TEMA 01

NOCIONES BÁSICAS

1.1.- Introducción.

El lenguaje C fue inventado por Dennis Ritchie en 1972 en el proceso de diseño del sistema operativo UNIX. Deriva del lenguaje B de Ken Thompson quien cooperó también en el diseño de la primera versión (UNIX v.5). Más tarde surgieron gran cantidad de nuevas implementaciones y se hizo necesario un estándar, el ANSI C. La implementación que usaremos es la de Visual C ++ 2008 Express Edition.

Algunas características del C son:

- Es un lenguaje de propósito general porque no está enfocado a hacer un tipo determinado de programas, con él se puede crear desde sistemas operativos hasta los más avanzados sistemas expertos.
- Es de medio nivel porque en él se puede trabajar a alto nivel (nivel lógico, más fácil) y a bajo nivel (nivel físico del ordenador, más rápido).
- Su código es fácilmente portable a otros sistemas.
- Es un lenguaje potente y eficiente: Usando C, un programador puede casi alcanzar la eficiencia del código ensamblador junto con la estructura del Algol o Pascal.

Para crear un programa en C se deben seguir los siguientes pasos:

1. Escribirlo en uno o más ficheros de texto de una forma reconocible por el compilador (se escriben con extensión .C o .CPP). Son los ficheros de código fuente.

2. Compilarlo. Los ficheros fuente se compilan creando ficheros de código objeto o código máquina con significado para el microprocesador (se escriben con extensión .OBJ).

Este código objeto no está completo, le falta una serie de rutinas que están en las librerías (.LIB) y a las que por ahora solo hace referencia.

3. Enlazarlo. Se unen todos los programas (.OBJ y .LIB) mediante el enlazador (linker) que produce un fichero ejecutable (.EXE, .COM) por el sistema operativo (por ejemplo tecleando nosotros su nombre en la línea de comandos).

4. Depurarlo. Se comprueba si hay errores.

5. Ejecutarlo. Si funciona correctamente, como nosotros queremos, el programa está acabado. En caso contrario habrá que corregirlo y volver a repetir todos los pasos anteriores.

El entorno de desarrollo integrado de Visual C++ proporciona facilidades para administrar cada estado del programa, desde la creación del código fuente a la construcción (compilado y enlazado) del código, hasta probar, depurar y optimizar el programa.

1.2.- Mi primer programa en C.

Todos los programas deben incluir las librerías, la función principal, abrir llave y cerrar llave.

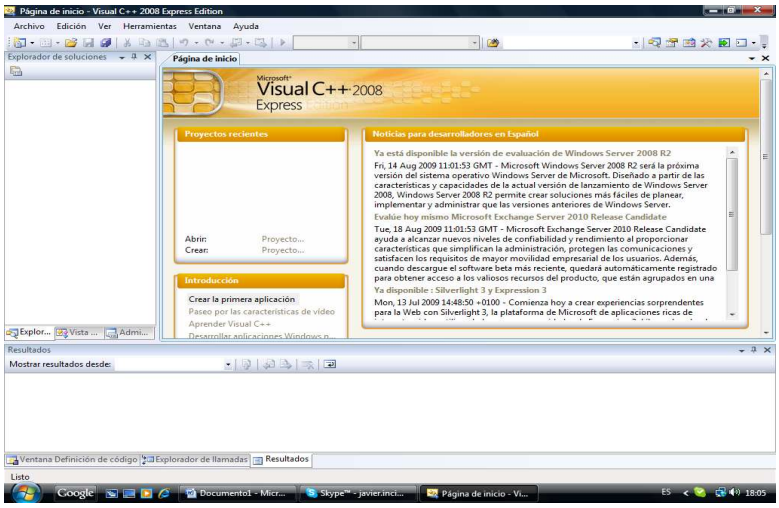
En programación, la experiencia es la gran maestra. Por ello, es conveniente hacer programas en C cuanto antes. Por ejemplo mi primer programa en C puede ser el siguiente:

Programa 001

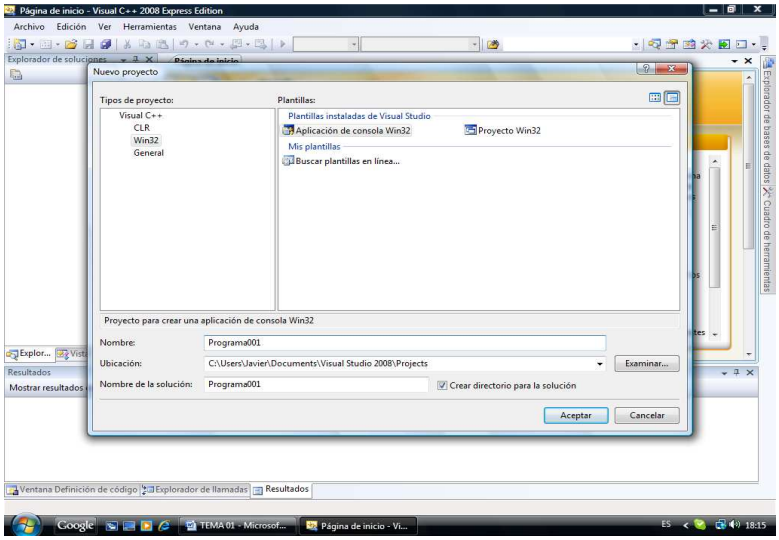
```
1ª línea    /* Programa 001 */
2ª línea    //  Programa que escribe una frase
3ª línea    # include <stdio.h>
4ª línea    void main ( )
5ª línea    {
6ª línea          printf( “Mi primer programa en C.” );
7ª línea    }
```

Este proyecto se llamará **Programa 001** y su misión es sacar por pantalla la frase: **Mi primer programa en C**. Para crear un nuevo proyecto y agregar un archivo de código fuente seguiremos los siguientes pasos.

1. Si hemos creado un icono de acceso directo en el escritorio hacemos doble clic en el botón izquierdo del ratón.
2. Si no hemos creado el icono de acceso directo pulsamos en Inicio, Todos los programas, Microsoft Visual C++ 2008 Express Edition, Microsoft Visual C++ 2008 Express Edition, haciendo un clic en el botón izquierdo del ratón aparece la siguiente imagen la primera vez y en sucesivas veces saldra una lista con los proyectos recientes.



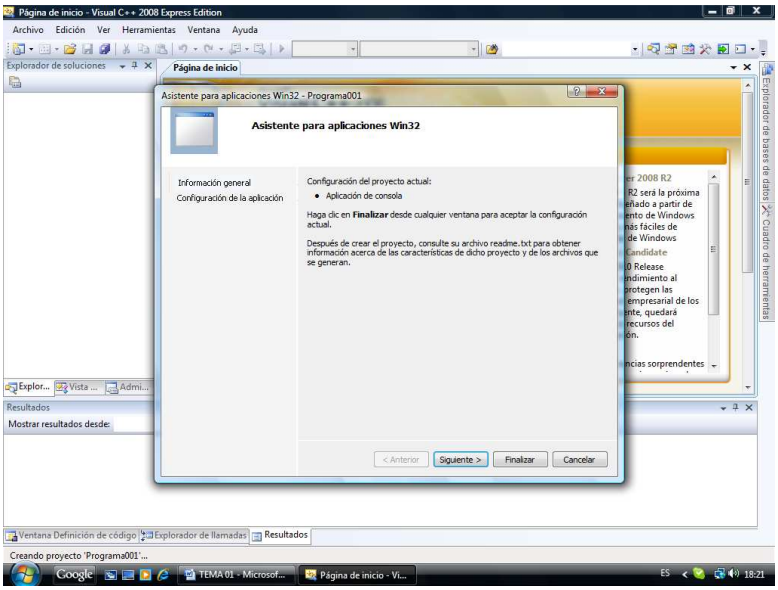
3. En el menú **Archivo**, seleccionamos **Nuevo** y, a continuación, hacemos clic en **Proyecto**.
4. En los tipos de proyecto **Visual C++**, hacemos clic en **Win32** y, a continuación, en plantillas instaladas de Visual Studio hacemos clic en **Aplicación de consola Win32**.



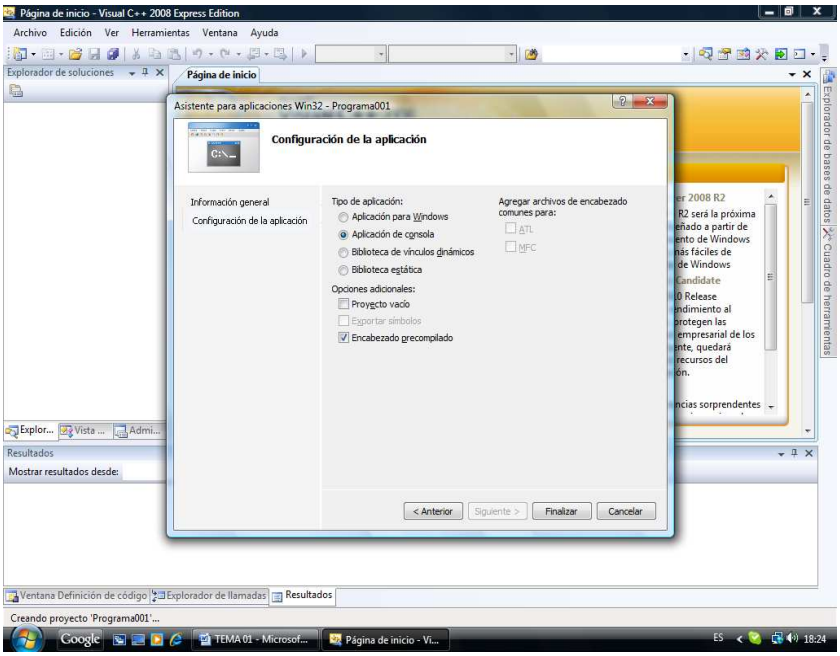
5. En Nombre escribimos el nombre del proyecto. En nuestro caso **Programa001**.

   De forma predeterminada, la solución que contiene el proyecto tiene el mismo nombre que el nuevo proyecto, pero podemos escribir un nombre diferente. **Podemos**, también, **escribir una ubicación diferente para el proyecto**.

   Hacemos clic en **Aceptar** para crear el nuevo proyecto y aparece la siguiente pantalla.



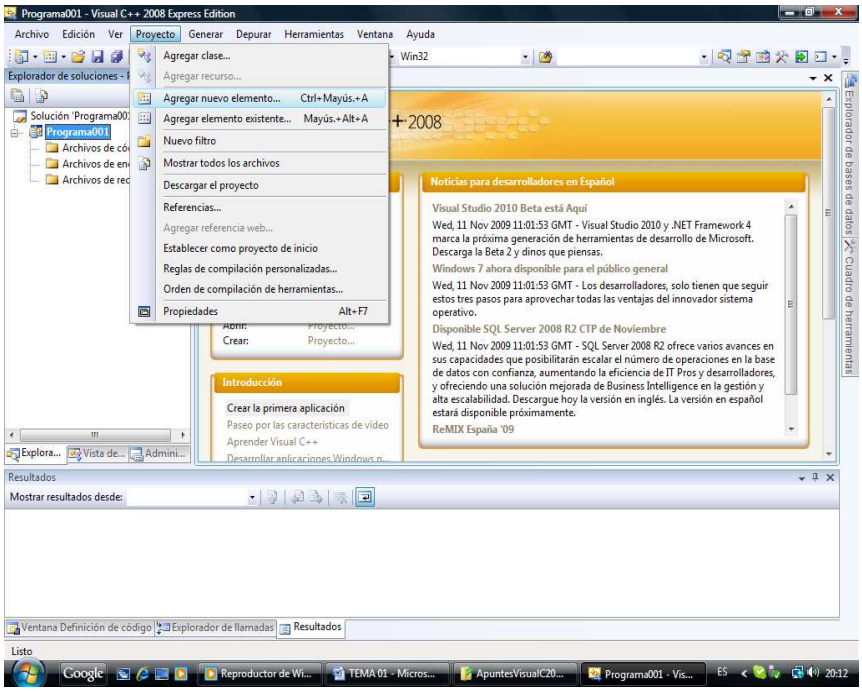
6. Hacemos clic en **Siguiente** o en **Configuracion de la aplicación** viendo la siguiente pantalla.



7. En el Asistente para aplicaciones Win32 en el apartado Tipo de aplicación debemos marcar: **Aplicación de consola** y en Opciones adicionales: **Proyecto vacío**. A continuación hacemos clic en **Finalizar**.

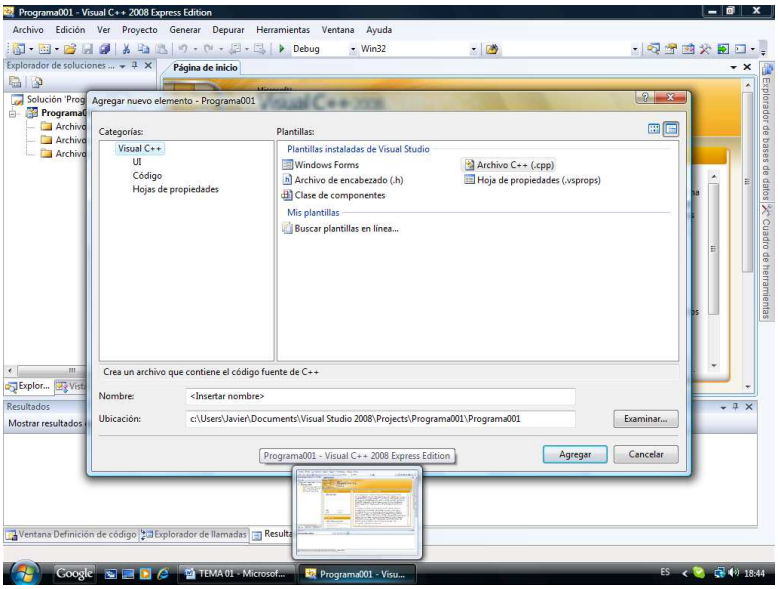
8. En la siguiente pantalla, si el Explorador de soluciones no está visible, hacemos clic en **Explorador de soluciones** en el menú **Ver**.

9. Para agregar un nuevo archivo de código fuente y teniendo seleccionado Programa001 en el Explorador de soluciones hacemos clik en **Proyecto** y después **Agregar Nuevo Elemento.**



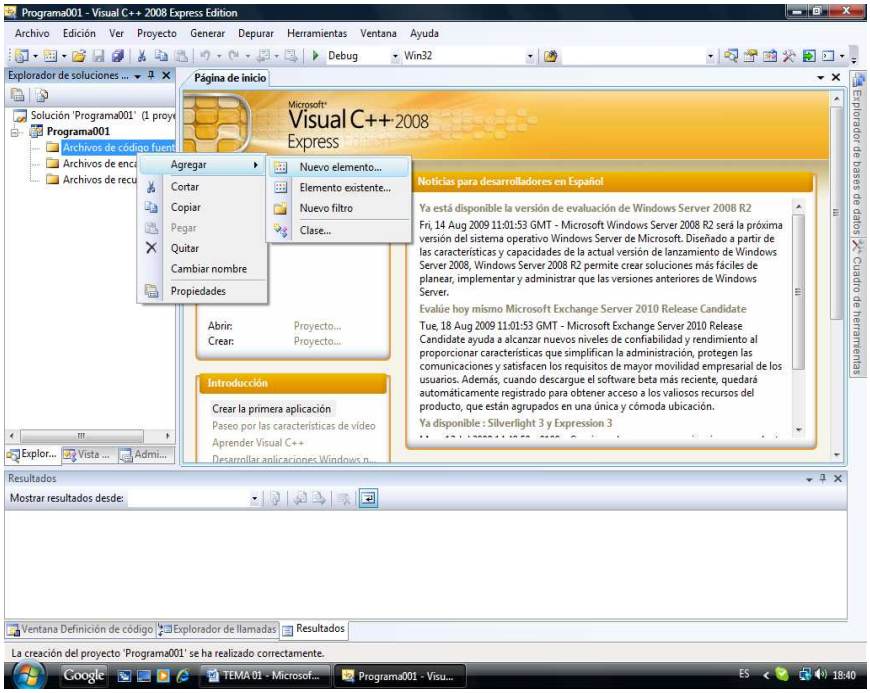
También se puede hacer con el ratón **botón derecho**, **Agregar**, **Nuevo Elemento**.

10. En la siguiente pantalla, en categorías marcamos **Visual C++ Código.**

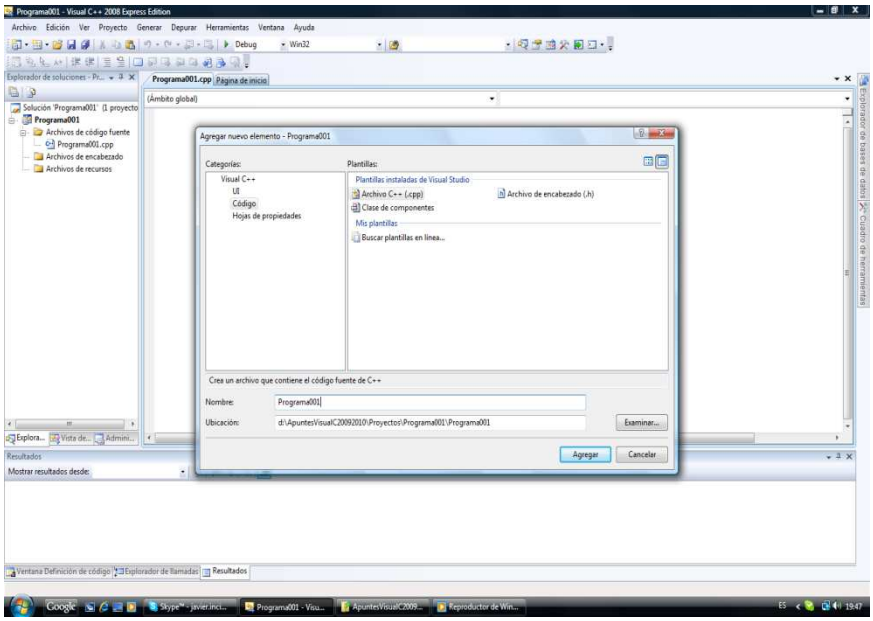


En Plantillas, Plantillas instaladas de Visual Studio, hacemos clic en **Archivo C++ (.cpp)**, escribiendo un nombre de archivo (lo llamamos igual que el proyecto, Programa001, aunque podemos variarlo) y a continuación, hacemos clic en **Agregar**.

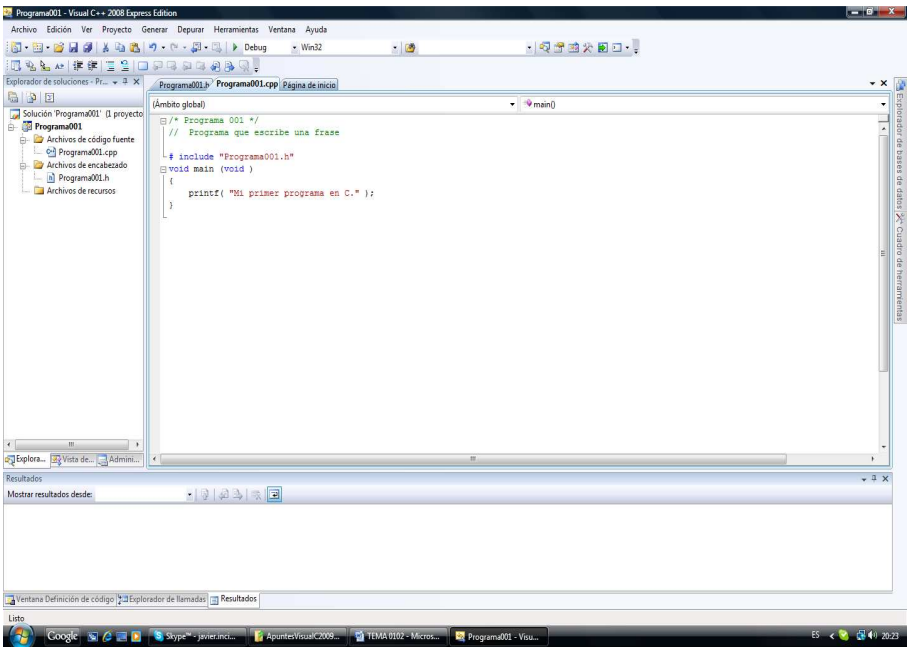
Antes de escribir el código debemos agregar un Archivo de Encabezado. Para ello y teniendo seleccionado **Programa001** en el Explorador de soluciones hacemos clik en botón derecho del ratón y en el menú que se despliega seleccionamoes **Agregar, Nuevo elemento**.



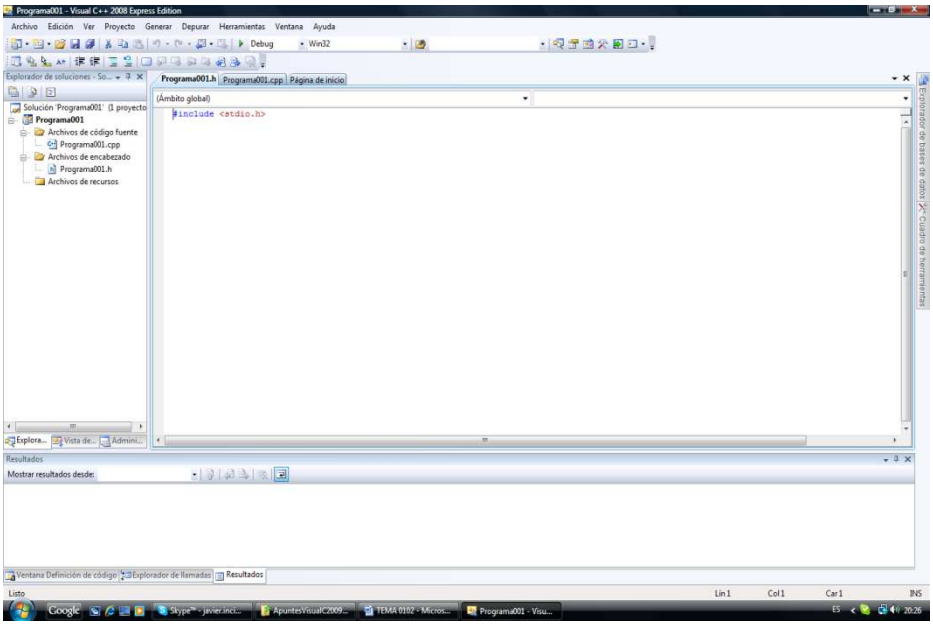
Aparece la siguiente pantalla y en **Plantillas, Plantillas instaladas en Visual Studio**, seleccionamos **Archivo de encabezado (.h).** En **nombre** ponemos **Programa001**.



Es momento de escribir el código del programa. El archivo Programa001.cpp que hemos creado se muesta en la carpeta Archivos de código fuente en el Explorador de soluciones, apareciendo en la derecha una ventana donde escribiremos dicho código.



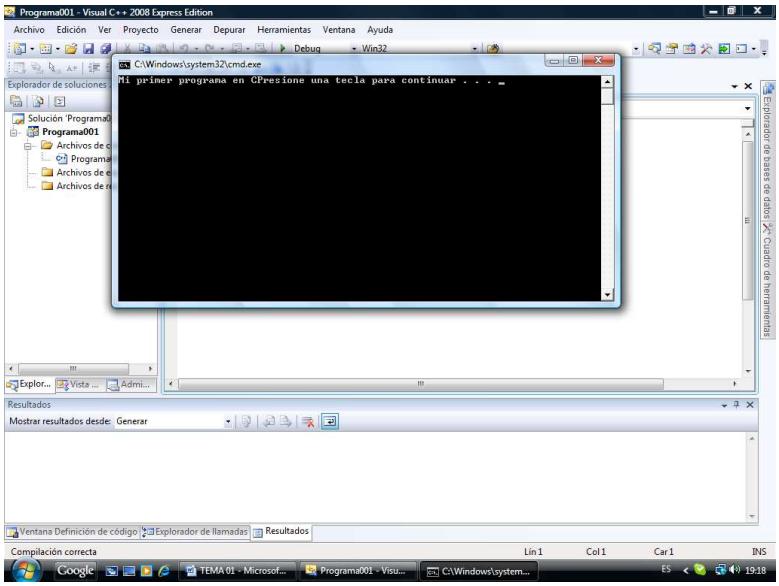
El archivo Programa001.h incluimos las bibliotecas, en este caso **stdio.h.**



11. En el menú **Generar**, hacemos clic en **Generar** solución. También desde el teclado con **F7**.

    Vemos en **Resultados** (ventana de abajo) muestra información sobre el progreso de la compilación y si hay algún fallo de escritura nos lo indicará (errores y advertencias). Si está correcto en Generar nos dirá 1 correcto 0 incorrectos.

12. En el menu **Depurar**, hacemos clic en **Iniciar sin depurar**.



Nota.- Si en el menu **Depurar**, hacemos clic en **Iniciar depuración** nos daría un "pantallazo" y no veríamos nada. Para que esto no ocurra debemos incluir una nueva línea con la función **getchar( )** que está incluida en la librería **stdio.h** (Librería de C)**.**

Analizamos el programa, teniendo en cuenta, qué el programa está escrito en minúsculas.

1ª línea — Mediante /* …… */ podemos insertar comentarios en el programa (Los comentarios acaban cuando escribo */).

2ª línea — Hace lo mismo que la primera línea (pero solo puedo escribir una línea).

3ª línea — Contiene una referencia a un archivo especial (stdio.h) que contiene informaciónque se debe incluir en el programa cuando se compila. Está información la maneja automáticamente el compilador.

4ª línea — Los programas en C están compuestos de unidades de programa llamadas funciones, las cuales son los módulos básicos del programa. En este caso, el programa está compuesto por una sola función llamada ***main.***

Todos los programas en C deben tener la función **main** (significa en español **principal**) pues la primer función que se ejecuta cuando hacemos funcionar el programa.

Los paréntesis que siguen a main la identifican como nombre de función.

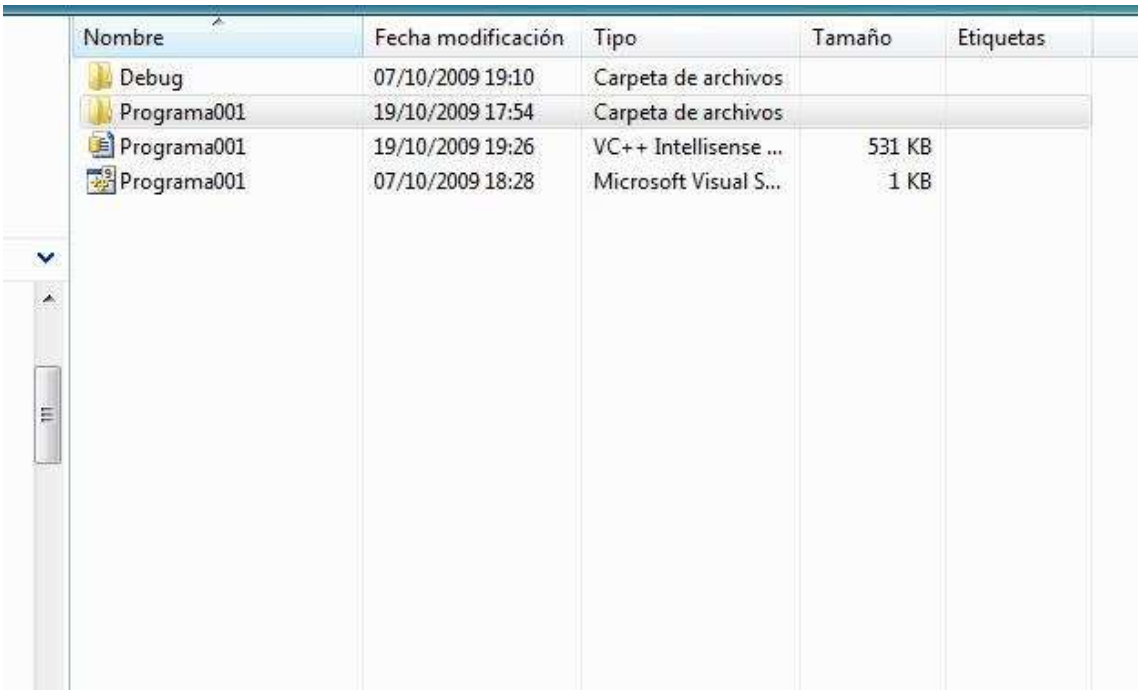
Un método de comunicación de datos entre las funciones es el uso de los **argumentos**. Los argumentos son los datos que se les pasa a las funciones. Estos se encierran entre paréntesis. En Visual C++ es necesario que devuelva argumentos (nos vale **void** delante de main).

5ª línea — Apertura de llave. Comienza el programa.

6ª línea — Realiza una llamada a una función denominada **printf**, con el argumento "Mi primer programa en C"; printf es una función de biblioteca que realiza una escritura en la salida estándar (normalmente la salida estándar es el monitor). La función printf escribe una cadena de caracteres (string). Cada instrucción en C acaba con punto y coma (;).

7ª — Cierre de llave. Finaliza el programa.

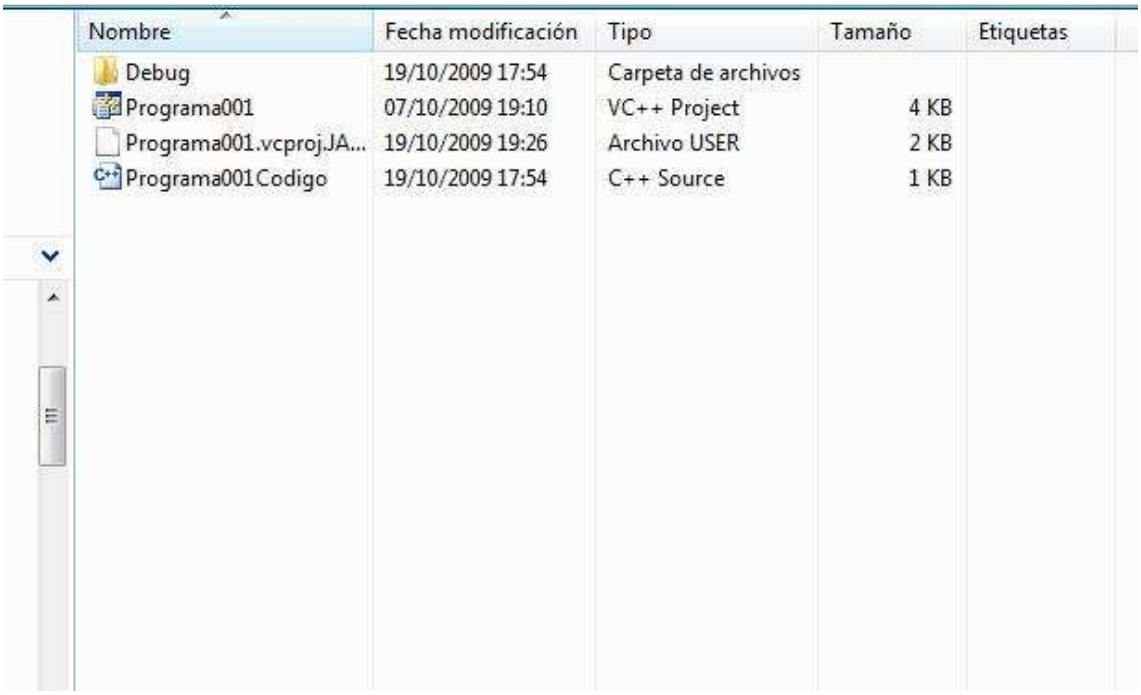
Donde hemos guardado el programa Programa001, Visual C 2008 ha generado varios archivos y carpetas.



Nombre	Fecha modificación	Tipo	Tamaño	Etiquetas
Debug	07/10/2009 19:10	Carpeta de archivos		
Programa001	19/10/2009 17:54	Carpeta de archivos		
Programa001	19/10/2009 19:26	VC++ Intellisense ...	531 KB	
Programa001	07/10/2009 18:28	Microsoft Visual S...	1 KB	

Dentro de la carpeta Programa001 nos interesa el programa llamado Programa001Codigo. En este archivo se encuentra escrito el código de nuestro programa.

Dentro de la carpeta Debug se encuentra el programa Programa001 (Tipo Aplicación) desde donde se puede ejecutar el programa.



Nombre	Fecha modificación	Tipo	Tamaño	Etiquetas
Debug	19/10/2009 17:54	Carpeta de archivos		
Programa001	07/10/2009 19:10	VC++ Project	4 KB	
Programa001.vcproj.JA...	19/10/2009 19:26	Archivo USER	2 KB	
Programa001Codigo	19/10/2009 17:54	C++ Source	1 KB	

Vamos a por un nuevo proyecto que llamaremos Programa002. Este programa me escribe dos frases, una debajo de la otra y haremos que la información de finalización del programa no aparezca inmediatamente.

Programa 002

```
1ª línea    /* Programa 002 */
2ª línea    //  Programa que escribe dos frases
3ª línea    # include <stdio.h>
4ª línea    void main ( )
5ª línea    {
6ª línea          printf( “Mi primer programa en C.\n” );
7ª línea          printf(“Me llamo Javier”);
8ª línea          getchar( );
9ª línea    }
```

Hay dos novedades respecto al primer ejemplo:

Primera: Aparece en la línea 6ª el código \n dentro de la cadena del primer printf. Hay un cierto número de caracteres no imprimibles. Para usarlos debemos representarlos mediante una combinación de secuencias de escape, estas son:

Código	Significado
\b	Retroceso, lo mismo que la tecla borrar
\f	Salto de página
\n	Nueva línea
\r	Retorno de carro, el curso al principio de la línea actual
\t	Tabulación horizontal
%%	Tanto por ciento %
\”	Comillas “
\´	Apóstrofe ‘
\0	Carácter nulo, suele delimitar las cadenas de caracteres
\\	Barra invertida \
\v	Tabulación vertical
\a	Alerta (bell, campanilla)
\ddd	Constante octal (ddd son tres dígitos como máximo)
\xddd	Constante hexadecimal (ddd son tres dígitos como máximo)

Segunda: Aparece una nueva función llamada **getchar**.

Es una función que espera que se pulse la tecla RETURN por parte del usuario. Esta función no necesita argumentos pero los paréntesis son necesarios ya que se trata de una función. Se encuentra en la librería **stdio.h**.

En algunas versiones de C, cuando ejecutamos un nuevo programa, vemos "los restos" del anterior programa ejecutado. Por ello es conveniente limpiar la pantalla. Existe, en C++, una función que nos realiza esta limpieza. Se trata de la función **clrscr( )** que se encuentra en la librería **conio.h** (Si ejecutamos el programa sin incluir la librería conio.h cuando compilamos nos da un error diciendonos que "que la función clrscr debería tener un prototipo", osea que le falta la librería para ir a buscar dicha función). En Visual C 2008 nos dice que "no se encontro el identificador" ya que no reconoce dicha función (Para poder limpiar la pantalla llamaremos al sistema y ejecutando la orden cls, escribiendo **system("cls")** e incluyedo la librería **windows.h** lo conseguiremos).

Hemos de observar, para poder diferenciar, que ocurre cuando depuramos mediante iniciar depuración e iniciar sin depurar.

1.3.- Elementos básicos de un programa de C.

Los programas funcionan con datos que contienen información. Estos datos se pueden dividir en constantes (valores fijos no alterables) y variables (valores que pueden cambiar).

Antes de utilizar un datos debemos declararlo dándole un nombre que utilizaremos después para referenciarlo. En la declaración avisamos al compilador de la existencia de una variable, del nombre de la misma y si queremos de su valor.

La longitud del nombre, identificador, depende del compilador, pero la mayoría considera los treinta y dos caracteres primeros. El nombre nunca debe comenzaar por un número o por ningún carácter que no sea una letra del alfabeto o el carácter subrayado. Solo son permitidas las letras, los números y el carácter subrayado. Un identificador siempre debe contener al menos una letra.

Ejemplos:

Numero	Válido
2Numero	No válido, comienza por dígito
Numero2	Válido
_Numero	Válido
_Numero!	No válido. Contiene un carácter no válido
#Numero	No válido. Comieza # y contiene un carácter no válido

Existen unas palabras reservadas que no pueden usarse como identificadores, poque el compilador de C las interpreta única y exclusivamente de una manera determinada. Estas palabras son:

auto	double	int	struct	break	else
long	switch	case	enum	register	type
def	char	extern	return	union	const
float	short	unsigned	continue	for	signed
void	default	goto	sizeof	volatile	do
if	static	while			

Vamos a continuar escribiendo, ejecutando y observando el siguiente programa:

```
/* Programa 003 */

#include <stdio.h>

void main()

{
      int horas,minutos;
      horas=3;
      minutos=60*horas;
      printf("Hay %d minutos en %d horas.\n",minutos,horas);
}
```

NOTA.-

En el apéndice A tenemos las bibliotecas de C brevemente comentadas.

Debemos ver las bibliotecas que se usan en ANSI C a través de internet.

www.conclase.net/c/librerias/index.php

Debemos ver las bibliotecas que se usan en C++

www.zator.com/Cpp/E5.htm

En el mismo programa de Visual podemos ver las bibliotecas que se usan.