

Fundamentos de Programación

**Unidad 6: Introducción al
lenguaje ANSI/ISO C++**



Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas

2019
Pablo Novara

UN POCO DE HISTORIA...



Dennis Ritchie



Bjarne Stroustrup

UN POCO DE HISTORIA...



Que quede bien claro:
vamos a aprender programar
con el **lenguaje C++**

ESTRUCTURA DE UN PROGRAMA EN C++

```
#include <iostream>
```

de aquí "salen" std::cout y std::endl

```
int main() {
```

Algoritmo Principal

```
    std::cout << "Hola mundo!" << std::endl;
```

Escribir

salto de linea

```
}
```

FinAlgoritmo

ESTRUCTURA DE UN PROGRAMA EN C++

```
#include <iostream>

using namespace std;
Sr. compilador, agréguele "std::" a todo

int main() {

    std::cout << "Hola mundo!" << std::endl;
    simplemente "cout" y "endl", ahora sin el "std::"

}
```

ESTRUCTURA DE UN PROGRAMA EN C++

```
#include <iostream>

// using namespace std;
al igual que pseudocodigo, "//" indica comentario

int main() {

    std::cout << "Hola mundo!" << std::endl;
    sin el "std::" ni el "using" no va a compilar

}
```

ERRORES

The screenshot shows the Zinjal IDE interface. At the top is a menu bar with Archivo, Edición, Ver, Ejecución, Depuración, Herramientas, and Ayuda. Below the menu is a toolbar with various icons for file operations like new, open, save, cut, copy, paste, and search. The main window has two tabs: "hola.cpp" and "<sin_titulo_2>*". The code editor displays the following C++ code:

```
1 #include <iostream>
2 //using namespace std;
3
4 int main() {
5   cout << "Hola mundo!" << endl;
6   return 0;
7 }
```

The line "cout << "Hola mundo!" << endl;" is highlighted in red, indicating a syntax error. In the bottom-left corner of the code editor, there is a small error icon (a red circle with a white 'X').

Below the code editor is a panel titled "Resultados de la Compilación" (Compilation Results). It shows the following output:

- ★ Compilación interrumpida!
- Errores (2)
 - E sin_titulo.cpp:5:2: error: 'cout' was not declared in this scope
 - E sin_titulo.cpp:5:27: error: 'endl' was not declared in this scope
- Advertencias (0)
- + Toda la salida

The two error messages from the compilation results are also highlighted in red.

✖ Estos son errores en tiempo de compilación

ESTRUCTURA DE UN PROGRAMA EN C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hola mundo!" << endl;

    return 0;
}
```

O usualmente indica que todo fue "normal"

ERRORES

Zinjal

Archivo Edición Ver Ejecución Depuración Herramientas Ayuda

hola.cpp <sin_titulo_2>*

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hola mundo!" << endl;
6     return 0;
7 }
```

Zinjal - Consola de Ejecucion

Hola mundo!

<< El programa ha finalizado: codigo de salida: 0 >>

<< Presione enter para cerrar esta ventana >>

Resultados de la Compilación

* Ejecutando...

The screenshot shows a development environment with a menu bar (Archivo, Edición, Ver, Ejecución, Depuración, Herramientas, Ayuda) and a toolbar with various icons. Two files are open: 'hola.cpp' and '<sin_titulo_2>*'. The code in 'hola.cpp' is a simple 'Hello World' program. A red circle highlights the misspelling of 'cout' in the output statement. The terminal window shows the program's output and its exit status. The status bar at the bottom indicates the application is running.

ESTRUCTURA DE UN PROGRAMA EN C++

The screenshot shows the Zinjal IDE interface. The menu bar includes Archivo, Edición, Ver, Ejecución, Depuración, Herramientas, and Ayuda. The toolbar contains various icons for file operations and project management. The code editor window displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << 42/0 << endl;
6     return 0;
7 }
```

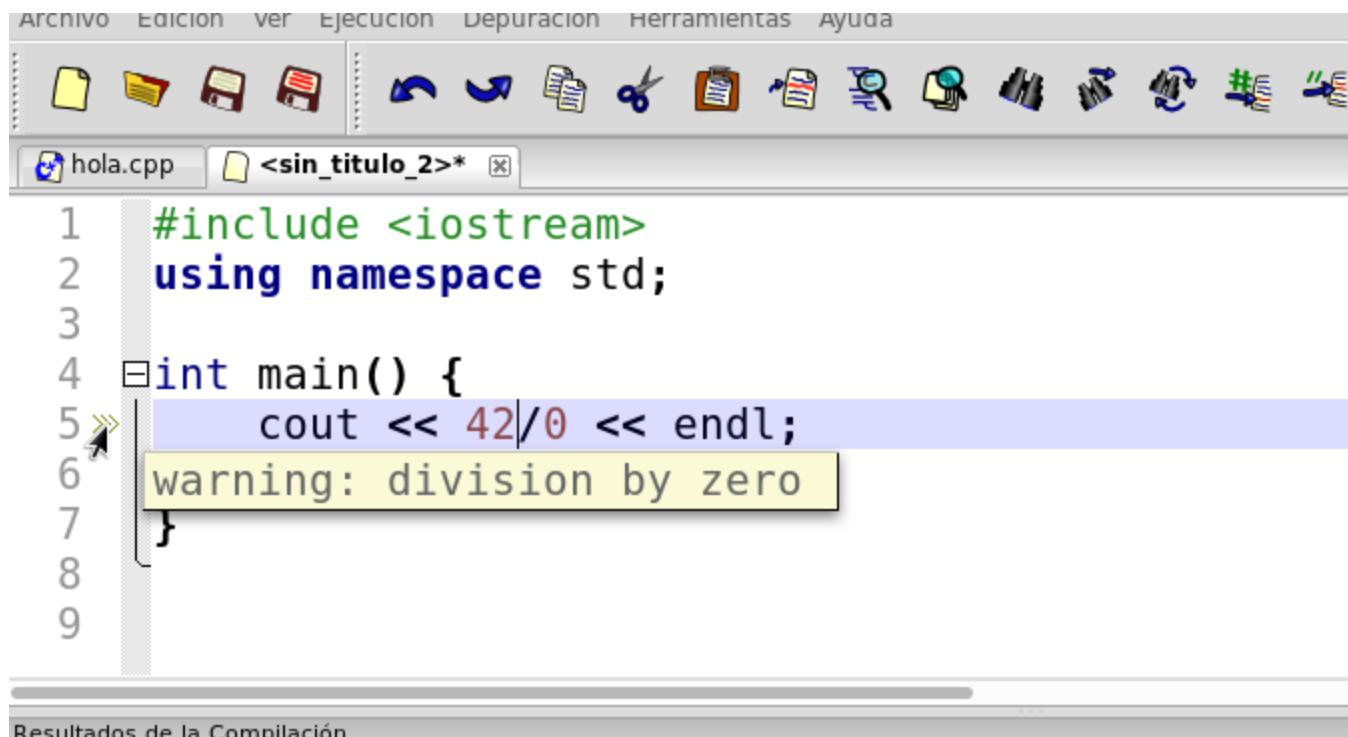
A red circle highlights the division operation `42/0`. The output window, titled "Zinjal - Consola de Ejecución", shows the error message:

```
<< El programa ha finalizado anormalmente: signal SIGFPE(8) >>
<< Presione enter para cerrar esta ventana >>
```

The word "anormalmente" and the error code "SIGFPE(8)" are circled in red.

✗ Esto es un error en tiempo **de ejecución**

ERRORES VS. ADVERTENCIAS



The screenshot shows a Windows-style IDE interface. The menu bar includes Archivo, Edicion, ver, Ejecucion, Depuracion, Herramientas, and Ayuda. The toolbar contains various icons for file operations like Open, Save, Print, and Search. Two tabs are visible: "hola.cpp" and "<sin_titulo_2>*". The code editor displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << 42/0 << endl;
6     warning: division by zero
7 }
```

A mouse cursor is positioned over the line containing the division by zero. A tooltip window appears with the text "warning: division by zero".

Below the code editor is a "Resultados de la Compilación" (Compilation Results) panel. It shows the following status:

- Ejecución Finalizada (Execution Completed)
- Errores (0) (Errors (0))
- Advertencias (1) (Warnings (1))
 - sin_titulo.cpp:5:12: warning: division by zero [-Wdiv-by-zero]
- Toda la salida (All Output)

!una "advertencia" no impide la compilación, pero indica que hay algo "sospechoso" en el código

ERRORES VS. ADVERTENCIAS

The screenshot shows the Zinjal IDE interface. At the top is a menu bar with Archivo, Edición, Ver, Ejecución, Depuración, Herramientas, and Ayuda. Below the menu is a toolbar with various icons. The main workspace shows two tabs: "hola.cpp" and "<sin_titulo_2>*". The code in "hola.cpp" is:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout , "Hola mundo!" ;
6
7 warning: left operand of comma operator has no effect
8
9 warning: right operand of comma operator has no effect
```

The line "cout , "Hola mundo!" ;" is highlighted in purple. Below the code, the "Resultados de la Compilación" panel shows:

- Ejecución Finalizada
- Errores (0)
- Advertencias (2)
 - sin_titulo.cpp:5:9: warning: left operand of comma operator has no effect [-Wunused-value]
 - sin_titulo.cpp:5:23: warning: right operand of comma operator has no effect [-Wunused-value]
- Toda la salida

✅ traten a las advertencias como si fueran errores!

VARIABLES EN C++

```
#include <iostream>
using namespace std;
```

```
int main() {
```

```
    float num;
```

Hay que declararlas (decir de qué tipo son) antes de usarlas

```
    num = 1.618;
```

En C++ se utiliza el "=" para la asignación

```
    cout << num << endl;
```

```
    return 0;
```

```
}
```

VARIABLES EN C++

```
#include <iostream>
using namespace std;

int main() {
    float num = 1.618;
    Se puede declarar e inicializar a la vez
    cout << num << endl;
    return 0;
}
```

VARIABLES EN C++

The screenshot shows the Zinjal IDE interface. The menu bar includes Archivo, Edición, Ver, Ejecución, Depuración, Herramientas, and Ayuda. The toolbar contains various icons for file operations like new, open, save, cut, copy, paste, and search. The code editor window has tabs for "hola.cpp" and "<sin_titulo_2>*". The code itself is:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     float num;
6     cout << num << endl;
7     return 0;
8 }
```

A red circle highlights the line "cout << num << endl;" with a question mark above it. A red box highlights the entire main function body. The output window titled "Zinjal - Consola de E" shows the result of running the program:

```
0
<< El programa ha finalizado: cod
<< Presione enter para cerrar est
```

The status bar at the bottom left says "Resultados de la Compilación". The compilation results pane shows:

- Ejecutando... (Running)
- Errores (0) (Errors (0))
- Advertencias (1) (Warnings (1))
 - sin_titulo.cpp:6:10: warning: 'num' is used uninitialized in this function [-Wuninitialized]
- Toda la salida (All output)

⚠ *num vale 0 de casualidad*

✖ *En general, es un error usar una var. sin inicializar*

VARIABLES EN C++

```
#include <iostream>
using namespace std;

int main() {
    string nombre;
    cout << "Ingresa tu nombre: ";
    cin >> nombre;
    cout << "Hola " << nombre << endl;
    return 0;
}
```

Leer *sin endl*

LECTURA EN C++

- Para la mayoría de los casos/tipos:

```
cin >> var1 >> var2 >> var3;
```

! En el caso de *string*, lee solo la primera palabra

- Versión alternativa para strings:

```
getline(cin, str);
```

✓ *getline* lee hasta el enter

! Si usan ambos, agregar *cin.ignore()*; al pasar de *cin>>* a *getline*

ESCRITURA EN C++

- Para la mayoría de los casos/tipos:

```
cout << var1 << var2 << var3;
```

- Se pueden además utilizar *manipuladores de flujo*

```
cout << setw(8) << right << setfill('_')  
      en 8 espacios, dato a la derecha, rellenando con '_'  
<< fixed << setprecision(3) << var;  
      reales con precisión fija, con 3 decimales
```

Salida: ____3.142



buscar como "manipulators" en CppReference

TIPOS DE DATOS NUMÉRICOS ENTEROS

Tipo	Valores	Tamaño
signed char	-127 ... +127	1 byte
unsigned char	0 ... 255	
short	-32765 ... +32765	2 bytes
unsigned short	0 ... 65335	
int	-2.14e9 ... +2.14e9	4 bytes
unsigned int	0 ... 4.29e9	
long long	-9.22e18 ... +9.22e18	8 bytes
unsigned long long	0 ... 1.84e19	

! los valores surgen de $2^n - 1$, siendo n el número de bits (1 byte = 8 bits)

TIPOS DE DATOS NUMÉRICOS REALES

Tipo	Valores	Tamaño
float	~8 dígitos	4 bytes
double	~16 dígitos	8 bytes
long double	~20 dígitos	10 bytes

⚠ Se guardan mediante una codificación similar a la notación científica. Las cantidades de dígitos son aproximadas y corresponden a la mantisa.

OTROS TIPOS DE DATOS

Tipo	Valores	Tamaño
bool 	true, false	1 byte
char 	un carácter (según código ascii)	1 byte
wchar_t	un carácter (según otro código)	2/4 bytes
string  wstring	de 0 a muchos caracteres	??
void	ninguno	?0 bytes?

!y además se pueden definir nuevos tipos...

TIPOS DE DATOS: EJEMPLOS

- Lógico:

```
bool b = true;
```

- Entero:

```
int i = 42;
```

- Real:

```
float f = 1.618;
```

- Carácter:

```
char c1 = 'a'; // comillas simples  
char c2 = 97; // c2 también vale 'a'
```

- Cadena:

```
string str = "Hola\nmundo"; // com. dobles
```

! la \ es el carácter de escape

USO DE const

El calificativo **const** indica al compilador que una variable no puede ser modificada.

```
int cont = 0; // se podrá luego  
               // modificar su valor  
const int res = 42; // NO se podrá luego  
                     // modificar su valor  
  
cont = cont+1; // ok  
res=7; // ERROR de compilación
```

⑧ ¿Para qué puede servir?

✓ Cuanto más información le doy al compilador sobre un tipo, más me ayuda a detectar errores

OPERADORES ARITMÉTICOS

Operador	En enteros	En flotantes
+ (unario)	positivo	positivo
- (unario)	negativo	negativo
+ (binario)	suma	suma
- (binario)	resta	resta
* (binario)	multiplicación	multiplicación
/ (binario)	división entera	división
% (binario)	resto de la div. entera	no aplica

⚠ *No hay operador para la potencia.
^ existe, pero significa otra cosa.*

OPERADORES RELACIONALES

Operador Significado

`==` igual

`!=` distinto

`<` menor

`>` mayor

`<=` menor o igual

`>=` mayor o igual

⚠ Se compara con doble igual...
un solo igual indica asignación

OPERADORES LÓGICOS

Operador	Significado
! (unario)	negación (no)
not (unario)	negación (no)
&& (binario)	conjunción (y)
and (binario)	conjunción (y)
(binario)	disyunción (o)
or (binario)	disyunción (o)

⚠️ los operadores & y | también existen, pero significan otra cosa

OPERADOR DE ASIGNACIÓN

- El operador de asignación es el igual:

```
var1 = var2;
```

 los operandos debe ser de tipos "compatibles"

- Se puede aplicar en cadena:

```
var1 = var2 = var3 = var4 = var5 = 0;
```

 La asignación retorna como resultado a la variable asignada.

Ejemplo:

```
var1 = var2 = 0; // equivale a var1=(var2=0);  
Luego de asignar 0 a var2, retorna var2,  
entonces, es como hacer primero var2=0;...  
...y luego var1=var2;
```

OPERADORES ARITMÉTICOS ABREVIADOS

Oper.	Ejemplo	Equivale a
<code>+ =</code>	<code>a += b</code>	<code>a = a + b</code>
<code>- =</code>	<code>a -= b</code>	<code>a = a - b</code>
<code>* =</code>	<code>a *= b</code>	<code>a = a * b</code>
<code>/ =</code>	<code>a /= b</code>	<code>a = a / b</code>
<code>% =</code>	<code>a %= b</code>	<code>a = a % b</code>
<code>++ (pre)</code>	<code>b = ++a</code>	<code>a=a+1; b=a</code>
<code>++ (post)</code>	<code>b = a++</code>	<code>b=a; a=a+1</code>
<code>-- (pre)</code>	<code>b = --a</code>	<code>a=a-1; b=a</code>
<code>-- (post)</code>	<code>b = a--</code>	<code>b=a; a=a-1</code>

FUNCIONES

Hay cientos de funciones distribuidas en las muchas bibliotecas del estándar,
y se pueden agregar más.

Ejemplos:

- ▶ En `<cmath>`:
 - ▶ `sin`, `cos`, `tan`, `acos`, `asin`, `atan`, `atan2`,
 - ▶ `pow`, `sqrt`, `exp`, `log`, `log2`, `log10`,
 - ▶ `trunc`, `round`, `ceil`, `floor`
- ▶ En `<cstdlib>`:
 - ▶ `rand`

DOCUMENTACIÓN INTERNA

El compilador ignora todo lo que se escribe...

- ▶ ...en una linea luego de //
- ▶ ...entre /* y */

Ejemplos:

```
...
float /*a, */ b; // a no se define, b sí
```

```
...
/* El segundo método
además permite extender
un comentario por varias
líneas */
...
```

EJEMPLO FINAL

1. Escriba un programa que permita ingresar los catetos de un triángulo rectángulo y determine e informe mediante una tabla:
 - longitud de la hipotenusa
 - el perímetro
 - el área

Ingrese los catetos: 5 7

Hipotenusa.....: 8.602

Area.....: 17.500

Perimetro.....: 20.602

```
#include <iostream> // por cin/cout
#include <cmath> // por sqrt
#include <iomanip> // por setfill, setw, fixed, right, left, setprecision
using namespace std;

int main() {

    // carga de datos
    cout << "Ingrese los catetos: ";
    float cat1, cat2;
    cin >> cat1 >> cat2;

    // calculos
    float hipo = sqrt( cat1*cat1 + cat2*cat2 );
    float perim = cat1+cat2+hipo;
    float area = cat1*cat2/2;

    // informe de resultados, tabla a dos columnas
    const int col1 = 15, col2 = 7; // ancho de las columnas
    // dejar linea en blanco, y setear el formato de reales a 3 decimales

    cout << endl << fixed << setprecision(3);
    // fila 1
    cout << setw(col1) << left << setfill(' ') << "Hipotenusa" << ":" 
        << setw(col2) << right << setfill(' ') << hipo << endl;
    // fila 2
    cout << setw(col1) << left << setfill(' ') << "Area" << ":" 
        << setw(col2) << right << setfill(' ') << area << endl;
    // fila 3
    cout << setw(col1) << left << setfill(' ') << "Perimetro" << ":" 
        << setw(col2) << right << setfill(' ') << perim << endl;

    return 0;
}
```