

# INF237 – Lenguaje de Programación Orientada a Objetos

Profesores: Johan Baldeón - David Allasi

# Agenda

- Introducción a la programación en C++:
  - C++.
  - Identificadores,
  - tipos de datos,
  - constantes y variables,
  - operadores.
- Escribiendo un programa en C++
- Averigüe y resuelva
- Resumen de lo aprendido

# Introducción a la programación en C++

C++. Identificadores, tipos de datos, constantes y variables, operadores.  
Escribiendo un programa en C++.

# El Lenguaje de Programación C++ (I)

- Bjarne Stroustrup diseña este lenguaje en los Laboratorios BELL, a mediados de los años 1980, con el objetivo de añadir a C nuevas características: Clases y Funciones Virtuales.
- Años más tarde, Alexander Stepanov y Adrew Koenig incorporan a C++ la librería STL, la cual contiene clases con contenedores y algoritmos genéricos que le dan una potencia única entre los lenguajes de alto nivel.
- En 1998 se aprueba el lenguaje ANSI C++.
- C++ sirvió como base del lenguaje C#.

# El Lenguaje de Programación C++ (2)

- Una particularidad de C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores) y de poder crear nuevos tipos que se comporten como tipos fundamentales.
- C++ permite trabajar tanto a alto como a bajo nivel.
- En C++, la expresión “C++” significa “incremento de C” y se refiere a que C++ es una extensión de C.



# Conociendo Visual Studio con soporte para C++

- El entorno de desarrollo integrado (IDE) que se incluye con Visual Studio con soporte para C++ es un completo entorno auto-contenido para CREAR, COMPILAR, ENLAZAR y PROBAR sus programas C++.
- Incorpora una serie de herramientas completamente diseñadas e integradas para hacer que todo el proceso de escribir programas en C++ sea muy fácil.
- Las partes fundamentales del IDE de Visual Studio con soporte para C++ son: el editor, el compilador, el enlazador y las bibliotecas.

# El Editor

- Ofrece un entorno interactivo en el que crear y editar código fuente de C++ se puede realizar con los servicios habituales como cortar y pegar.
- Reconoce palabras reservadas en el lenguaje C++ y les asigna un color característico.
- Otra característica muy útil es el IntelliSense, el cual analiza el código a medida que entren en él y pone de relieve algo que es incorrecto con un garabato rojo.

# El Compilador

- Convierte el código fuente de su programa en C++ en código objeto.
- Detecta e informa de los errores en el proceso de compilación.
- La salida de código objeto se almacena en archivos de objetos que tienen nombres con la extensión .obj.



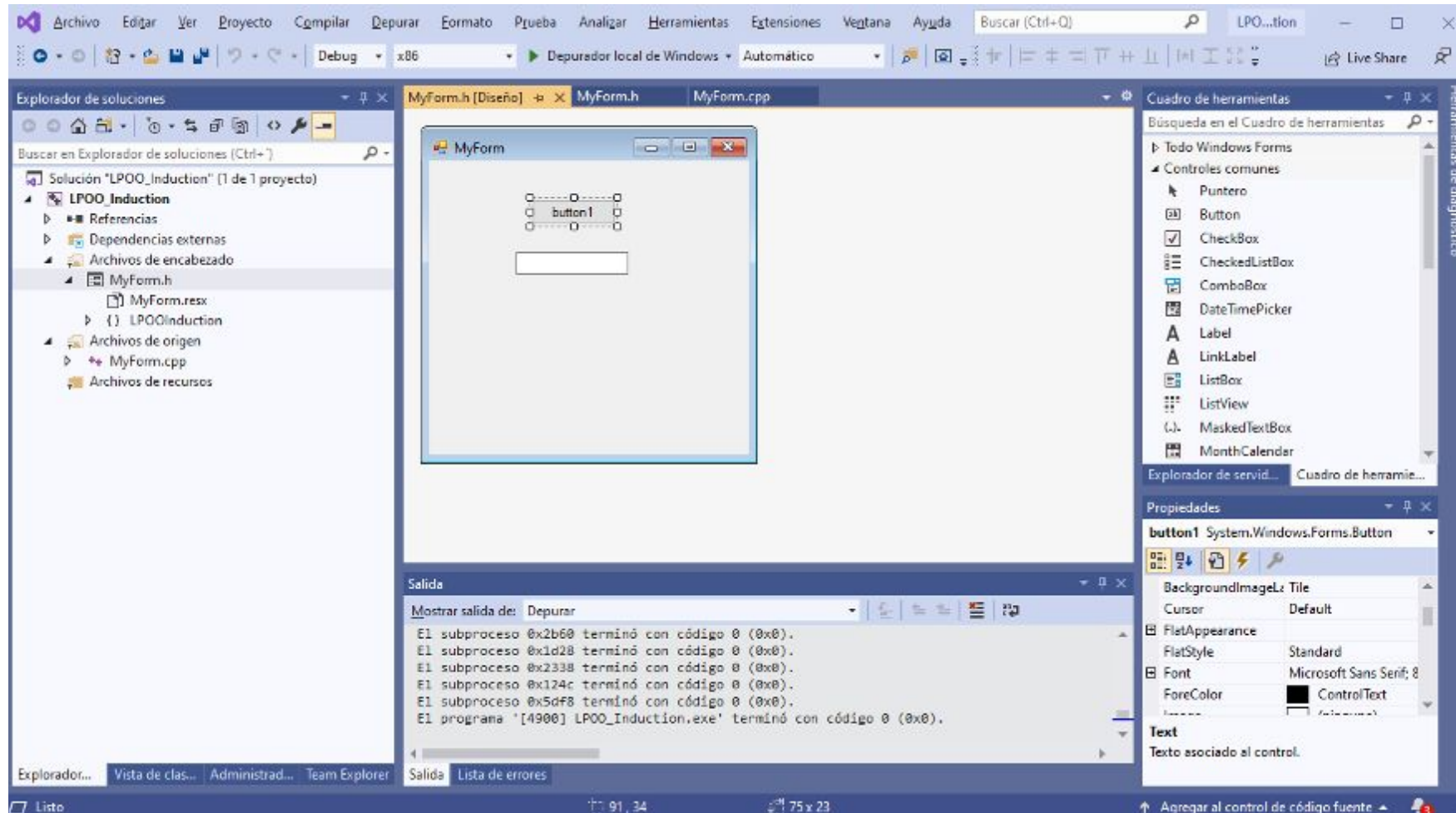
# El Enlazador

- Combina los diferentes módulos generados por el compilador y agrega los módulos de código necesario de bibliotecas que utiliza el programa todo en un ejecutable, por lo general, en la forma de archivo .exe
- Detecta y reporta errores, como por ejemplo: si una parte del código del programa no se encuentra o una biblioteca no existe.

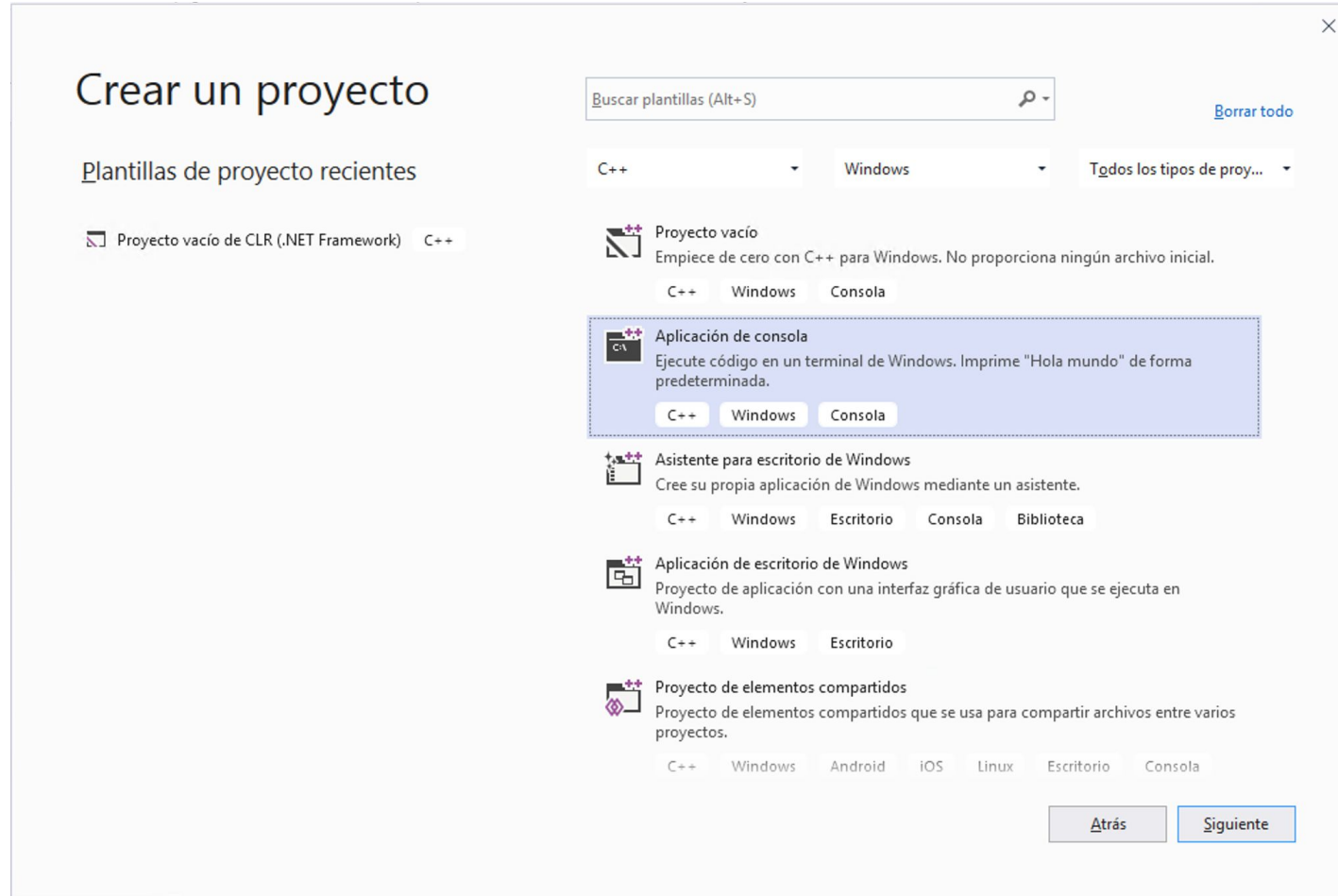
# Las Bibliotecas

- Son una colección de rutinas de código (unidades) previamente escritos que soporta y extiende el lenguaje C++.
- Ayudan a mejorar en gran medida la productividad de codificación pues ahorra el esfuerzo de escribir y probar código para ciertos tipos de operaciones.
- Existen por ejemplo bibliotecas para manejo de funciones numéricas (math.h), manejo de cadena de caracteres (string.h), etc.
- También pueden soportar la definición de nuevos tipos de datos y funciones.

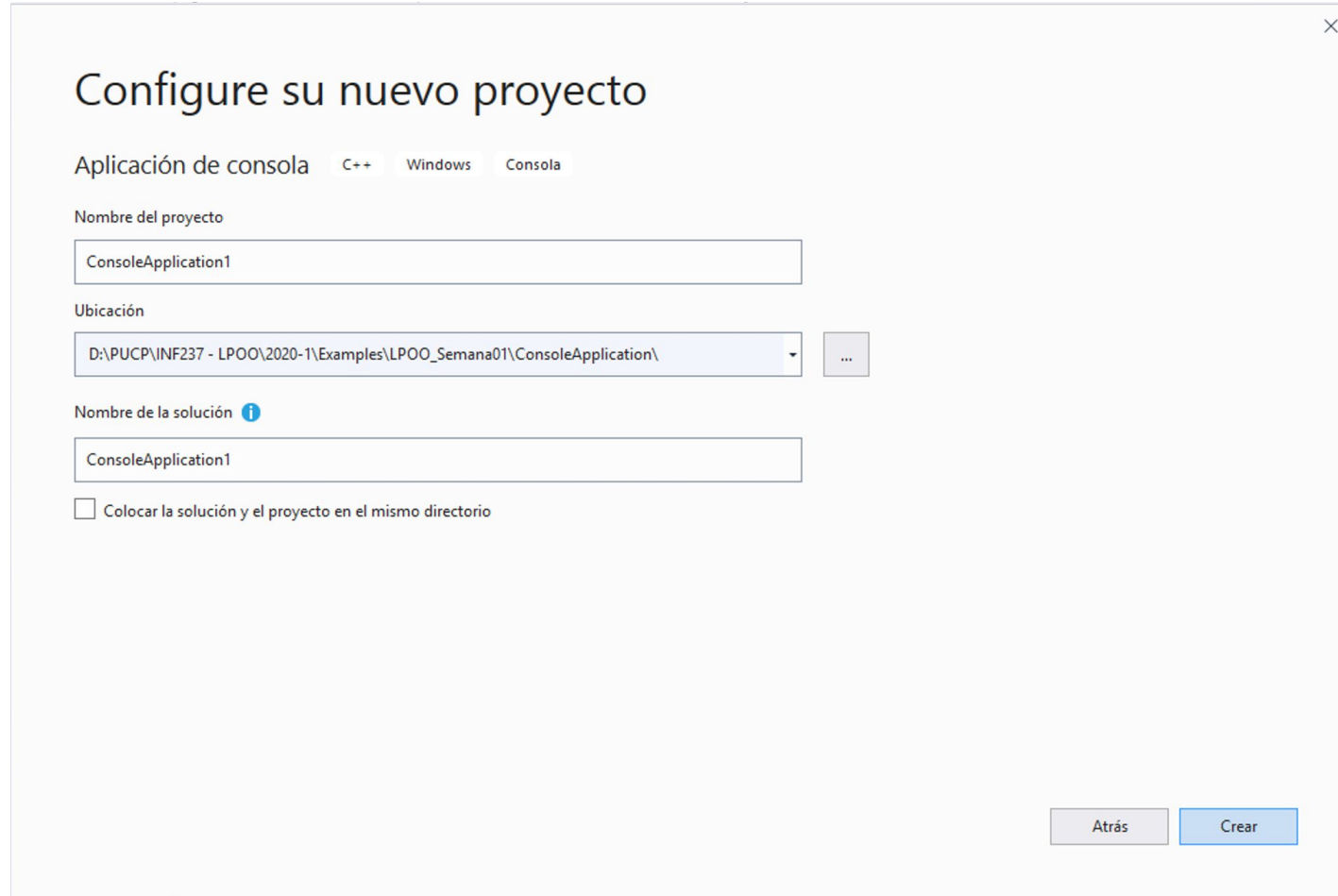
# El IDE



# Creando un proyecto de consola



# Creando un proyecto de consola



Configure your new project

Aplicación de consola C++ Windows Consola

Nombre del proyecto

ConsoleApplication1

Ubicación

D:\PUCP\INF237 - LPOO\2020-1\Examples\LPOO\_Semana01\ConsoleApplication\

Nombre de la solución ⓘ

ConsoleApplication1

☐ Colocar la solución y el proyecto en el mismo directorio

Atrás Crear

# Creando un proyecto de consola

```
// ConsoleApplication1.cpp : Este archivo contiene la función "main". La ejecución del programa comienza y termina en esta función.
//

#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Ejecutar programa:
// Depurar programa:

// Sugerencias para el depurador:
// 1. Use la ventana de Depuración
// 2. Use la ventana de Variables
// 3. Use la ventana de Puntos de interrupción
// 4. Use la ventana de Observación
// 5. Vaya a Proyecto -> Configuración de depuración
```

Consola de depuración de Microsoft Visual Studio

Hello World!

D:\PUCP\INF237 - LP00\2020-1\Examples\LP00\_Semana01\ConsoleApplication1\ConsoleApplication1.exe (proceso 4708) se cerró con el código 0.

Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas -> Opciones -> Depuración -> Cerrar la consola automáticamente al detenerse la depuración.

Presione cualquier tecla para cerrar esta ventana. . .

00 % No se encontraron problemas.

Mostrar salida de: Depurar

SHIMVIEW: ShimInfo(Complete)

'ConsoleApplication1.exe' (Win32):

'ConsoleApplication1.exe' (Win32):

'ConsoleApplication1.exe' (Win32):

El subproceso 0xe78 terminó con código 0 (0x0).

El programa '[4708] ConsoleApplication1.exe' terminó con código 0 (0x0).



# Conceptos básicos

# Identificadores

- Los elementos que se utilizan en un programa deberán estar identificados a través de un **nombre**.
- Esto se aplica para:
  - Variables
  - Constantes
  - Funciones
  - Librerías usadas
  - Clases

# Reglas de formación de identificadores

- Longitud:
  - letras y números en un máximo de 32 caracteres
- Primer carácter:
  - letra o subrayado ( \_ )
- Resto de caracteres:
  - letras, dígitos o el símbolo de subrayado ( \_ )
- Prohibiciones:
  - uso de palabras reservadas
- C++ es *case sensitive*:
  - Diferencia las mayúsculas de las minúsculas

# Ejemplos de identificadores

Correctos	Incorrectos
cont	1cont
prueba23	hola!
balance_total	$\alpha$

Case Sensitive
cuenta
Cuenta
CUENTA

# Tipos de Datos Primitivos en C++

- boolean: true o false
- char: character
- int: enteros
- float: reales
- double: reales en un rango mayor
- void: no contiene valores, no representa un tipo específico

# Tipos de Datos Primitivos en C++

Tipo	Rango	# de bits que ocupa
bool	true o false	8
char	-128 a 127	8
unsigned char	0 a 255	8
wchar_t	0 a 65,535	16
short	-32768 a 32767	16
unsigned short	0 a 65,535	16
int	-2,147,483,648 a 2,147,483,647	32
unsigned int	0 a 4,294,967,295	32

Tipo	Rango	# de bits que ocupa
long	-2,147,483,648 a 2,147,483,647	32
unsigned long	0 to 4,294,967,295	32
long long	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807	64
unsigned long long	0 a 18,446,744,073,709,551,615	64
float	3.4E-38 a 3.4E+38	32
double	1.7E-308 a 1.7E+308	64
long double	1.7E-308 a 1.7E+308 3.4E-4932 a 1.1E+4932	64 u 80 (según versión)



# Tipo de dato int: Enteros

- Se aplica a los números enteros.
- El rango de valores que admite es -2,147,483,648 a 2,147,483,647.
- Para almacenar estos valores, C emplea 32 bits (4 bytes) de la memoria:

$$2^{32} = 4,294,967,296$$

- Ejemplos:

```
int numero;  
int x, y;  
int valor = 123;
```

# Tipo de dato float: coma flotante

- Tipo usado para números reales (de coma flotante)
- El rango de posibles valores es
$$3.4 \times 10^{-38} \text{ a } 3.4 \times 10^{+38}$$
- Para su manipulación, C++ emplea 4 bytes.
- Otro tipo de variable para números reales es double
- Su rango es el correspondiente a 8 bytes por lo que es de mayor precisión

# Tipo void: sin valor

- Especifica un conjunto vacío de valores o ningún tipo.
- Usos:
  - Declarar funciones o métodos que no devuelven valor alguno.  
`void func()`
  - Declarar que una función no tiene parámetros  
`int func(void);`
  - Creación de punteros

# Variables

- Nombres de variable corresponden a posiciones en la memoria de la computadora.
- Cada variable dispone de un nombre, un tipo, un tamaño y un valor.
- Cuando un nuevo valor es puesto en una variable (a través de `scanf`, por ejemplo), reemplaza (destruye) el valor anterior.

# Declaración de variables de un tipo determinado

- Sintaxis

Tipo\_de\_dato *lista de variables;*

- Ejemplos:

- `char c;`

- `int i, j;`

- `long int potencia;`

- `double radio, longitud;`

- Existen tres sitios donde se pueden declarar variables en un programa:

- Dentro de las funciones: variables locales

- Como parámetros de las funciones

- Fuera de todas las funciones: variables globales

# Asignación de valores a las variables

- Respetar el tipo de dato
  - Existe compatibilidad de tipos:
    - char-int
    - float-int
- Se usa el símbolo de asignación: =
- Ejemplos:
  - `int x; float y;`
  - `x=10;`
  - `y= x/4.5; /* Usando operadores */`



# Variables locales

- Declaradas dentro de las funciones de un programa.
- Ejemplo:

```
void func1 (void) {  
    int x;  
    x =1;  
}
```

Variable local de la función  
func1

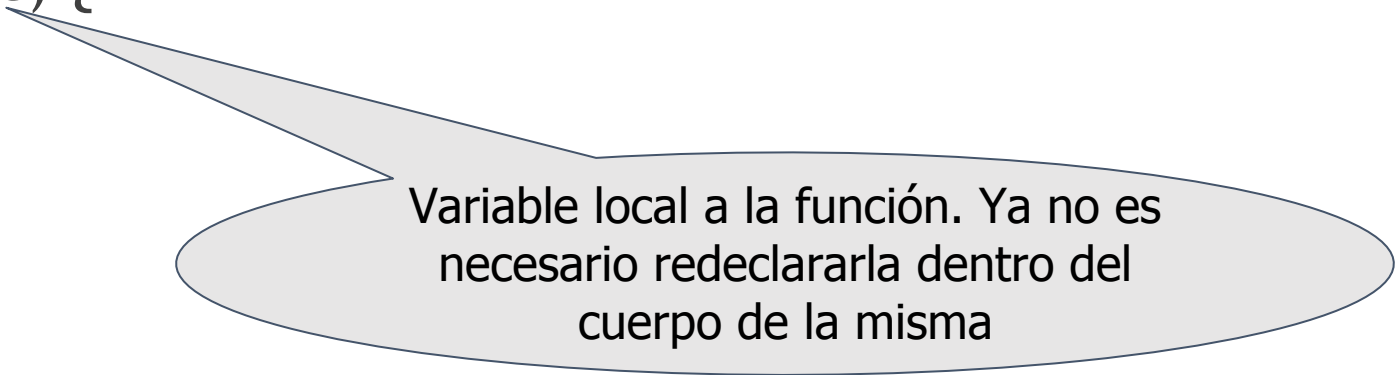
```
void func2 (void) {  
    int x = 145;  
}
```

Variable local de la función  
func2, sin relación con func1.

# Parámetros formales de funciones

- Son los argumentos de las funciones, sean funciones secundarias o el main.
- Se comportan como variables locales de cualquier función.
- Ejemplo:

```
int Cent_a_Farh(int c) {  
    . . .  
}
```



Variable local a la función. Ya no es necesario redeclararla dentro del cuerpo de la misma

# Variables globales

- Acceso a lo largo de todo el programa.
- Utilizables en la función main y en cualquier otra función adicional
- Los cambios realizados en una función se verán reflejados *en adelante*.
- Recomendable declararlas al inicio del programa.

# Constantes (I)

- Como en todo lenguaje de programación, las constantes son identificadores cuyo valor *no cambia a lo largo del programa*
- Dos formas:
  - Uso de la palabra reservada **const**
  - Uso de la sección de declaraciones **#define**

# Constantes (2)

- Usando const
  - Sintaxis:
    - `const tipo_de_dato identificador = valor;`
  - Ejemplo:
    - `const int cuenta=100;`

# Constantes (3)

- Usando #define:
  - Se coloca al comienzo del programa (global):

```
#define PI 3.14159
```

```
#define MAXIMO 999
```

```
#define ULTIMALETRA 'Z'
```

```
#define MENSAJE “Introduzca su edad:”
```



# Constantes de barra invertida

- Los llamados *caracteres de barra invertida*, son constantes de tipo char que representan caracteres especiales.

Caracter	Significado
\n	Imprime un salto de línea
\r	Imprime un retorno de carro
\t	Imprime una tabulación (uso de la tecla TAB)
\"	Imprime una comilla doble
\'	Imprime una comilla simple
\\	Imprime un back slash
\0	Caracter nulo

# Operadores

# Introducción a los operadores

- Es uno o varios caracteres que le indican al compilador que realice determinadas operaciones aritméticas o lógicas.
- Se analizarán:
  - Operadores Aritméticos
  - Operadores Incrementales
  - Operadores Relacionales
  - Operadores Lógicos
  - Otros

# Operadores aritméticos

- Puede aplicarse los operadores comunes a todos los lenguajes de programación

Operador	Acción
+	Suma
-	Resta, número negativo
/	División
%	Módulo o resto de la división entera
--	Decremento en 1
++	Incremento en 1

# Operadores aritméticos: división / (I)

- Particularidades del operador /
- Su uso es permitido entre variables enteras, arrojando como resultado el cociente de la división entera.
- Ejemplo 1:
  - `int a=10, b= 3;`
  - `c = a / b; /*El valor de c es 3*/`

# Operadores aritméticos: división / (2)

- Su uso es permitido entre variables reales, teniéndose dos alternativas:
  - Ejemplo 2:
    - `float f;`
    - `f = 10/ 3; /*El valor de f es 3*/`
  - Ejemplo 3:
    - `float f;`
    - `f = 10.0 / 3.0; /*Recién aquí el valor es 3.33*/`

# Operadores aritméticos: módulo %

- Este operador proporciona el residuo de la división entera entre dos números enteros.
- *No debe ser usado con números de tipo float*
- Ejemplo:
  - `int a=10, b=3, c;`
  - `c = a%b; /* c tendrá el valor de 1*/`

# Operador de incremento: ++

- Incremento ++
  - Incrementa en 1 el valor de una variable
  - Sea `int x= 10;`
  - Las siguientes instrucciones son equivalentes:
    - `x = x + 1;`
    - `x++;`
    - `++x;`
  - Uso indistinto para enteros y float



# Operador de decremento: --

- Decremento --
  - Decrementa en 1 el valor de una variable
  - Sea `int x= 10;`
  - Las siguientes instrucciones son equivalentes:
    - `x = x - 1;`
    - `x--;`
    - `--x;`
  - Uso indistinto para enteros y float

# Operadores relacionales (I)

\*Establece relaciones entre valores de identificadores

Operador	Significado
>=	Mayor o igual
>	Mayor
<=	Menor o igual
<	Menor
==	Igual
!=	No igual

# Operadores relacionales (2)

Por ejemplo:

```
int a=2, b=3, c=6;
```

```
(a == 5)    // false, porque a no es igual a 5
```

```
(a*b >= c)  // true, porque (2*3 >= 6) es verdadero
```

```
(b+4 > a*c) // false, porque (3+4 > 2*6) es falso
```

```
((b=2) == a) // true
```

# Operadores lógicos

- Se manejarán tres operadores lógicos:
  - && que representa a la conjunción **y**
  - || que representa a la disyunción **o**
  - ! que representa a la negación **no**

Por ejemplo:

`!(5 == 5) // false, porque la expresión (5 == 5) es verdadero`

`!(6 <= 4) // true, porque (6 <= 4) es falso`

`!true // false`

`!false // true`

# Operador ternario condicional

El operador condicional evalúa una expresión, devolviendo un valor si esa expresión se evalúa como verdadera, y uno diferente si la expresión se evalúa como falsa. Su sintaxis es:

```
condition ? result1 : result2
```

Si `condition` es verdadero, toda la expresión se evalúa como `result1`, de lo contrario, como `result2`.

Por ejemplo:

```
7==5 ? 4 : 3    // retorna 3, porque 7 no es igual a 5.
```

```
7==5+2 ? 4 : 3  // retorna 4, porque 7 es igual a 5+2.
```

```
5>3 ? a : b     // retorna el valor de a, porque 5 es mayor que 3.
```

```
a>b ? a : b     // retorna el mayor entre a o b.
```

# Operador coma

El operador coma (,) se usa para separar dos o más expresiones que están incluidas en donde solo se esperaba una expresión. Cuando el conjunto de expresiones tiene que ser evaluado para un valor, sólo se considera la expresión que está más a la derecha.

Por ejemplo, el siguiente código:

```
a = (b=3, b+2)
```

Primero se asignaría el valor de 3 a b, y luego se asignaría  $b + 2$  a la variable a. Entonces, al final, la variable a contendría el valor de 5 mientras que la variable b contendría el valor de 3.

# Operador de casting de tipo explícito

Los operadores de casting de tipos permiten convertir un valor de un tipo dado a otro tipo. Hay varias maneras de hacer esto en C++. La forma más simple, el cual ha sido heredado de C, es preceder la expresión a ser convertida por el nuevo tipo encerrado entre paréntesis ():

```
int i;  
float f = 3.14;  
i = (int) f;    // i tendrá el valor de 3
```

Otra forma de hacer lo mismo en C++ es usar la notación funcional precediendo la expresión a ser convertida por el tipo y encerrando la expresión entre paréntesis:

```
i = int (f);    // i tendrá el valor de 3
```

# Entrada y Salida en C++

Las bibliotecas estándar de C++ proporcionan las capacidades de entrada/salida (E/S).

La E/S de C++ se da en flujos de bytes. Un flujo es una secuencia de bytes. En las operaciones de entrada, los bytes fluyen desde un dispositivo (por ejemplo un teclado, una unidad de disco) hacia la memoria principal. En operaciones de salida los bytes fluyen de la memoria principal hacia un dispositivo (por ejemplo la pantalla, la impresora, una unidad de disco).



# Entrada y Salida en C++: cin y cout (I)

- cin es el flujo de entrada estándar que normalmente es el teclado
- cout es el flujo de salida estándar que por lo general es la pantalla.
- Se debe incluir el archivo de encabezado `iostream` para el manejo del flujo de entrada/salida.
- Por ejemplo, para imprimir una línea de texto:

```
#include <iostream>
int main() {
    cout << "Hola, que facil es LPOO";
    cout << endl;
    return 0;
}
```

# Entrada y Salida en C++: cin y cout (2)

- Otro ejemplo, para leer desde el teclado y escribir en pantalla:

```
int main() {  
    int i1,i2,sum;  
    cout << "Ingrese el 1er numero entero\n"; cin >> i1;  
    cout << "\nIngrese el 2do numero entero\n"; cin >> i2;  
    sum = i1+i2;  
    cout << "\nLa suma es " << sum << endl;  
  
    return 0;  
}
```

# Salida en C++: put (2)

- La función miembro `put` envía a la salida un caracter:

```
cout.put('A'); // Imprime A
```

- Las llamadas a `put` pueden ponerse en cascada como en:

```
cout.put('A').put('\n'); // Imprime A y un salto de línea
```

- ¿Qué imprime la siguiente instrucción?

```
cout.put('A').put('\n').put('B').put(65);
```

# Escribiendo un programa en C++

# Estructura de un programa en C++

// Comentarios de mi primer programa en C++

#include <iostream>

Directiva que indica al preprocesador que incluya una sección de código C++ estándar conocida como la cabecera *iostream*, que permite realizar operaciones de entrada y salida estándar, como escribir la salida de este programa (Hola Mundo! ) a la pantalla.

int main()

{

std::cout << "Hola Mundo!";

}

Inicia la declaración de la función *main*, que es una función especial en todos los programas en C++, ya que es la función que se invoca cuando el programa se ejecuta.

Cuerpo de la función *main* delimitado por una llave de apertura y una llave de cierre. En su interior se encuentran las instrucciones a realizar cuando el programa se ejecute.

Hola Mundo!

Lo que se muestra en la pantalla del usuario al ejecutar el programa.

# Estructura de un programa en C++ (2)

- Ejemplo 2: Programa que convierte grados Celsius a Fahrenheit:

```
#include <iostream>
```

```
int main() {
```

```
    float c, f;
```

```
    std::cout << "Ingrese el valor en grados Celsius: ";
```

```
    std::cin >> c;
```

```
    f = (9.0 / 5.0) * c + 32;
```

```
    std::cout << "Su equivalente en grados Fahrenheit es: " << f;
```

```
}
```

```
Ingrese el valor en grados Celsius: 28
```

```
Su equivalente en grados Fahrenheit es: 82.4
```

# Averigüe y resuelva (I)

- ¿Cómo haría para leer una cadena (string) y guardarlo en un arreglo de caracteres?
- Elabore un programa que solicite su nombre, apellido paterno, y fecha de nacimiento en el formato DD MM AA e imprima en pantalla:

“Hola <nombre> <apellido>, su edad es 19 años.”

# Averigüe y resuelva (2)

- ¿Qué son los namespaces y por qué se utilizan?
- Actualice los tres programas anteriores para que utilicen namespaces.

## Para responder en el cuestionario del PAIDEIA

- ¿Qué ventajas y desventajas encuentra en el lenguaje C++?
- ¿Qué tipos de usos y aplicaciones se la da a C++?
- ¿Sabía que hay versiones de C++? Describa dos nuevas características de la versión más actual.
- ¿Qué es un puntero? ¿C++ lo permite?



# Averigüe y resuelva (3)

## Para responder en el cuestionario del PAIDEIA

- Además de los tipos de operadores revisados, ¿cómo funcionan los operadores a nivel de bit?
- ¿Qué son las enumeraciones (*enum*) y cómo se usan?
- ¿Qué hace el operador *sizeof* ?
- ¿Qué hace el operador *static\_cast* ?
- ¿Qué es el preprocesador? y ¿qué hacen las directivas de preprocesamiento?
- ¿Qué son las declaraciones en C++? Brinde dos ejemplos.

# Resumen de lo aprendido (I)

- C++ es un lenguaje de programación que proviene de la extensión del lenguaje C para manipular objetos.
- C++ ofrece una gran cantidad de bibliotecas con muchas funciones que ayudan a escribir código rápidamente.
- La compilación y ejecución de los programas en C++ son más rápidas que la mayoría de otros lenguajes.
- C++ es un lenguaje compilado.
- Los identificadores son los nombres que se les da a las variables y funciones de un programa.
- Los tipos de datos en C++ se clasifican en primitivos y derivados.

# Resumen de lo aprendido (2)

- Una variable es un espacio en la memoria en el que se asigna un valor determinado de acuerdo con el tipo de dato declarado.
- Una constante es un valor que no puede ser alterado y existen dos maneras de declararlas en C++.
- Las bibliotecas estándar en C++ permite la entrada y salida de datos.

# Bibliografía

- Kernighan, B., & Ritchie, D. (1988). The C programming language (2nd ed.). Prentice Hall
- Stroustrup, B. (2014). A tour of C++. Pearson Education.
- Stroustrup, B. (2014). Programming: Principles and Practice Using C++ (2nd ed.). Addison-Wesley.
- Stroustrup, B. (2018). *The C++ programming language* (4th ed.). Addison-Wesley.
- Stroustrup, B. (1994). *The design and evolution of C++*. Reading, Mass: Addison-Wesley.