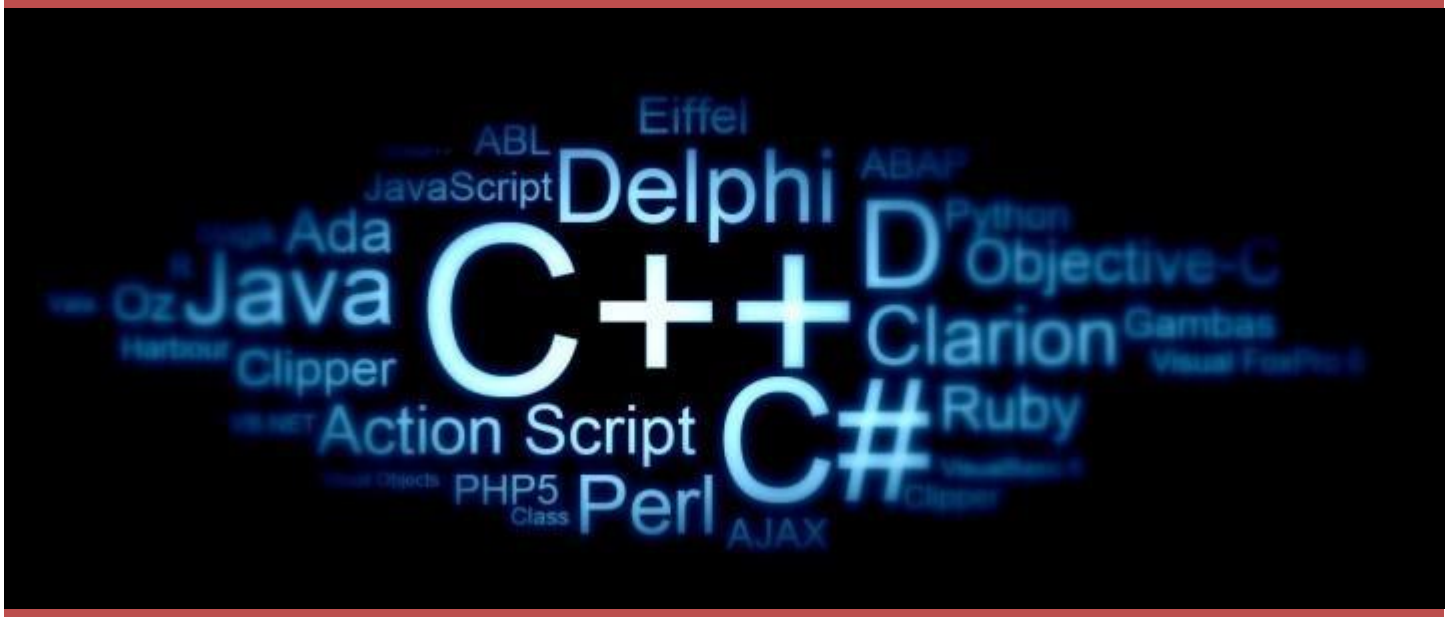
ESBA

BARRIO NORTE

HERRAMIENTAS DE PROGRAMACION

LIC. Eduardo Shimoyama



Unidad I: Introducción e historia de la programación.

Tema 1 – Introducción a la programación.

a. Programas
b. Lenguajes
c. Compiladores
d. Interpretes

Tema 2 – Historia del Lenguaje C

a. Comienzo del Lenguaje C

Unidad II: Desarrollo de la programación.

Tema 3 – Fases en el desarrollo de un programa.

Tema 4 – Elementos del lenguaje C.

Unidad I:
INTRODUCCION E HISTORIA DE LA PROGRAMACION

¿QUÉ ES UN PROGRAMA?

Probablemente alguna vez haya utilizado una Computadora para escribir un documento o para divertirse con algún juego. Recuerde que en el caso de escribir un documento, primero tuvo que poner en marcha un procesador de textos, y que si quiso divertirse con un juego, lo primero que tuvo que hacer fue poner en marcha el juego. Tanto el procesador de textos como el juego son programas de computadora.

Poner un programa en marcha es lo mismo que ejecutarlo. Cuando ejecutamos un programa, nosotros sólo vemos los resultados que produce (el procesador de textos muestra sobre la pantalla el texto que escribimos; el juego visualiza sobre la pantalla las imágenes que se van sucediendo) pero no vemos lo que hace el programa para conseguir esos resultados.

Si nosotros escribimos un programa, entonces sí que conocemos su interior y por lo tanto, sabemos cómo trabaja y por qué trabaja de esa forma. Esto es una forma muy diferente y curiosa de ver un programa, lo cual no tiene nada que ver con la experiencia adquirida en la ejecución de distintos programas.

Un programa no es nada más que una serie de instrucciones dadas al computadora en un lenguaje entendido por él, para decirle exactamente lo que queremos hacer. Si la computadora no entiende alguna instrucción lo comunicará mediante un mensaje de error.

LENGUAJES DE PROGRAMACION:

Un programa tiene que escribirse en un lenguaje entendible por la computadora.

Desde el punto de vista físico, una computadora es una máquina electrónica. Los elementos físicos (memoria, unidad de proceso, etc.) de que dispone el computadora para representar los datos son de tipo binario; esto es, cada elemento puede diferenciar dos estados (dos niveles de voltaje). Cada estado se denomina genéricamente bit y se simboliza por 0 o 1. Por lo tanto, para representar y manipular información numérica, alfabética y alfanumérica se emplean cadenas de bits. Según esto, se denomina byte a la cantidad de información empleada por un computadora para representar un carácter generalmente un byte es una cadena de ocho bits.

Así, por ejemplo, cuando un programa le dice a la computadora que visualice un mensaje sobre el monitor, o que lo imprima sobre la impresora, las instrucciones correspondientes para llevar a cabo esta acción, para que puedan ser entendibles por la computadora, tienen que estar almacenadas en la memoria como cadenas de bits. Esto hace pensar que escribir un programa utilizando ceros y unos (lenguaje máquina), llevaría mucho tiempo y con muchas posibilidades de cometer errores. Por este motivo, se desarrollaron los lenguajes ensambladores.

Un lenguaje ensamblador utiliza códigos nemotécnicos para indicarle al hardware (componentes físicos de la computadora) las operaciones que tiene que realizar. Un código nemotécnico es una palabra o abreviatura fácil de recordar que representa una tarea que debe realizar el procesador de la computadora. Por ejemplo: El código MOV le requiere a la PC que mueva alguna información desde una posición de memoria a otra. Para traducir un programa escrito en ensamblador a lenguaje máquina (código binario) se utiliza un programa llamado ensamblador que ejecutamos mediante la propia computadora. Este programa tomará como datos nuestro programa escrito en lenguaje ensamblador y dará como resultado el mismo programa pero escrito en lenguaje máquina, lenguaje que entiende el computadora.

Para traducir un programa escrito en ensamblador a lenguaje máquina (Código binario) se utiliza un programa llamado ensamblador que ejecutamos mediante el propio ordenador. Este programa tomará como datos nuestro programa escrito en lenguaje ensamblador y dará como resultado el mismo programa pero escrito en lenguaje máquina, lenguaje que entiende el ordenador.

Cada modelo de ordenador, dependiendo del procesador que utilice, tiene su propio lenguaje ensamblador. Debido a esto decimos que estos lenguajes están orientados a la máquina.

Hoy en día son más utilizados los lenguajes orientados al problema o lenguajes de alto nivel. Estos lenguajes utilizan una terminología fácilmente comprensible que se aproxima más al lenguaje humano.

Cada sentencia de un programa escrita en un lenguaje de alto nivel se traduce en general en varias instrucciones en lenguaje ensamblador.

Por ejemplo:

“ printf("hola"); “

La función printf del lenguaje C le dice al computador que visualice en el monitor la cadena de caracteres especificada. Este mismo proceso escrito en lenguaje ensamblador necesitará de varias instrucciones.

A diferencia de los lenguajes ensambladores, la utilización de lenguajes de alto nivel no requiere en absoluto del conocimiento de la estructura del procesador que utiliza el ordenador, lo que facilita la escritura de un programa.

El lenguaje C se denomina como un lenguaje de nivel medio, puesto que combina elementos de lenguajes de alto nivel (Fortran, Pascal, Basic…) con el funcionalismo del lenguaje ensamblador.

C permite la manipulación de bits, bytes y direcciones (los elementos básicos con que funciona la computadora).

Otras características del C es que posee muy pocas palabras clave (32, donde 27 fueron definidas en la versión original y cinco añadidas por el comité del ANSI, enum, const, signed, void y volatile). Todas las palabras clave de C están en minúsculas (C distingue entre las mayúsculas y minúsculas). En la siguiente tabla se muestran las 32 palabras clave:

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

Los programas en C consisten en una o más funciones. La única función que debe estar absolutamente presente es la denominada main, siendo la primera función que es llamada cuando comienza la ejecución del programa. Aunque main no forma técnicamente parte del lenguaje C, hay que tratarla

como si lo fuera, pues si se emplea para nombrar una variable, probablemente confundirá al compilador.

La forma general de un programa en C es:

Instrucciones del preprocesador

Declaraciones globales

```
tipo_devuelto main(lista de parámetros)
{
secuencia de sentencias
}

tipo_devuelto función_1(lista de parámetros)
{
secuencia de sentencias
}

tipo_devuelto función_2(lista de parámetros)
{
secuencia de sentencias
}
…..
…..
tipo_devuelto función_n(lista de parámetros)
{
secuencia de sentencias
}
```

El programa así escrito se denomina programa fuente y puede estar escrito en uno o varios ficheros.

Para que el programa pueda ser ejecutado se debe compilar y enlazar (linkar) con todas aquellas funciones de la biblioteca que se necesiten.

El proceso de compilar consiste en traducir el programa fuente a código o lenguaje máquina.

El proceso de linkaje (enlazado) consiste en añadir rutinas (propias o bibliotecas existentes en el mercado) que también están en código máquina, es decir, están en objeto.

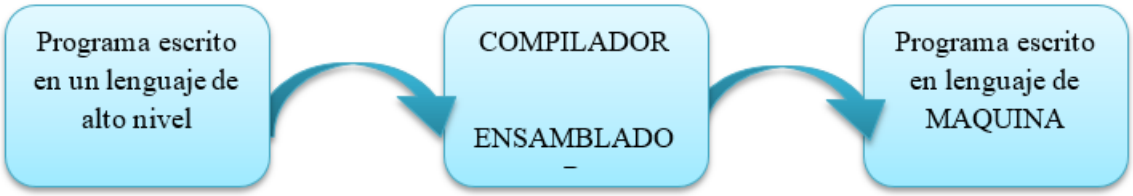
Una vez enlazado el programa objeto, tenemos un programa ejecutable que se puede ejecutar en el ordenador.

Estos procesos son realizados por un programa llamado compilador. El compilador en las máquinas Alpha del C. P. D. es el DEC OSF/1 Versión 4.0. Para compilar y enlazar un programa con este compilador basta con hacer cc nombre_del_programa.c para crear, si no hay errores, un ejecutable (a.out). Existen múltiples opciones en el compilador que se pueden comprobar con el comando de ayuda de los sistemas operativos.

Los ejemplos del curso siguen la sintaxis aceptada por el estándar ANSI, con lo que son portables con cualquier otro compilador que lo lleve implementado.

COMPILADORES

Para traducir un programa escrito en un lenguaje de alto nivel (programa fuente) a lenguaje máquina se utiliza un programa llamado compilador. Este programa tomará como datos nuestro programa escrito en lenguaje de alto nivel y dará como resultado el mismo programa pero escrito en lenguaje máquina, lenguaje que entiende el ordenador. Después de la traducción se ejecuta automáticamente un programa denominado enlazador encargado de incorporar las funciones de la biblioteca del lenguaje utilizado necesarias para nuestro programa. Este último paso será explicado detalladamente más adelante. Si durante la traducción se detectan errores de sintaxis, el enlace no se efectúa.



Por ejemplo, un programa escrito en el lenguaje C necesita del compilador C para poder ser traducido. Posteriormente el programa traducido podrá ser ejecutado directamente por el ordenador.

INTERPRETES

A diferencia de un compilador, un intérprete no genera un programa escrito en lenguaje máquina a partir del programa fuente, sino que efectúa la traducción y ejecución simultáneamente para cada una de las sentencias del programa. Por ejemplo, un programa escrito en el lenguaje Basic necesita el intérprete Basic para ser ejecutado. Durante la ejecución de cada una de las sentencias del programa, ocurre simultáneamente la traducción.

A diferencia de un compilador, un intérprete verifica cada línea del programa cuando se escribe, lo que facilita la puesta a punto del programa. En cambio la ejecución resulta más lenta ya que acarrea una traducción simultánea.

TEMA 2 – HISTORIA DEL LENGUAJE C

El C es un lenguaje de programación de propósito general. Sus principales características son:

- Programación estructurada.
- Economía en las expresiones.
- Abundancia en operadores y tipos de datos.
- Codificación en alto y bajo nivel simultáneamente.
- Reemplaza ventajosamente la programación en ensamblador.
- Utilización natural de las funciones primitivas del sistema.
- No está orientado a ningún área en especial.
- Producción de código objeto altamente optimizado.
- Facilidad de aprendizaje.

El comienzo del Lenguaje C

El lenguaje C nació en los laboratorios Bell de AT&T y ha sido estrechamente asociado con el sistema operativo UNIX, ya que su desarrollo se realizó en este sistema y debido a que tanto UNIX como el propio compilador C y la casi totalidad de los programas y herramientas de UNIX fueron escritos en C. Su eficiencia y claridad han hecho que el lenguaje ensamblador apenas haya sido utilizado en UNIX.

Este lenguaje está inspirado en el lenguaje B escrito por Ken Thompson en 1970 con intención de recodificar el UNIX, que en la fase de arranque estaba escrito en ensamblador, en vistas a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el lenguaje BCPL concebido por Martin Richard en 1967.
En 1972, Dennis Ritchie toma el relevo y modifica el lenguaje B, creando el lenguaje C y reescribiendo UNIX en dicho lenguaje. La novedad que proporcionó
el lenguaje C sobre el B fue el diseño de tipos y estructuras de datos.

Los tipos básicos de datos eran char (carácter), int (entero), float (reales en simple precisión) y double (reales en doble precisión). Posteriormente se añadieron los tipos short (enteros de longitud < longitud de un int), long (enteros de longitud > longitud de un int), unsigned (enteros sin signo) y enumeraciones. Los tipos estructurados básicos de C son las estructuras, las uniones y los arrays. Estos permiten la definición y declaración de tipos derivados de mayor complejidad.

Las instrucciones de control de flujo de C son las habituales de la programación
estructurada: if, for, while, switch-case, todas incluidas en su predecesor BCPL. C incluye también punteros y funciones y permite que cualquier función pueda ser llamada recursivamente.
Una de las peculiaridades de C es su riqueza de operadores. Puede decirse que prácticamente dispone de un operador para cada una de las posibles operaciones en código máquina.

Hay toda una serie de operaciones que pueden hacerse con el lenguaje C, que realmente no están incluidas en el compilador propiamente dicho, sino que las realiza \n preprocesador justo antes de la compilación. Las dos más importantes son #define (directriz de sustitución simbólica o de definición) e #include (directriz de inclusión en el fichero fuente).

Finalmente, C, que ha sido pensado para ser altamente transportable y para programar lo improgramable, igual que otros lenguajes tiene sus inconvenientes. Carece de instrucciones de entrada/salida, de instrucciones para manejo de cadenas de caracteres, entre otras, con lo que este trabajo queda para la biblioteca de funciones, con la consiguiente pérdida de transportabilidad.

Por otra parte, la excesiva libertad en la escritura de los programas puede llevar a errores en la programación que, por ser correctos sintácticamente no se detectan a simple vista. Por otra parte, las precedencias de los operadores convierten a veces las expresiones en pequeños rompecabezas. A pesar de todo, C ha demostrado ser un lenguaje extremadamente eficaz y expresivo.

Lenguaje C++

C++ fue desarrollado a partir del lenguaje de programación C y, con pocas excepciones, incluye a C. Esta parte de C incluida en C++ es conocida como C-, y puede compilarse como C++ sin problemas.

En 1980 se añaden al lenguaje C características como clases (concepto toma- do de Simula67), comprobación del tipo de los argumentos de una función y con- versión, si es necesaria, de los mismos, así como otras características; el resultado fue el lenguaje denominado C con Clases.

En 1983/84, C con Clases fue rediseñado, extendido y nuevamente imple- mentado. El resultado se denominó Lenguaje C++. Las extensiones principales fueron funciones virtuales, funciones sobrecargadas (un mismo identificador puede representar distintas funciones), y operadores

sobrecargados (un mismo operador puede utilizarse en distintos contextos y con distintos significados). Después de algún otro refinamiento más, C++ quedó disponible en 1985. Este lenguaje fue creado por Bjarne Stroustrup (AT&T Bell Laboratories) y documen- tado en varios libros suyos.

El nombre de C++ se debe a Rick Mascitti, significando el carácter evolutivo de las transformaciones de e("++" es el operador de incremento de C).

Posteriormente, C++ ha sido ampliamente revisado y refinado, lo que ha dado lugar a añadir nuevas características, como herencia múltiple, funciones miembro static y const, miembros protected, plantillas referidas a tipos y manipulación de excepciones. Se han revisado características como sobrecarga, enlace y manejo de la memoria. Además de esto, también se han hecho pequeños cambios para in- crementar la compatibilidad con C.

C++ es, por lo tanto, un lenguaje híbrido, que, por una parte, ha adoptado todas las características de la OOP que no perju.diquen su efectividad; por ejemplo, funciones virtuales y la ligadura dinámica (dynamic binding), y por otra parte, mejora sustancialmente las capacidades de C. Esto dota a C++ de una potencia, eficacia y flexibilidad que lo convierten en un estándar dentro de los lenguajes de programación orientados a objetos.

UNIDAD II:
DESARROLLO DE LA PROGRAMACION

REALIZACION DE UN PROGRAMA EN C

En este apartado se van a exponer los pasos a seguir en la realización de un programa, por medio de un ejemplo. La siguiente figura, muestra lo que un usuario de C debe hacer para desarrollar un programa.

Según lo que hemos visto, un ordenador sólo puede ejecutar programas escritos en lenguaje máquina. Por lo tanto, es necesario disponer de herramientas que permitan la traducción de un programa escrito en un lenguaje de alto nivel, en nuestro caso en C, a lenguaje máquina.

Por lo tanto, en la unidad de disco de nuestro sistema tienen que estar almacenadas las herramientas necesarias para editar, compilar, y depurar nuestro programa. Por ejemplo, supongamos que queremos escribir un programa denominado saludo.c. Las herramientas (programas) que tenemos que utilizar y los ficheros que producen son:

Programa	Produce el fichero
Editor	saludo.c
Compilador C	saludo.cpp
Enlazador	saludo.exe
Depurador	ejecuta paso a paso el programa ejecutable

La tabla anterior indica que una vez editado el fichero fuente saludo.c, se compila obteniéndose el fichero objeto saludo.cpp, el cual es enlazado con las rutinas necesarias de la biblioteca de C dando lugar a un único fichero ejecutable saludo.exe. Es decir que muestra el proceso

paso a paso desde el punto inicial de programación hasta el fichero ejecutable que luego podrá ser utilizado por el usuario final.

EDICION DE UN PROGRAMA

Para editar un programa, primeramente llamaremos, para su ejecución, al programa editor o procesador de textos que vayamos a utilizar. Podemos utilizar el procesador de textos suministrado con el compilador o nuestro propio procesador. El nombre del fichero elegido para guardar el programa en el disco, debe tener como extensión .c. El paso siguiente, es escribir el texto correspondiente al programa fuente. Cada sentencia del lenguaje C finaliza con un punto y coma y cada línea del programa la finalizamos pulsando la tecla Entrar (Enter).

Como ejercicio para practicar lo hasta ahora expuesto, escribir el siguiente ejemplo:

```
/ * * * * * * * * * * * * * * * * * * * * saludo * * * * * * * * * * * * * * * * * * *  /
/ * saludo.cpp */

#include <stdio.h>
main ( )
{
printf (" Hola, MUNDO.\n") ;
}
```

¿Qué hace este programa?

Comentamos brevemente cada línea de este programa. No apurarse si algunos de los términos no quedan muy claros ya que todos ellos se verán con detalle en capítulos posteriores.

Las dos primeras líneas son simplemente comentarios. Los comentarios no son tenidos en cuenta por el compilador y se colocan entre /* */.

La tercera línea incluye el fichero de cabecera stdio.h que contiene las declaraciones necesarias para las funciones de entrada-salida (E/S) que aparecen en el programa; en nuestro caso para printf. Esto significa que, como regla general, antes de invocar a una función hay que declararla. Las palabras reservadas de C que empiezan con el símbolo # reciben el nombre de directrices del compilador y son procesadas por el preprocesador de C cuando se invoca al compilador, pero antes de iniciarse la compilación.

A continuación se escribe la función principal main. Observe que una función se distingue por el modificador 0 que aparece después de su nombre y que el cuerpo de la misma empieza con el carácter {y finaliza con el carácter}.

La función printf, es una función de la biblioteca de C que escribe sobre el monitor la expresión que aparece especificada entre comillas. La secuencia de escape V que aparece a continuación de la cadena de caracteres indica al ordenador que después de escribir el mensaje, avance el cursor de la pantalla al principio de la línea siguiente. Observe que la sentencia finaliza **con punto y coma**.

El programa editado está ahora en la memoria. Para que este trabajo pueda tener continuidad, el programa escrito se debe grabar en el disco utilizando la orden correspondiente del editor.

Compilar y ejecutar el programa

El siguiente paso es compilar el programa; esto es, traducir el programa fuente a lenguaje máquina para posteriormente enlazarlo con las funciones necesarias de la biblioteca de C y obtener así un programa ejecutable. Nosotros utilizaremos como compilador el DEV C++, pero también se pueden realizar llamadas a compiladores comunes en MS-DOS o UNIX, etc.

Ejemplos:

cl saludo.c

La orden cl de MS-DOS invoca al compilador C y al enlazador para producir el fichero ejecutable saludo.exe.

cc saludo.c -o saludo

En UNIX la orden cc de invoca al compilador C y al enlazador para producir el fichero ejecutable saludo. Si no hubiéramos añadido la opción -o saludo, el fichero ejecutable se denominaría, por defecto, a.out.

Al compilar un programa, se pueden presentar errores de compilación, debidos a que el programa escrito no se adapta a la sintaxis y reglas del compilador. Estos errores se irán corrigiendo hasta obtener una compilación sin errores.

Biblioteca de funciones

Como ya dijimos anteriormente, C carece de instrucciones de E/S, de instrucciones para manejo de cadenas de caracteres, etc. con 1o que este trabajo queda para la biblioteca de funciones provista con el compilador. Una función es un conjunto de instrucciones que realizan una tarea específica. Una biblioteca es un fichero separado en el disco (generalmente con extensión .lib en MS-DOS o con extensión .a en UNIX) que contiene las funciones que realizan las tareas más comunes, para que nosotros no tengamos que escribirlas. Como ejemplo, hemos visto anteriormente la función printf. Si esta función no existiera, sería labor nuestra el escribir el código necesario para visualizar los resultados sobre la pantalla.

Para utilizar una función de la biblioteca simplemente hay que invocarla utilizando su nombre y pasar los argumentos necesarios entre paréntesis. Por ejemplo:

```
printf ( "Hola, MUNDO. \n" ) ;
```

La figura siguiente muestra como el código correspondiente a las funciones de biblioteca invocadas en nuestro programa es añadido por el enlazador cuando se está creando el programa ejecutable.

Guardar el programa ejecutable en el disco

Como hemos visto, cada vez que se realiza el proceso de compilación y enlace del programa actual, C genera automáticamente sobre el disco un fichero ejecutable. Este fichero puede ser ejecutado directamente desde el sistema operativo, sin el soporte de C, escribiendo el nombre del fichero a continuación del símbolo del sistema (prompt del sistema) y pulsando Entrar.

Cuando se crea un fichero ejecutable, primero se utiliza el compilador c para compilar el programa fuente, dando lugar a un fichero intermedio conocido como fichero objeto (con extensión .obj en MS-Dos o .o en UNIX). A continuación se utiliza el programa enlazador (link) para unir en un único fichero ejecutable, el módulo o los módulos del programa compilados separadamente y las funciones de la biblioteca del compilador C que el programa utilice.

Al ejecutar el programa, pueden ocurrir errores durante la ejecución. Por ejemplo, puede darse una división por cero. Estos errores solamente pueden ser detectados por C cuando se ejecuta el programa y serán notificados con el correspondiente mensaje de error.
Hay otro tipo de errores que no dan lugar a mensaje alguno. por ejemplo: un programa que no termine nunca de ejecutarse, debido a que presenta un lazo, donde no se llega a dar la condición de terminación.

Depurar un programa

Una vez ejecutado el programa, la solución puede ser incorrecta. Este caso exige un análisis minucioso de cómo se comporta el programa a lo largo de su ejecución; esto es, hay que entrar en la fase de depuración del programa.

La forma más sencilla y eficaz para realizar este proceso, es utilizar un programa depurador. En la CLASE Nº 1 se explica como nosotros configuraremos y utilizaremos el depurador que incluye el compilador DEV C++.