

# C++ | 01

Martin **Nelbren** Cuellar | 2022-10-19 **v1.2**

# El origen de C++

“ C++ evolucionó a partir de C, que fue desarrollado por [Dennis Richie](#) en los laboratorios Bell. C está disponible para la mayoría de las computadoras y es independiente del hardware. Con un diseño cuidadoso, es posible escribir programas en C que sean **portables** para la mayoría de las computadoras. (Deitel, 2008). ”

# ¿Que es C++?

“ El lenguaje C++, que es una extensión de C, lo desarrollo [Bjarne Stroustrup](#) en 1979, en los laboratorios Bell. Conocido en un principio como "C con clases", se cambio su nombre a C++ a principios de la década de 1980. C++ ofrece varias características que "pulen" al lenguaje C pero, lo más importante es que proporciona las capacidades de una programación orientada a objetos. (Deitel, 2008). ”

# Procedamos con lo básico

# **Tipos primitivos, modificadores, rango**

*Conocer, aplicar y comparar los diferentes tipos  
primitivos y sus modificadores*

# Tipos primitivos

Tipo	Contiene	Ejemplo
bool	Booleano	true
char	Caracter	'x'
wchar_t	Caracter UNICODE	'Φ'
int	Número Entero	35000
float	Número Real	5.5
double	Número Real	-956.1007
<b>void</b>	Vacio	

# Modificadores de tipo

## Modificadores Contiene

---

signed	Números negativos y positivos
--------	-------------------------------

unsigned	Números positivos
----------	-------------------

short	Optimiza espacio y 16 bits
-------	----------------------------

long	Anchura de al menos 32 bits
------	-----------------------------

“ **NOTA:** existen también *calificadores de tipo* `const` el cual definen al tipo como una constante. ”

# C++\_numeric-limits-2

```
[neibren@ndev-vps-01:~/numeric-limits-2]$ ./numeric-limits-2
      TYPE NAMES, MODIFIERS AND NUMERIC LIMITS FOR C++ TYPES OF VARIABLES
=====
          TYPE INTEGER? SIGNED? DIG10 MAXDIG10          MIN          MAX BYTES BITS
=====
  bool      true  false   0    0           0           1    1    8
  char      true  true   2    0          -128          127    1    8
unsigned char  true  false   2    0           0           255    1    8
 wchar_t    true  true   9    0         -2147483648        2147483647    4    32
 int8_t     true  true   2    0          -128          127    1    8
 uint8_t    true  false   2    0           0           255    1    8
 int16_t    true  true   4    0         -32768          32767   2    16
 uint16_t   true  false   4    0           0           65535   2    16
 int32_t    true  true   9    0         -2147483648        2147483647    4    32
 uint32_t   true  false   9    0           0           4294967295   4    32
 int64_t    true  true  18    0       -9223372036854775808  9223372036854775807    8    64
 uint64_t   true  false  19    0           0           18446744073709551615    8    64
 short      true  true   4    0          -32768          32767   2    16
 unsigned short  true  false   4    0           0           65535   2    16
 int         true  true   9    0         -2147483648        2147483647    4    32
 unsigned int  true  false   9    0           0           4294967295   4    32
 long        true  true  18    0       -9223372036854775808  9223372036854775807    8    64
 unsigned long  true  false  19    0           0           18446744073709551615    8    64
 long long   true  true  18    0       -9223372036854775808  9223372036854775807    8    64
 unsigned long long  true  false  19    0           0           18446744073709551615    8    64
 float       false  true   6    9          1.17549e-38        3.40282e+38    4    32
                           EPSILON: 1.19209e-07 MIN EXPONENT: -37 MAX EXPONENT: 38
 double      false  true   15   17         2.22507e-308        1.79769e+308    8    64
                           EPSILON: 2.22045e-16 MIN EXPONENT: -307 MAX EXPONENT: 308
 long double  false  true   18   21         3.3621e-4932        1.18973e+4932   16   128
                           EPSILON: 1.0842e-19 MIN EXPONENT: -4931 MAX EXPONENT: 4932
=====

2020-02-06 - http://neibren.com - basado en https://rextester.com/BBXAM15894
[neibren@ndev-vps-01:~/numeric-limits-2]$
```

# Ejercicio de Clase 01

```
#include <iostream>
using namespace std;
int main() {
    int i = 65;
    char c = i + 1;
    float f = 67.5;
    cout << "i=" << i << ", c=" << c << ", f=" << f;
    cout << endl << "i=" << (char)i << ", c=";
    cout << (int)c << ", f=" << (char)((int)f) << endl;
}
```

# Estructuras de control en C++

*Comprender las diferencias entre Java y C++ en términos de estructuras de control*

# Estructura de control if

```
if (condicion) {  
    cout << "Se cumplió la condición" << endl;  
} else {  
    cout << "No se cumplió la condición" << endl;  
}
```

# Estructura de control **switch**

```
#include <iostream>
using namespace std;
int main(void) {
    int opcion;
    cin >> opcion;
    switch(opcion){
        case 1: cout << "Opción 1"; break;
        case 2: cout << "Opción 2"; break;
        default: cout << "Default"; break;
    }
}
```

# Definición de “verdadero” y “falso” en C++

- **Verdadero** sí la **condición** es un valor distinto de **cero**.



Se cumplio la condición

- **Falso** sí la **condición** es un valor igual **cero**.



No se cumplio la condición

**“** **NOTA:** se usa el tipo `bool` en variables que solo  
pueden tomar dos valores `true` o `false`, sin  
embargo, se puede usar cualquier tipo que acepte  
un valor de `0` para ~~falso~~ y **cualquier otro valor**  
**diferente de cero** para ✓ **verdadero.** **”**

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    if (6.66) cout << "1. 6.66" << endl;
    if (0) cout << "2. 0 <--- OJO!" << endl;
    if (-1) cout << "2. -1" << endl;
    if ('A') cout << "3. 'A'" << endl;
    if (2 > 1) cout << "4. 2 > 1" << endl;
    if (1 > 2) cout << "4. 1 > 2 <--- OJO!" << endl;
    if (a = 0) cout << "5. a = 0 <--- OJO!" << endl;
    a = 1; b = 2;
    if (a == b) cout << "5. a(" << a << ") == b(" << b << ")" <- OJO!" << endl;
    a = 1; b = 2 - 1;
    if (a == b) cout << "5. a(" << a << ") == b(" << b << ")" )" << endl;
    if ('\0') cout << "6. '\0' <- OJO!" << endl;
}
```

```
#include <iostream>
using namespace std;
int main() {
    int a = !true;
    if (a == false) cout << "6. a(" << (bool)a << ")" << endl;
    a = true;
    if (!(!a)) cout << "7. a(" << (bool)a << ")" << endl;
    if (a == true) cout << "8. a(" << (bool)a << ")" << endl;
    if (a) cout << "9. a(" << (bool)a << ")" << endl;
    if (a == false) cout << "10. a(" << (bool)a << ") <- OJO!" << endl;
    if (!a) cout << "10. a(" << (bool)a << ") <- OJO!" << endl;
    a = !a;
    if (!a) cout << "10. a(" << (bool)a << ")" << endl;
    if ('0') cout << "11. '0'" << endl;
}
```

# Instrucciones de repetición en C++

# Instrucción de repetición while

```
#include <iostream>
using namespace std;
int main(void) {
    int contador = 0;
    while (contador <= 9) {
        cout << contador++ << " ";
    }
}
```

```
0 1 2 3 4 5 6 7 8 9
```

# Uso de bloque de instrucciones versus una instrucción

```
while (contador <= 9)
    cout << contador++ << " ";
```

“ **NOTA:** El primer bloque {} de instrucciones es equivalente al de **una** sola instrucción porque justamente solo se necesita **una** instrucción. ”

## **break y continue**

- **break**

Palabra reservada para salirse del ciclo más interno

- **continue**

Palabra reservada que causa que el código del bloque de un ciclo salte directamente a evaluar la condición.

# Instrucción de repetición do-while

```
#include <iostream>
using namespace std;
int main(void) {
    int opcion = 0;
    do {
        cout << "1=Opcion#1, 2=Opcion#2, 0=Fin: ";
        cin >> opcion;
    } while (opcion);
}
```

```
1=Opcion#1, 2=Opcion#2, 0=Fin: 1
1=Opcion#1, 2=Opcion#2, 0=Fin: 0
```

# Instrucción de repetición for

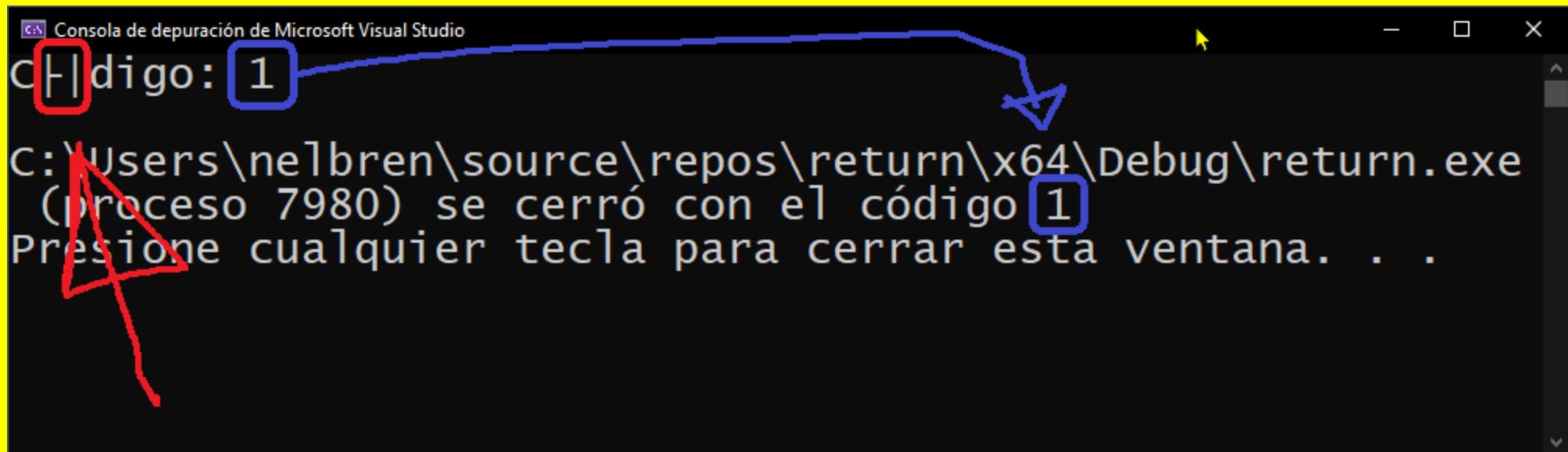
```
#include <iostream>
using namespace std;
int main(void) {
    for(int i = 9; i; i--) {
        cout << i << " ";
    }
}
```

```
9 8 7 6 5 4 3 2 1
```

# Uso de código de retorno

```
#include <iostream>
using namespace std;
int seleccion() {
    int codigo;
    cout << "Código: ";
    cin >> codigo;
    return codigo;
}
int main(void) {
    return seleccion();
}
```

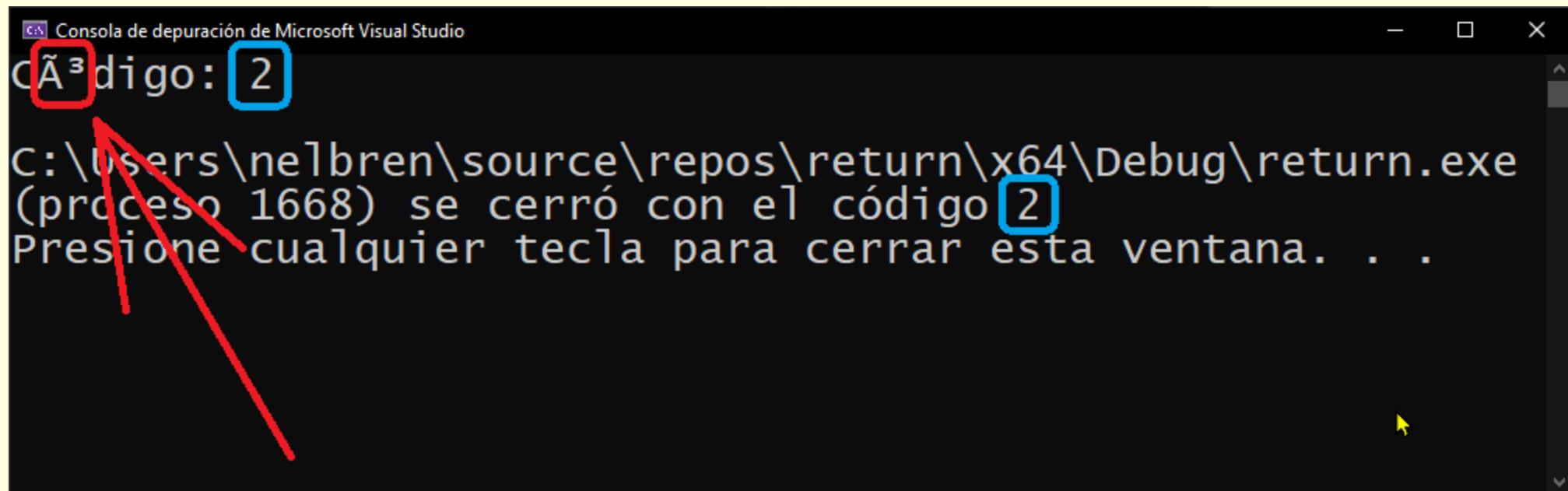
# X C++ | digo != Código !!



# Fijar localización

```
#include <iostream>
#include <locale>
using namespace std;
int seleccion() {
    int codigo;
    setlocale(LC_ALL, "spanish");
    cout << "Código: ";
    cin >> codigo;
    return codigo;
}
int main(void) {
    return seleccion();
}
```

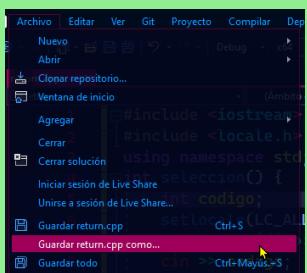
# **X CÃ³digo != Código !**



# Guardar el archivo fuente para que muestre los acentos

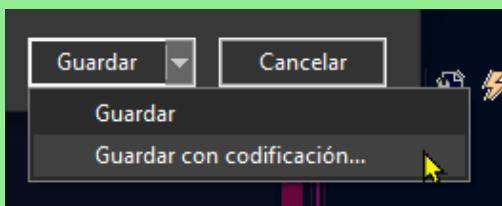
## Paso 1

Guardar como...



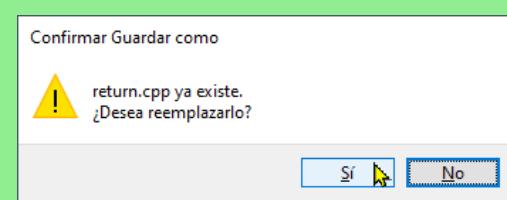
## Paso 2

Guardar  
con  
codificación



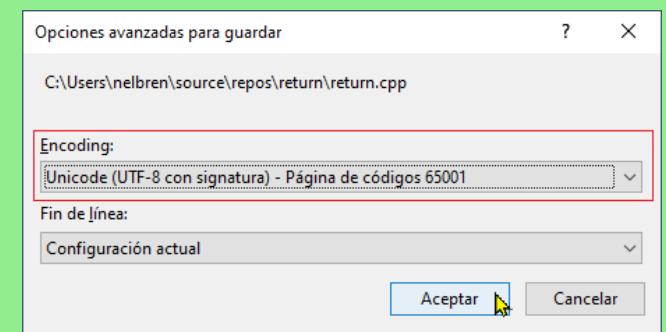
## Paso 3

Reemplazarlo

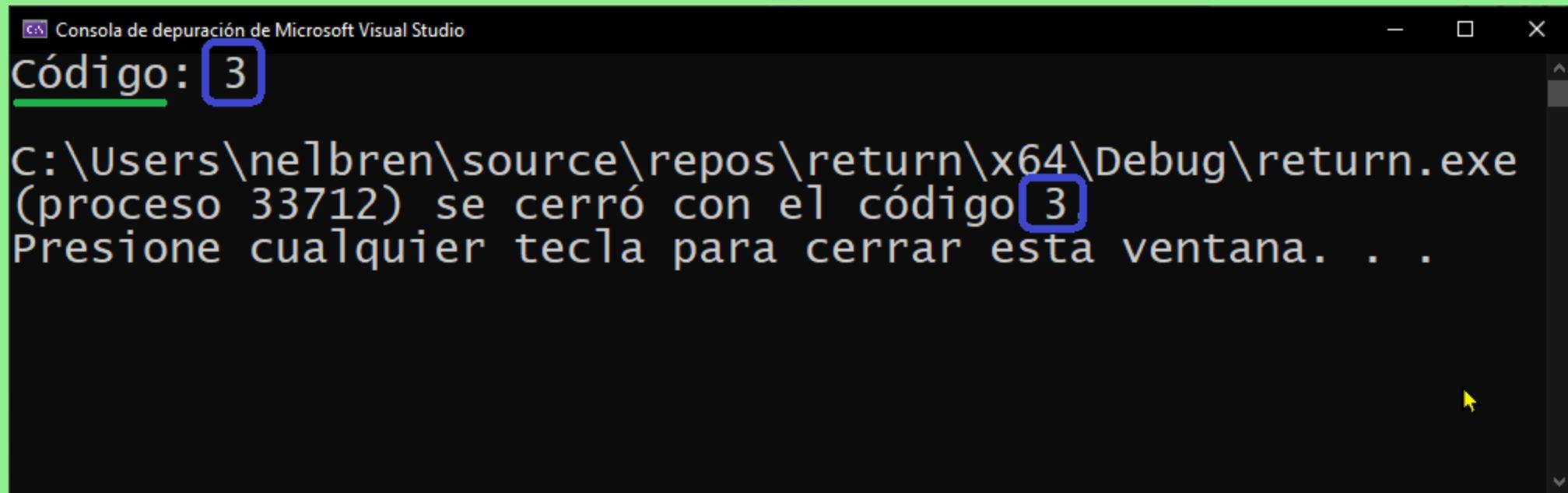


## Paso 4

Unicode (UTF-8  
con firma)  
- Página de  
códigos 65001



# ✓ Código == Código 😎



Consola de depuración de Microsoft Visual Studio

Código: 3

```
c:\Users\nelbren\source\repos\return\x64\Debug\return.exe
(proceso 33712) se cerró con el código 3
Presione cualquier tecla para cerrar esta ventana. . .
```

```
@REM return.bat
@ECHO OFF
return.exe
IF %ERRORLEVEL% EQU 0 GOTO CERO
IF %ERRORLEVEL% EQU 1 GOTO UNO
GOTO DESCONOCIDO
:CERO
ECHO Se retorno un cero.
GOTO FIN
:UNO
ECHO Se retorno un uno.
GOTO FIN
:DESCONOCIDO
ECHO No se que hacer.
:FIN
```

**PS:** echo \$LastExitCode | **UNIX:** echo \$?

# Ejercicio de Tarea 01 - Hacer un programa:

- Que solicite un **número** y finalicé el bucle con **cero**
- Muestre **Par** o **Impar** utilizando el operador de *modulo* **%** y el *operador ternario* ( condicion ? verdadero : falso )
- Máximo **12** líneas de **código** y solo usar **2** **cout**
- La salida debe ser **idéntica** a la siguiente:

```
Número: 1  
El número 1 es impar.  
Número: 2  
El número 2 es par.  
Número: 0  
El número 0 es par.
```

# Ejercicios de Punto de Control

#	Tipo - Capítulo	Página	Ejercicio	Puntos
1	C++ - 01	31	<b>01</b>	0.5
2	Libro - 2	64	<b>2.28</b>	0.5
3	Libro - 4	151	<b>4.17</b>	0.5
4	Libro - 5	196	<b>5.12</b>	0.5
<b>Total</b>				<b>2</b>

# Rúbrica

#	Descripción	Sanción
1	Mal <b>sangrado</b> de código	25%
2	No usar variables <b>significativas</b>	25%
3	<b>Nombre</b> de archivos: ejercicio_X.cpp Donde <b>X</b> es el número (#) de ejercicio	25%
4	Si incluye cualquier otro archivo	25%
5	Archivo <b>comprimido</b> : CPP_1_YYY.zip Donde <b>YYY</b> es la cuenta del alumno	25%

**¡Gracias por su atención!**



# Referencias

- Deitel, P.J. & Deitel, H.M. (2008). Cómo programar en C++. Pearson Educación. | Cap. 2, 4, 5 y Apéndice C
- Shows C++ numeric limits for all possible numerical types
- c++\_numeric-limits-2