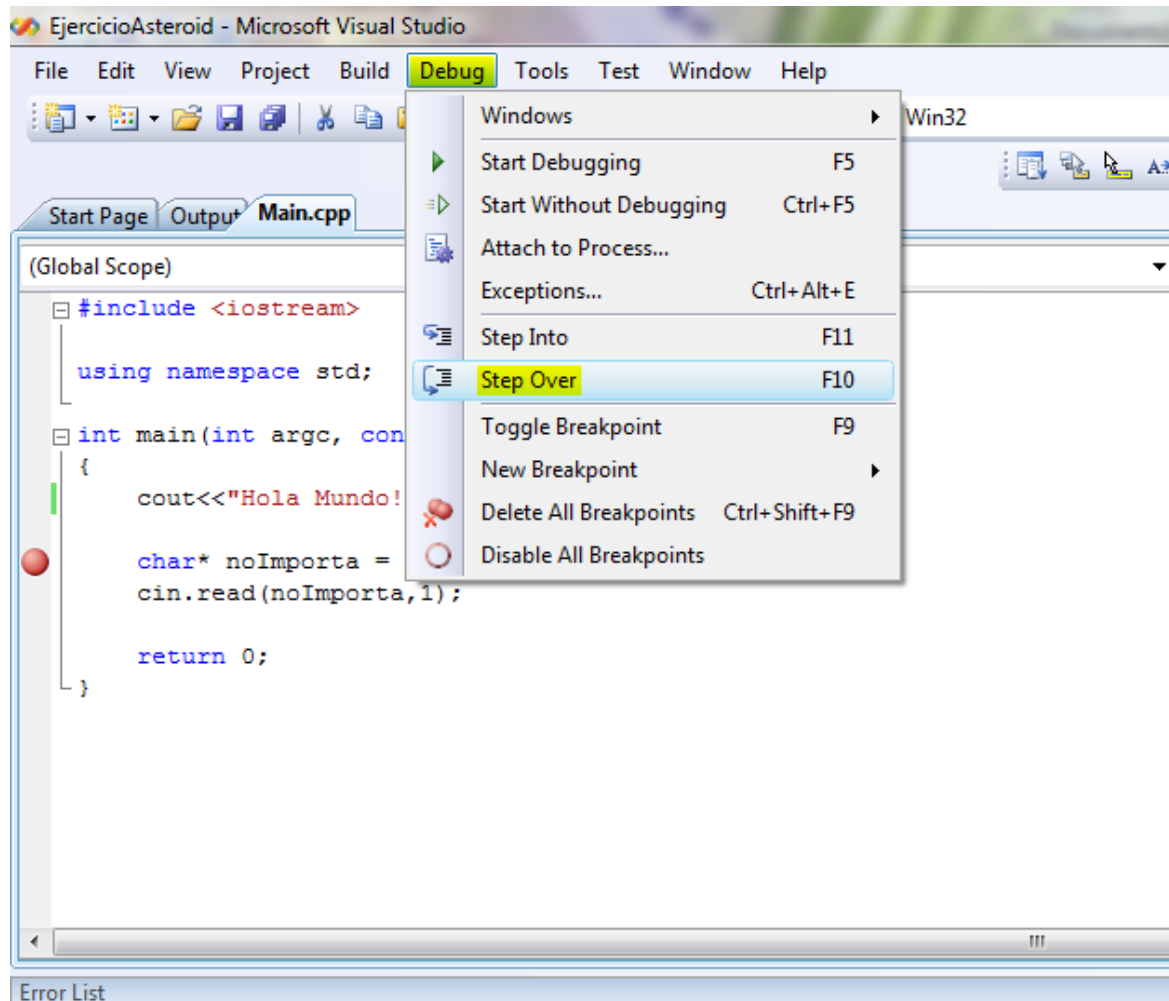
using namespace std;

int main(int argc, const char* argv[])
{
    cout<<"Hola Mundo!!!"<<endl;
    char* noImporta = new char;
    cin.read(noImporta,1);

    return 0;
}
```

A yellow circle with a red arrow points to the line `cin.read(noImporta,1);`, indicating the current execution point. A vertical green bar is on the left margin of the code editor.

Luego si queremos que la ejecución pase a la siguiente línea hacemos:



Alternativamente podemos presionar F10