

INFORMÁTICA INDUSTRIAL

PROGRAMACIÓN BÁSICA C++

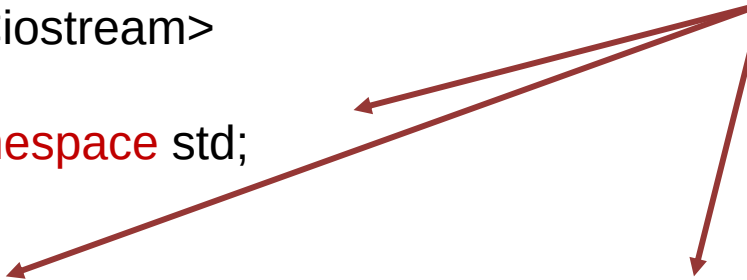
Profesor: M. Abderrahim / F. Abu-Dakka / A. Castro
Departamento de Ingeniería de Sistemas y Automática

1. Introducción

Introducción - Estructura de un programa C++

```
#include <iostream>

using namespace std;
int main()
{
    cout << "\n Hello World" << endl;
    return 0;
}
```



The diagram illustrates the flow of execution in a C++ program. Three red arrows originate from a single point on the right and point to the `#include <iostream>`, `using namespace std;`, and `cout << "\n Hello World" << endl;` statements, indicating the order of execution.

Introduction – Program Structure

C++

```
#include <iostream>
```

Librerías

```
/* My first simple C/C++ program */
```

Comentarios

```
// This also a comment
```

```
using namespace std;
```

```
int main ()
```

Todos los programas en C tienen una función “main” (principal); también empiezan en el main

```
{
```

Función para imprimir por pantalla

```
cout << "Welcome to C/C++!\n" << endl ;
```

```
return 0;
```

```
}
```

Qué imprimir

Final de la orden

Los corchetes indican el comienzo y el final de “main”

Final de línea

Retorna cero (el main es una función de tipo entero.)

Introducción - Estructura de un programa

```
#include <iostream>
```

Instrucciones del preprocesador

```
#include <stdio.h>
```

```
using namespace std;
```

```
/* Programa ejemplo */
```

Comentario

```
// This is line comment in C++
```

```
int main()
```

Nombre de la función

```
{
```

```
    int num;
```

Sentencia de declaración

```
    num=1;
```

Sentencia de asignación

```
    cout << "numero es " << num << endl;
```

Sentencia de función

```
    return 0;
```

Uso de tabuladores

Sentencias acaban en ;

```
}
```

Uso de corchetes

Introducción - COMPILACIÓN

–MS-DOS / Windows

- | | |
|--------------------|---------------------------------|
| • Editor de textos | 1. Código fuente (programa.cpp) |
| • Compilador | 2. Código objeto (programa.o) |
| • Linker | 3. Ejecutable (programa.exe) |

–UNIX

- | | |
|--------------------|---------------------------------|
| • Editor de textos | 1. Código fuente (programa.cpp) |
| • Compilador | 2. Código objeto (programa.o) |
| • Linker | 3. Ejecutable (a.out) |

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Algoritmo:

Mostrar “¡Hola
Mundo!”

Programa C++:

```
#include <iostream>

using namespace std;
int main()
{
    cout<<"¡Hola Mundo!";

    return 0;
}
```

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Programa C++:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"¡Hola Mundo!";

    return 0;
}
```

Incluye declaraciones de la
librería estándar de
entrada/salida

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Las llaves marcan el **comienzo** y **fin** de un bloque de instrucciones

Programa C++:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"¡Hola Mundo!";

    return 0;
}
```

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Instrucción (**llamada a función**)
para escribir "¡Hola Mundo!"

Programa C++:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"¡Hola Mundo!";

    return 0;
}
```

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Las “Sentencias” (líneas de instrucciones) siempre terminan en **punto y coma (;)**

Programa C++:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"¡Hola mundo!";

    return 0;
}
```

Estructura Básica de un Programa en C++

Ejemplo: Hola Mundo

Programa C++:

Devolver un cero (al ser
main una función de tipo
entero)

```
#include <stdio.h>
using namespace std;
int main()
{
    cout<<"¡Hola Mundo!";
    return 0;
}
```


Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

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

```
int main()  
{
```

```
    return 0;  
}
```

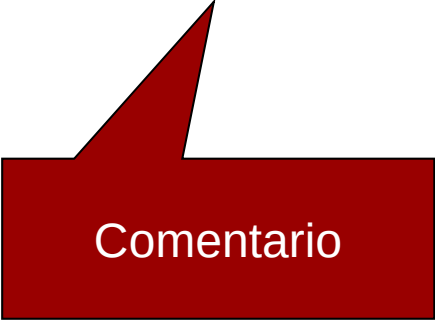
Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Mostrar los números del 0
al 9 */
```

```
int main()
{
```



Comentario

```
  return 0;
}
```

Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    return 0;
}
```



Declaración de
variable

Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    cuenta = 0;
    while ( cuenta < 10 )
    {
        cout<<cuenta<<endl;
        cuenta=cuenta+1;
    }
    return 0;
}
```

Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Asignación de un valor
(expresión derecha) a una
variable (izquierda).

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    cuenta = 0;
    while ( cuenta < 10 )
    {
        cout<<cuenta<<endl;
        cuenta=cuenta+1;
    }
    return 0;
}
```

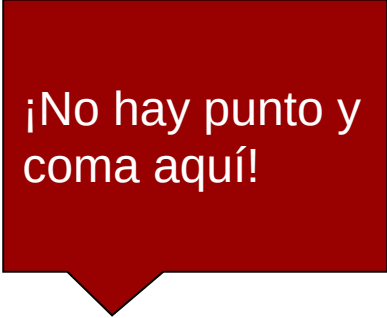
Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    cuenta = 0;
    while ( cuenta < 10 )
    {
        cout<<cuenta<<endl;
        cuenta=cuenta+1;
    }
    return 0;
}
```



Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    cuenta = 0;
    while ( cuenta < 10 )
    {
        cout<<cuenta<<endl;
        cuenta=cuenta+1;
    }
    return 0;
}
```

Cadena de formato

Ejemplo: Contar hasta 10

Mostrar los números
del 0 al 9

Fijar la cuenta a 0
Mientras que (la cuenta
sea menor que 10)
{
 mostrar cuenta
 sumar 1 a la cuenta
}

```
#include <iostream>
using namespace std;
/* Print out numbers 0 to 9 */
int main()
{
    int cuenta;

    cuenta = 0;
    while ( cuenta < 10 )
    {
        cout<<cuenta<<endl;
        cuenta=cuenta+1;
    }
    return 0;
}
```

Ejemplo: ¿Cuál es el signo?

Find the sign of a number

mostrar "Introduzca un
número"
introducir num

si (num es menos que 0)
entonces

```
{  
    mostrar num " es  
    negativo"  
}
```

sino

```
{  
    mostrar num " es  
    positivo"  
}
```

```
}
```

```
#include <iostream>  
using namespace std;  
/* Find the sign of a number */  
int main()  
{  
    float num;  
    cout<<"Introduzca un número: ";  
    cin>>num;  
    if ( num < 0 )  
    {  
        cout<<num<<" es negativo\n";  
    }  
    else  
    {  
        cout<<num<<" es positivo\n";  
    }  
    return 0;  
}
```

2. Tipos de datos

Tipos básicos

- ***bool***
- ***char*** caracteres
- ***int*** enteros
- ***float*** decimales
- ***short*** enteros “cortos”
- ***long*** enteros “largos”
- ***double*** decimales con doble precisión

Tipos

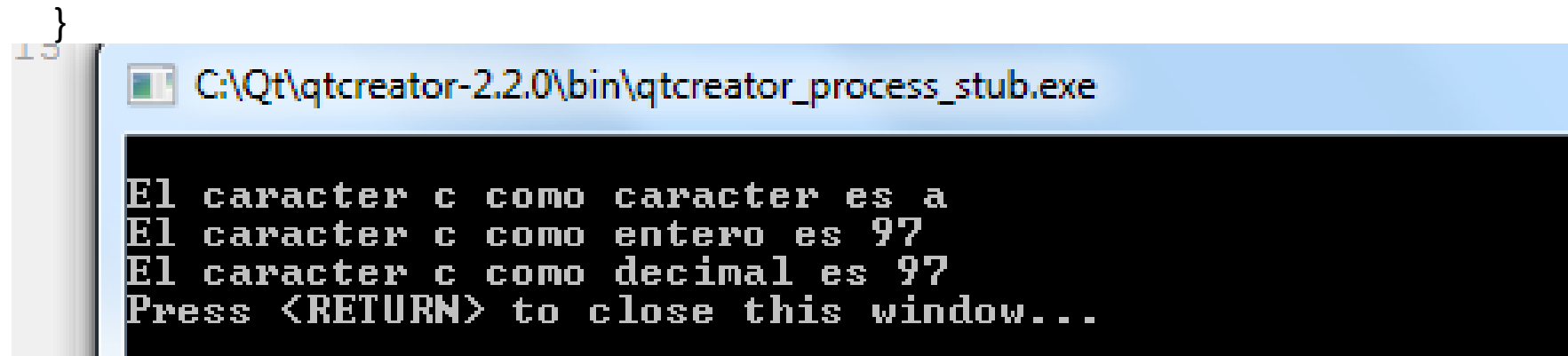
- Es importante elegir bien el tipo de variable: un int no almacena decimales y un float no permite resto entero.
- Es importante usar para cada variable, su código de conversión correspondiente en las funciones.

Representación y conversión de una variable

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    c = 'a';
    cout<<"\nEl caracter c como caracter es "<<c;
    cout<<"\nEl caracter c como entero es "<<(int)c;
    cout<<"\nEl caracter c como decimal es "<<(float)c;
    cout<<endl;
    return 0;
}
```

Representación y conversión de una variable

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    c = 'a';
    cout<<"\nEl caracter c como caracter es "<<c;
    cout<<"\nEl caracter c como entero es "<<(int)c;
    cout<<"\nEl caracter c como decimal es "<<(float)c;
    cout<<endl;
    return 0;
}
```

A screenshot of a Qt Creator console window. The title bar shows the file path "C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe". The console output is as follows:
El caracter c como caracter es a
El caracter c como entero es 97
El caracter c como decimal es 97
Press <RETURN> to close this window...
The text is displayed in a monospaced font on a black background.

Representación y conversión de una variable

```
1  #include <iostream>
2
3  using namespace std;
4  int main()
5  {
6      float c;
7      c = 98.0;
8      cout<<"\nEl caracter c como caracter es "<<(char)c;
9      cout<<"\nEl caracter c como entero es "<<(int)c;
10     cout<<"\nEl caracter c como decimal es "<<c;
11     cout<<endl;
12
13     return 0;
14 }
15
```

C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe

```
El caracter c como caracter es b
El caracter c como entero es 98
El caracter c como decimal es 98
Press <RETURN> to close this window...
```

Representación y conversión de una variable

```
1  #include <iostream>
2
3  using namespace std;
4  int main()
5  {
6      int c;
7      c = 98;
8      cout<<"\nEl caracter c como caracter es "<<(char)c;
9      cout<<"\nEl caracter c como entero es "<<c;
10     cout<<"\nEl caracter c como decimal es "<<(float)c;
11     cout<<endl;
12
13     return 0;
14 }
15
```

C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe

```
El caracter c como caracter es h
El caracter c como entero es 98
El caracter c como decimal es 98
Press <RETURN> to close this window...
```

Tipos básicos

Tamaños de los tipos

	Ms-Dos	Unix
– bool	8 bits	8 bits
– char	8 bits	8 bits
– int	16bits	32bits
– float	32bits	32bits
– short	16bits	16bits
– long	32bits	32bits
– double	64bits	64bits



Rangos de los tipos básicos

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 or 4 bytes	1 wide character

- Tabla obtenida de <http://www.cplusplus.com/doc/tutorial/variables/>
- Los datos de las columnas *Size* y *Range* dependen del sistema y la arquitectura para la que se están compilando. Los valores aquí presentados son los más comunes para arquitecturas de 32-bits.

Tipos básicos

```
#include <iostream>

using namespace std;
/* Programa ejemplo */
int main()
{
    int num, Num;

    num=1;

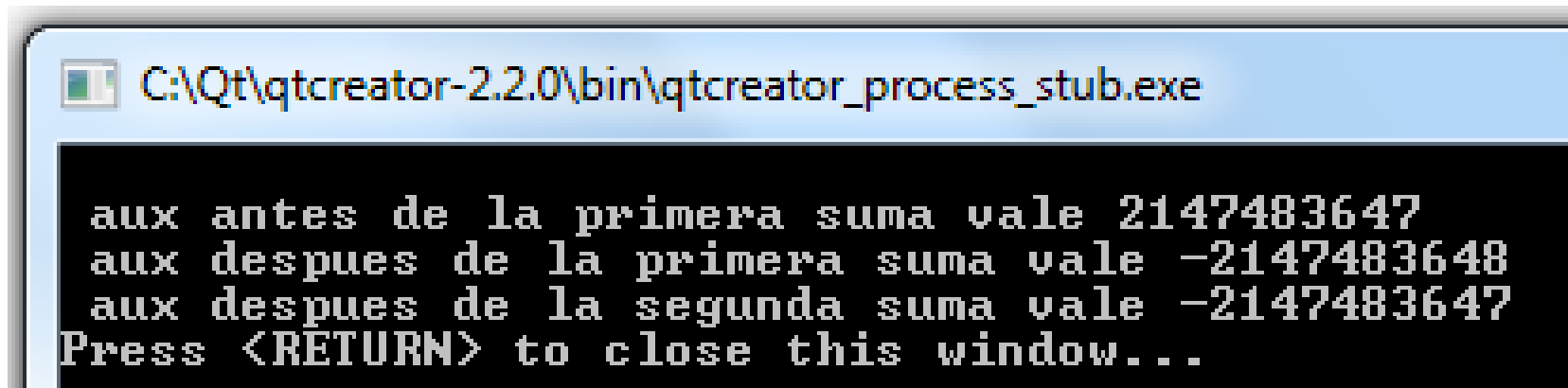
    cout<<"numero es "<<num<<endl;

    return 0;
}
```

- Si un entero es 4 bytes (32 bits)
 - Desde $-2^{31} = -2.147.483.648$
 - Hasta $2^{31}-1 = 2.147.483.647$
 - » (por el cero)
- **OJO** con el desbordamiento
- **OJO** con las mayúsculas

Rangos de los tipos

```
#include<iostream>
using namespace std;
int main()
{
    int aux=2147483647;
    cout << "\n aux antes de la primera suma vale " << aux;
    aux= aux+1;
    cout << "\n aux despues de la primera suma vale " << aux;
    aux= aux+1;
    cout << "\n aux despues de la segunda suma vale " << aux << endl;
    return 0;
}
```



The screenshot shows a console window titled "C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe". The output text is as follows:

```
aux antes de la primera suma vale 2147483647
aux despues de la primera suma vale -2147483648
aux despues de la segunda suma vale -2147483647
Press <RETURN> to close this window...
```

Rangos de los tipos

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

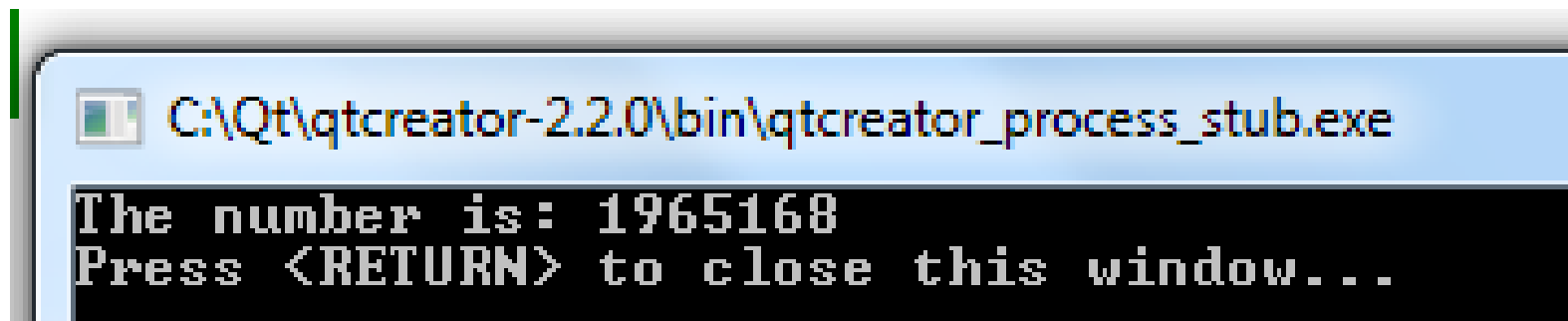
```
    int num, Num;
```

```
    num = 1 ;
```

```
    cout << "The number is: " << Num << endl ;
```

```
    return 0;
```

```
}
```



The screenshot shows a console window titled "C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe". The output text is "The number is: 1965168" followed by "Press <RETURN> to close this window...".

tipo *char*

- Sirve para:
 - caracteres
 - enteros: -128/127
- Los valores son los definidos por el código ASCII
 - 'a' vale 97
 - 'A' vale 65
 - '2' vale 50 **no** 2

Constantes

- Enteros/Decimales
 - 42 char/int
 - 42L long int
 - 4.2F float
- Si empieza por cero = octal
 - 042 es realmente 34: $4 \cdot 8 + 2$.
- Si empieza por 0x
=hexadecimal
 - 0x42 es realmente 66: $4 \cdot 16 + 2$.

Sizeof

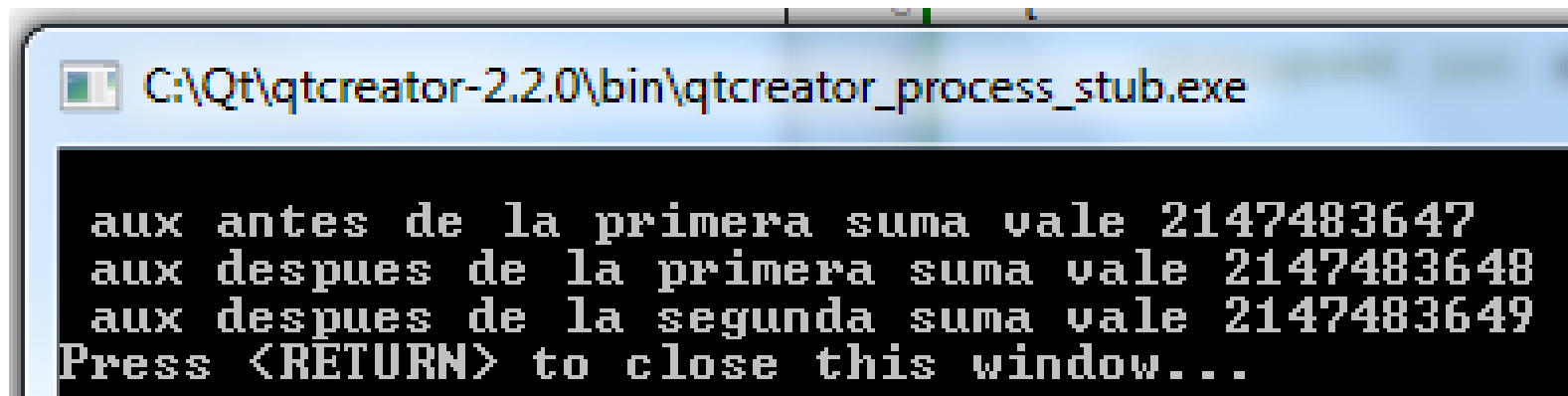
- Los tamaños de los enteros dependen:
 - Del Sistema Operativo
 - Del compilador
- sizeof() devuelve el número de bytes
 - sizeof(int)

UNSIGNED

- Indica que todos los números van a ser positivos
- Se aumenta el rango de la variable
- Solo tenemos número positivos
 - unsigned char a a=0-255
 - unsigned int a a=0-4.294.967.295
 - unsigned short int a
 - unsigned long int a

UNSIGNED

```
#include<iostream>
using namespace std;
int main()
{
    unsigned int aux=2147483647;
    cout << "\n aux antes de la primera suma vale " << aux;
    aux= aux+1;
    cout << "\n aux despues de la primera suma vale " << aux;
    aux= aux+1;
    cout << "\n aux despues de la segunda suma vale " << aux << endl;
    return 0;
}
```



The screenshot shows a console window titled "C:\Qt\qtcreator-2.2.0\bin\qtcreator_process_stub.exe". The output text is as follows:

```
aux antes de la primera suma vale 2147483647
aux despues de la primera suma vale 2147483648
aux despues de la segunda suma vale 2147483649
Press <RETURN> to close this window...
```

Conversiones y Cast

- Si en una expresión existen variables de distinto tipo se convierten *automáticamente* al tipo mas amplio.

char ☒ short ☒ int ☒ long ☒
float double

- Con el signo de igualdad el resultado se convierte al tipo de la variable izda ☒

Si *i* es variable entera $i=3/9.0$ i vale 0

- Un *cast* es una conversión explícita ☒

Aunque *f* sea un float, $f=3/9$ ☒ $f=0$

$f=3/(\text{float})9$ $f=3/9.0=0.333$

Variables

- Entidades que contienen los valores.
- Todas las variables deben declararse antes de usarse
 - Informar al compilador
 - Reservar memoria
- Los nombres pueden ser de hasta 254 caracteres
 - Sólo los 31 primeros se usan
- Se pueden utilizar A-Z a-z 0-9 y _
- Deben empezar por un carácter alfabético

Declaración de variables

- Es la línea en la que se pone el tipo de variable, y luego el nombre de la variable.
 - `int i;`
- Se puede declarar más de una variable del mismo tipo en una misma línea, separándolas por comas.

Declaración de variables

```
int a;
```

```
int number=2, Number=3;
```

```
float Number2;
```

```
Number2=3/9;
```

```
char letra;
```

```
letra='a';
```

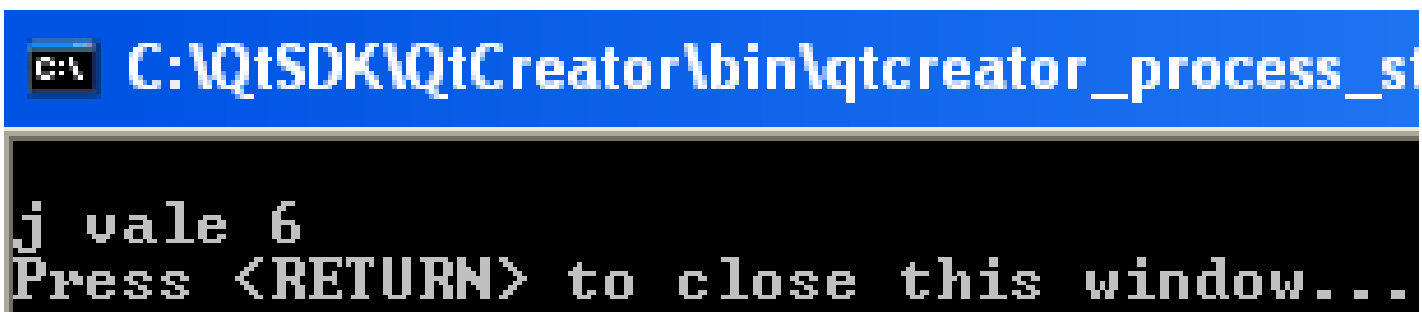
```
char letra2='v';
```

Declaración de variables

```
#include <iostream>

using namespace std;

int main()
{
    int i, j;
    i = 2;
    j = 3 * i;
    cout << "\n j vale " << j << endl;
    return 0;
}
```



The screenshot shows a console window with a blue title bar containing the text "C:\QtSDK\QtCreator\bin\qtcreator_process_s". The console output displays "j vale 6" followed by the prompt "Press <RETURN> to close this window...".

Nombres reservados

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

Atributos de las variables

- Tipo. Cuando se declara: `int i;`
- Alcance. Parte del programa donde puede ser utilizada.
 - Dentro de una función: desde la def. hasta el final de la función.
 - Fuera de una función: desde la def. hasta el final de fichero.
- Static/automatic
 - `int i; static int j;`
- Global/local
- Const: `const double pi = 3.14159265;`
- Register: Optimización
- Volatile: interrupciones

Ejemplo de variable *static*

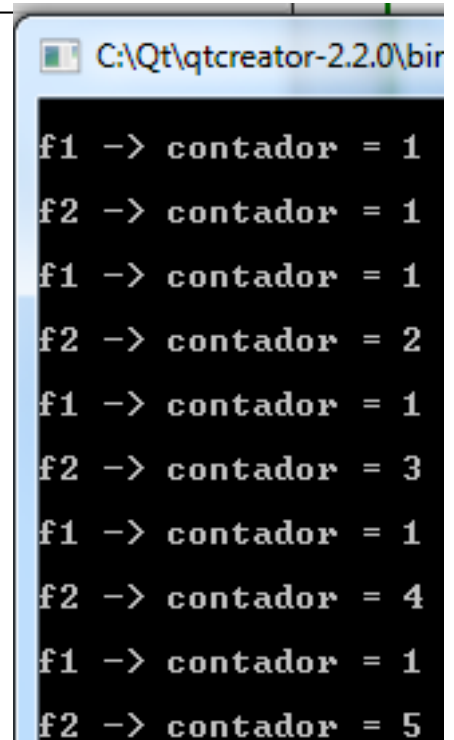
```
#include <iostream>
using namespace std;
void f1(void)
{
    int contador=0;
    cout << "\nf1 -> contador = " << ++contador << endl;
}

void f2(void)
{
    static int contador=0;
    cout << "\nf2 -> contador = " << ++contador << endl;
}
```

...

...

```
int main()
{
    for(int i = 0; i < 5; i++ )
    {
        f1();
        f2();
    }
}
```



```
C:\Qt\qtcreator-2.2.0\bin
f1 -> contador = 1
f2 -> contador = 1
f1 -> contador = 1
f2 -> contador = 2
f1 -> contador = 1
f2 -> contador = 3
f1 -> contador = 1
f2 -> contador = 4
f1 -> contador = 1
f2 -> contador = 5
```

¿Cuál es el resultado?

3. Operadores matemáticos

Operadores matemáticos

- Suma +
- Resta -
- Multiplicación *
- División /
- Resto (enteros) %

Operadores matemáticos

- Precedencia
 - Multiplicación, división y resto $\succ$ suma y resta
 - Multiplicación = división = resto
 - suma = resta | multiplicación = división = resto
 - Criterio: Primera operación por la izquierda
 - Uso de paréntesis:
 - $5+4*3=5+12=17$
 - $(5+4)*3=9*3=27$

Operadores matemáticos

- Precedencia y tipos

int i;

i=5*32/9; i=160/9=17

i=5/9*32; i=0*32=0

i=5/9.0*32; i=0.555*32=17

Operadores matemáticos

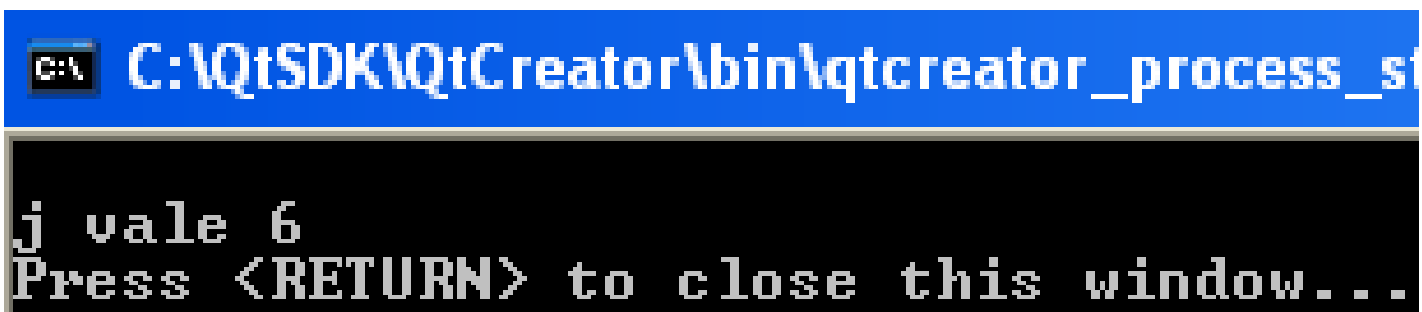
- Se pueden aplicar sobre constantes
- También se pueden aplicar sobre variables
- Se pueden combinar ambos.

Operadores matemáticos

```
#include <iostream>

using namespace std;

int main()
{
    int i, j;
    i = 2;
    j = 3 * i;
    cout << "\n j vale " << j << endl;
    return 0;
}
```



The screenshot shows a console window with a blue title bar containing the text "C:\QtSDK\QtCreator\bin\qtcreator_process_s". The console output displays "j vale 6" followed by the prompt "Press <RETURN> to close this window...".

Operadores ++ --

++i es equivalente a $i=i+1$

i++ es equivalente a $i=i+1$

(i+j)++ ***es ilegal***

--i es equivalente a $i=i-1$

i-- es equivalente a $i=i-1$

(i+j)-- ***es ilegal***

- En expresiones:

++i hace primero la suma y luego asigna el valor

i++ asigna primero el valor y luego hace la suma

Operadores ++ --

```
#include <iostream>
using namespace std;
/* Programa ejemplo */
void main() {
    int a,b,c;
    a=b=c=0;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=++b + ++c;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=b++ + c++;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=b-- + --c;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}
```

	a	b	c
a=b=c=0;	0	0	0
a=++b + ++c;	?	?	?
a=b++ + c++;	?	?	?
a=b-- + --c;	?	?	?

Operadores ++ --

```
#include <iostream>
using namespace std;
/* Programa ejemplo */
void main() {
    int a,b,c;
    a=b=c=0;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=++b + ++c;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=b++ + c++;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
    a=b-- + --c;
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}
```

	a	b	c
a=b=c=0;	0	0	0
a=++b + ++c;	?	?	?
++b	0	1	0
++c	0	1	1
a=b+c;	2	1	1
a=b++ + c++;	?	?	?
a=b+c	2	1	1
b++	2	2	1
c++;	2	2	2
a=b-- + --c;	?	?	?
--c	2	2	1
a=b+c	3	2	1
b--;	3	1	1

Operadores de asignación

- $=$ $+=$ $-=$ $*=$ $/=$
- *variable operador de asignación expresión*

k += 2

- *variable = variable operador expresión*

k = k + 2

- Ej.:

k*=2

k*=3+x



Operadores de asignación

- $=$ $+=$ $-=$ $*=$ $/=$
- *variable operador de asignación expresión*

k += 2

- *variable = variable operador expresión*

k = k + 2

- Ej.:

k*=2 k=k*2

k*=3+x k=k*(3+x)

Operaciones con bits

- Complemento a uno $\sim$

$$\begin{array}{rcl} \sim & 00000101 & (5) \\ = & 11111010 & (250) \end{array}$$

- AND $\&$

$$\begin{array}{rcl} & 00000101 & (5) \\ & \underline{00000110} & (6) \\ \& & 00000100 & (4) \end{array}$$

- OR $|$

$$\begin{array}{rcl} & 00000101 & (5) \\ & \underline{00000110} & (6) \\ | & 00000111 & (7) \end{array}$$

- OR Exclusivo $\wedge$

$$\begin{array}{rcl} & 00000101 & (5) \\ & \underline{00000110} & (6) \\ \wedge & 00000011 & (3) \end{array}$$

Operaciones con bits

- Desplazamientos

- variable *desplazamiento* num_bits

i=5;

i=i << 2 (dos bits a la izda)

00000101 (5)

<<

2



i=i >> 2 (dos bits a la dcha)

00000101 (5)

>>

2



Operaciones con bits

- Desplazamientos

- variable *desplazamiento* num_bits

i=5;

i=i << 2 (dos bits a la izda)

00000101 (5)

<< 2

= 00010100 (20)

i=i >> 2 (dos bits a la dcha)

00000101 (5)

>> 2

= 00000001 (1)

Descarga de QT Creator



Download Qt, the cross-platform application framework — Qt - A cross-platform application and U - Windows Internet Explorer

http://qt.nokia.com/downloads

Google Qt creator

Favoritos Domiciliar Nomina, Cuentas, ... Portada Universidades Co... Sitios sugeridos Hotmail gratuito Más complementos

Qt Download Qt, the cross-p... iBienvenido a Facebook en E...

Google Esta página está escrita en inglés. ¿Quieres traducirla con la barra Google? Más información ¿Esta página no está escrita en inglés? Ayúdanos a mejorar Traducir Desactivar traducción al inglés

Jump to: Qt libraries Qt Creator Visual Studio Add-in

 The Qt SDK includes the tools you need to build desktop, embedded and mobile applications with Qt from a single install. This is the recommended way to get started with Qt. The latest SDK has a self updater feature that will keep you up to date as new versions are made available.

The Qt SDK version 1.1.3 released on September 1st contains:

Qt libraries version 4.7.4	Symbian and MeeGo simulators
Qt Creator IDE version 2.3	Qt Mobility version 1.2
Qt development tools	Remote compilers

Windows*:

 Online installer - 15 MB	 Offline installer - 1.4 GB
---	---

Linux/X11 32-bit:

 Online installer - 23 MB	 Offline installer - 1.2 GB
--	--

Linux/X11 64-bit:

 Online installer - 23 MB	 Offline installer - 1.2 GB
--	--

Mac OS X:

 Online installer - 12 MB	 Offline installer - 1.2 GB
--	--

What the Qt SDK means for Qt users

Sep 02, 2011

Qt SDK Update introducing Qt Creator 2.3 and other updates

Sep 01, 2011

Qt Creator 2.3.0 released

Sep 01, 2011

Qt 4.7.4 released

Sep 01, 2011

Qt Simulator 1.2: What's new?

Sep 01, 2011

More...

The Qt Blog

Release day! Qt SDK update, Qt 4.7.4, Qt Creator 2.3 and more

Sep 01, 2011

9 Million downloads a day and Belle. Why the opportunity with Nokia is growing for Qt developers

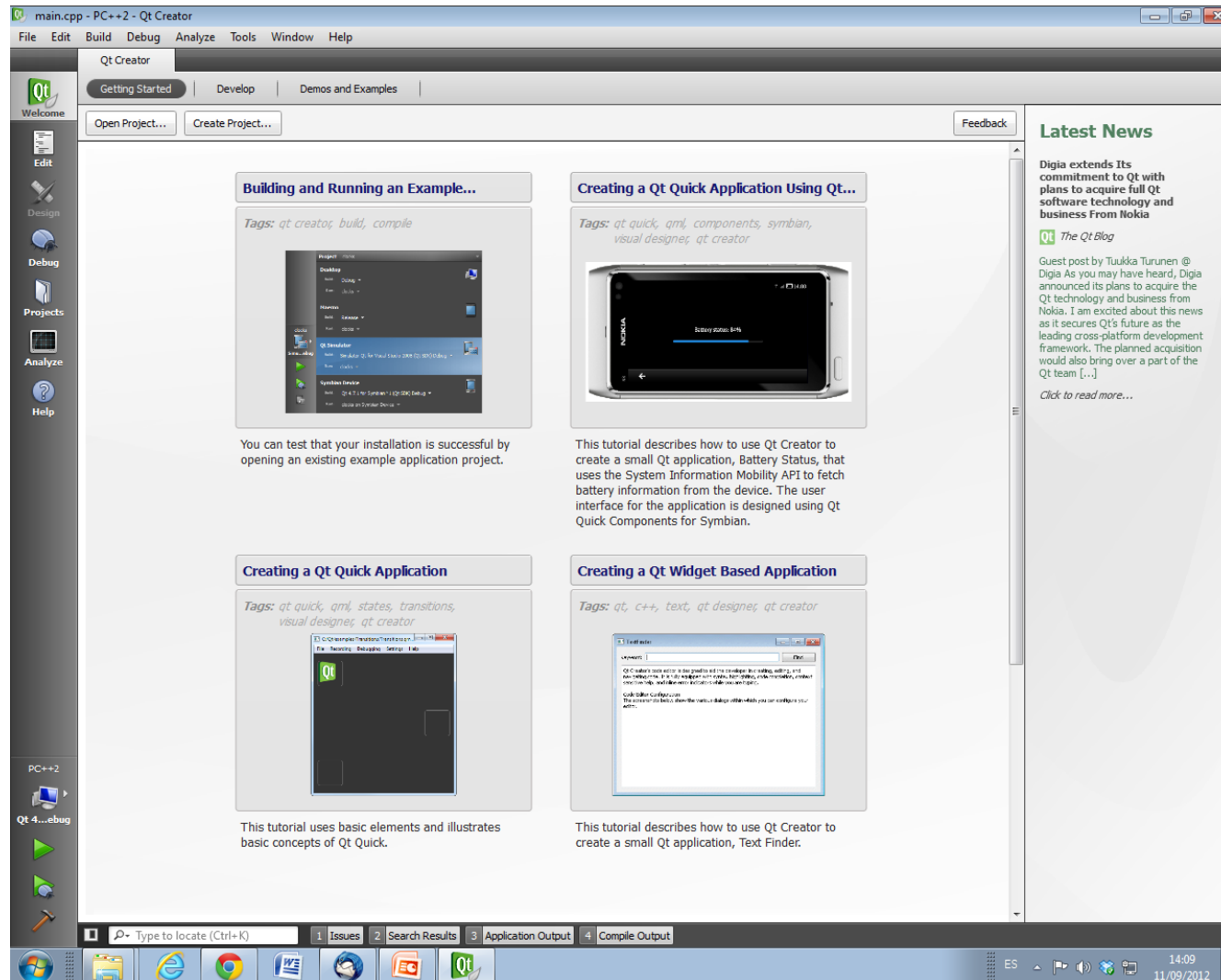
License

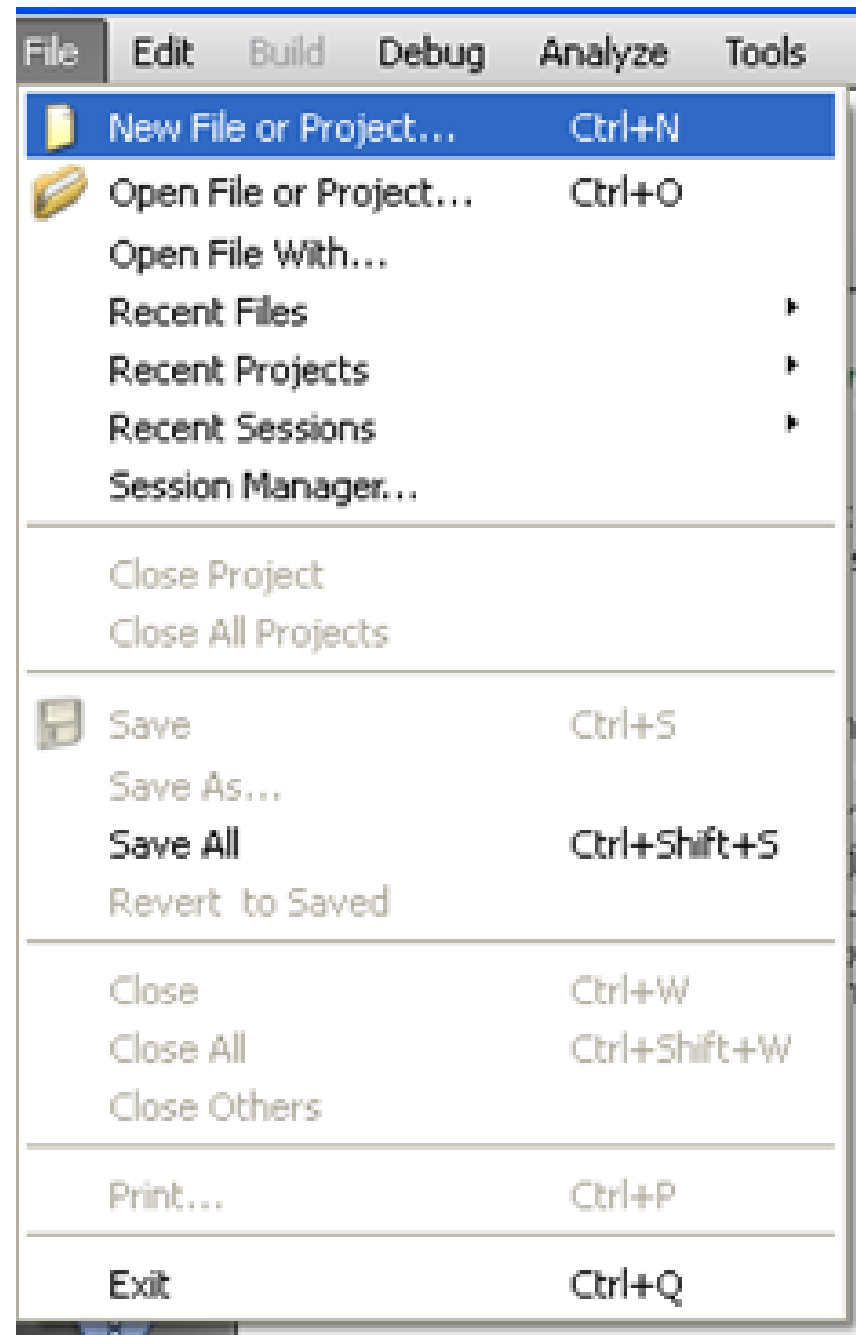
http://qt.nokia.com/downloads/sdk-windows-cpp

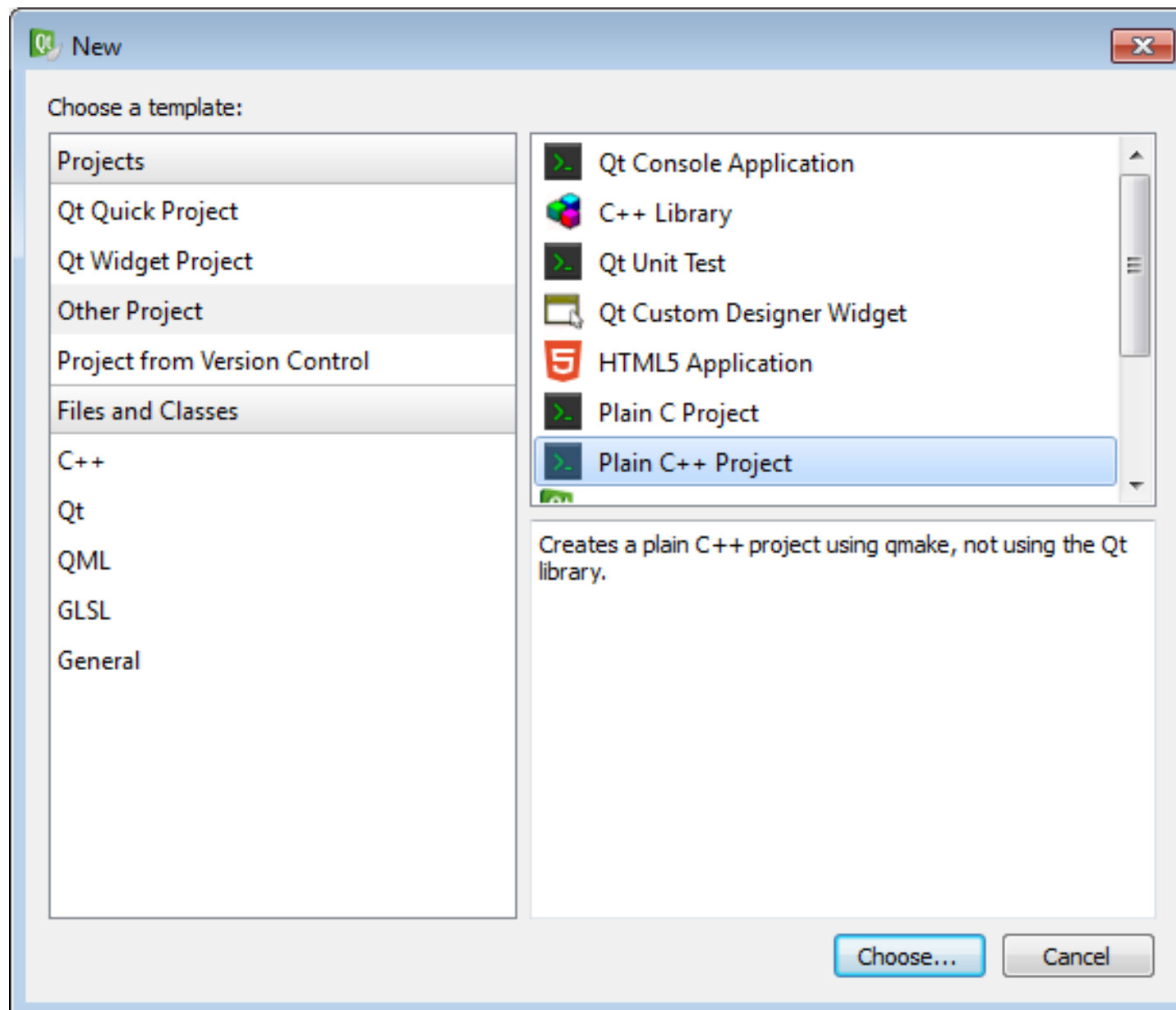
Internet 100%

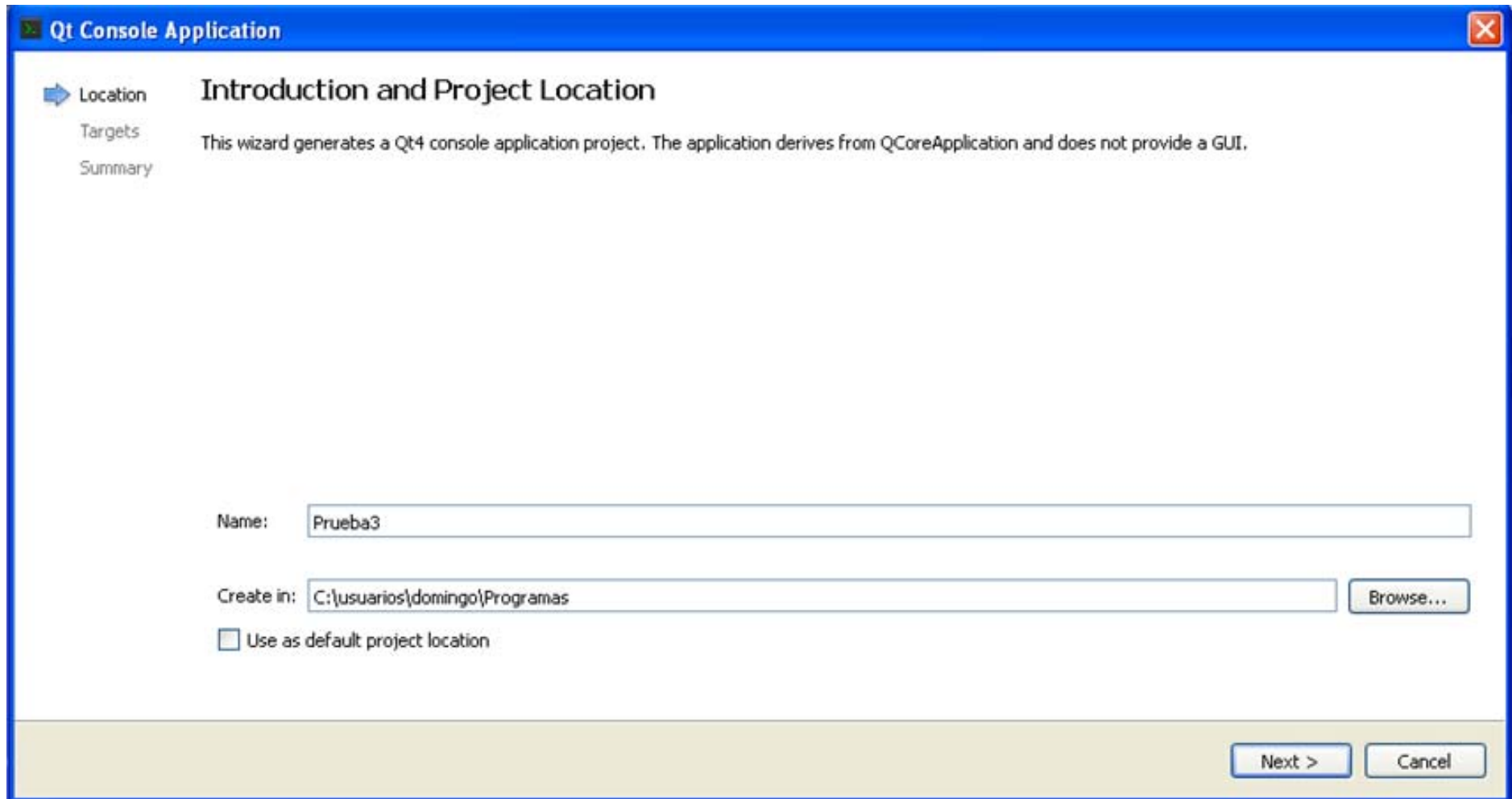
Inicio Microsoft PowerPoint ... main.cpp - Prueba3 - ... Documento1 - Micros... Download Qt, the cro... ES 18:52

Realización de un programa ejemplo en QT Creator









The image shows a Qt Console Application wizard window. The title bar is blue and contains the text "Qt Console Application" and a close button. The window has a sidebar on the left with three items: "Location" (selected), "Targets", and "Summary". The main area is titled "Introduction and Project Location" and contains the text: "This wizard generates a Qt4 console application project. The application derives from QApplication and does not provide a GUI." Below this text are three input fields: "Name:" with the value "Prueba3", "Create in:" with the value "C:\\usuarios\\domingo\\Programas", and a checkbox labeled "Use as default project location" which is unchecked. To the right of the "Create in:" field is a "Browse..." button. At the bottom right of the window are two buttons: "Next >" and "Cancel".

Qt Console Application

Location
Targets
Summary

Introduction and Project Location

This wizard generates a Qt4 console application project. The application derives from QApplication and does not provide a GUI.

Name:

Create in:

☐ Use as default project location

Target Setup

Qt Creator can set up the following targets for project **Prueba3**:

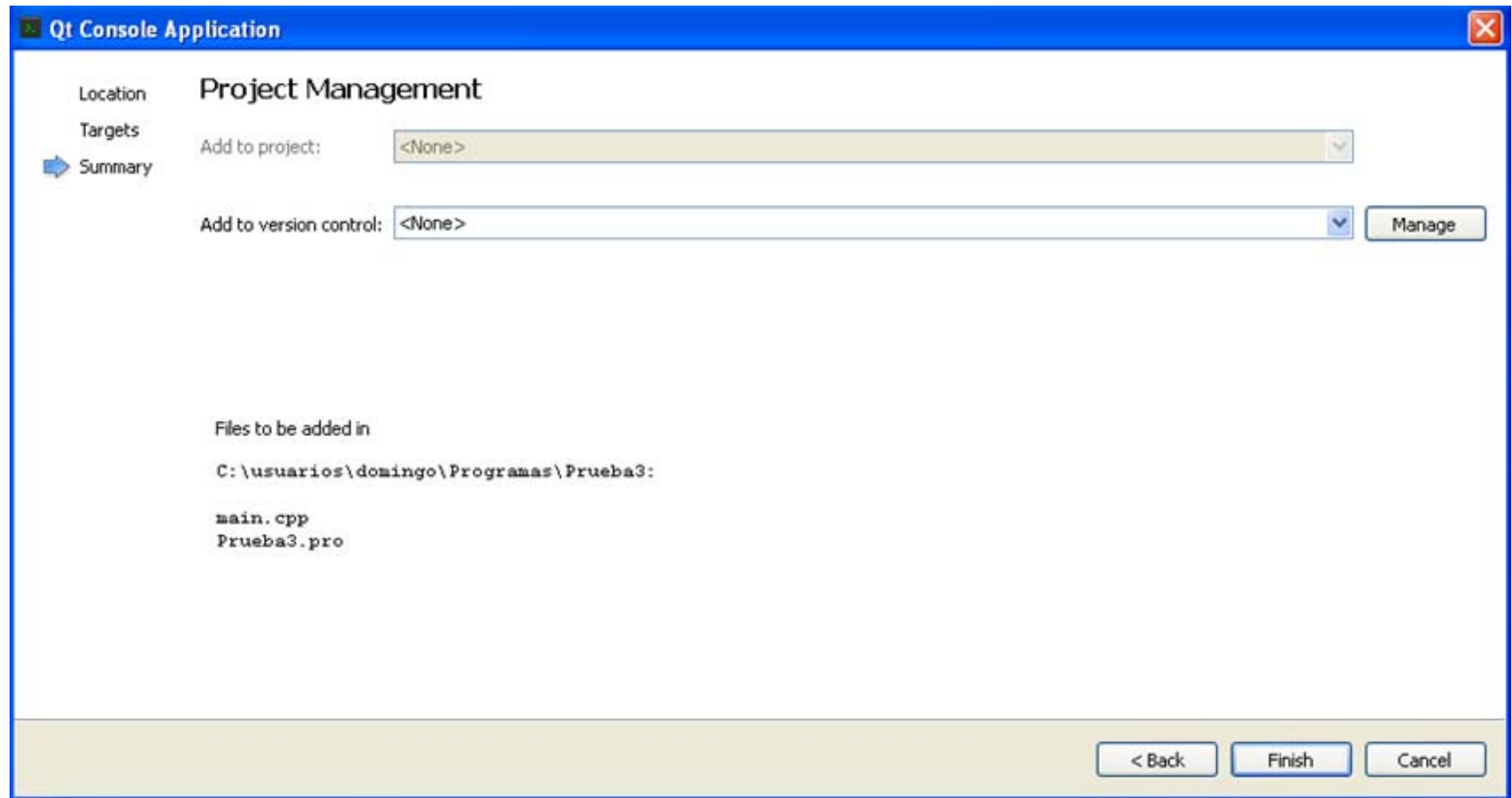
☐  **Symbian Device**

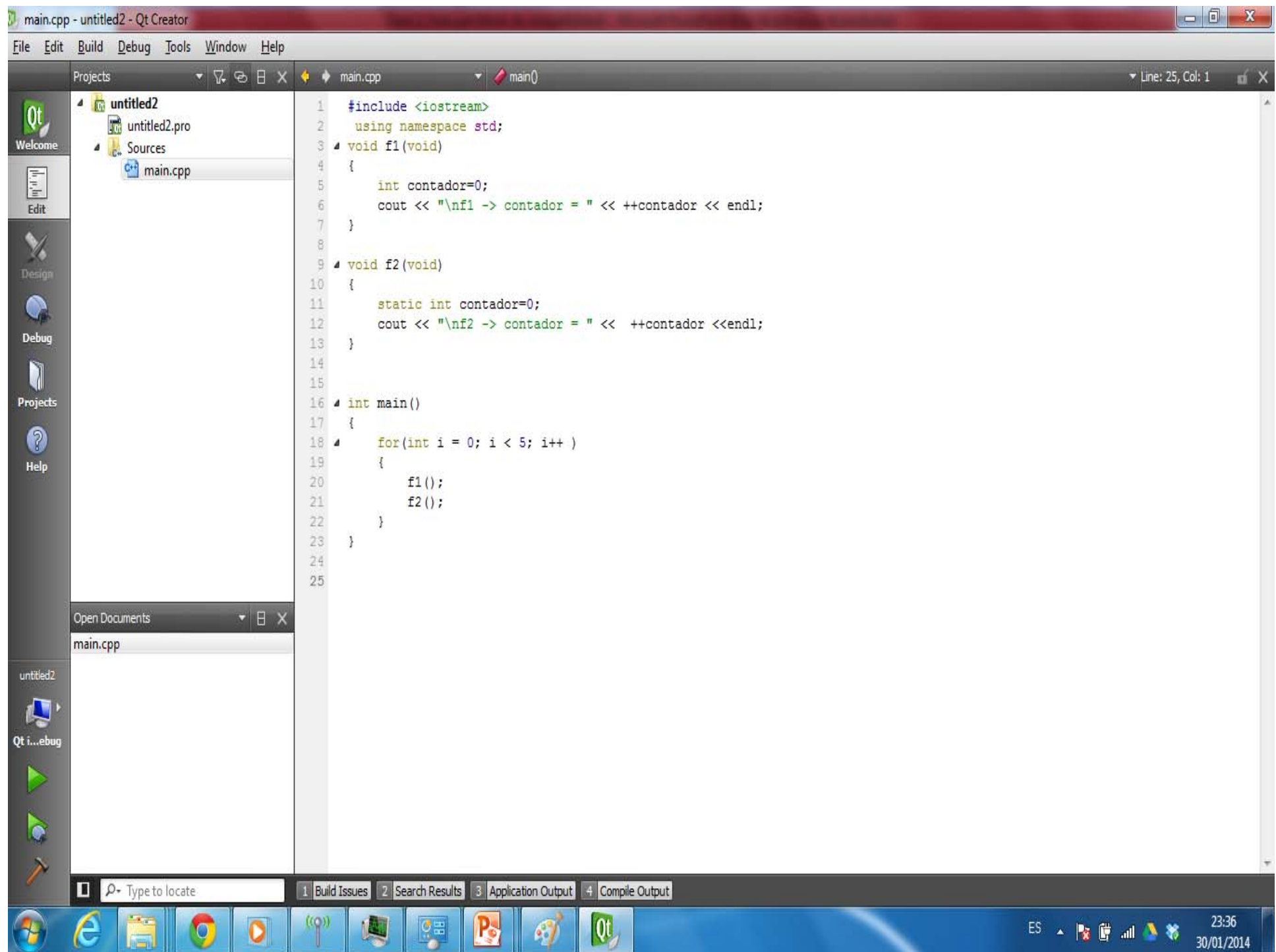
☒  **Desktop**

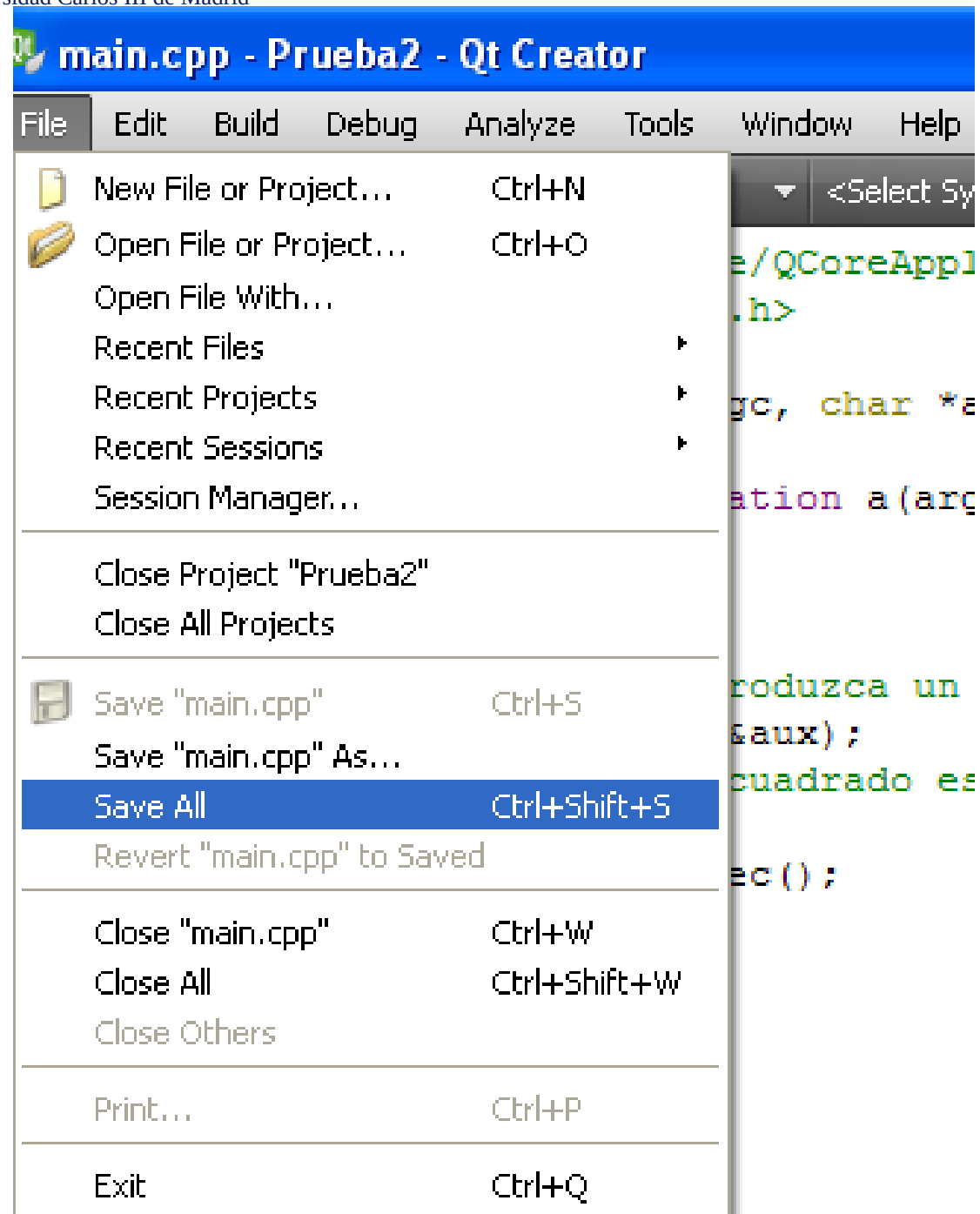
Create Build Configurations: For Each Qt Version One Debug And One Release 
☒ Use Shadow Building

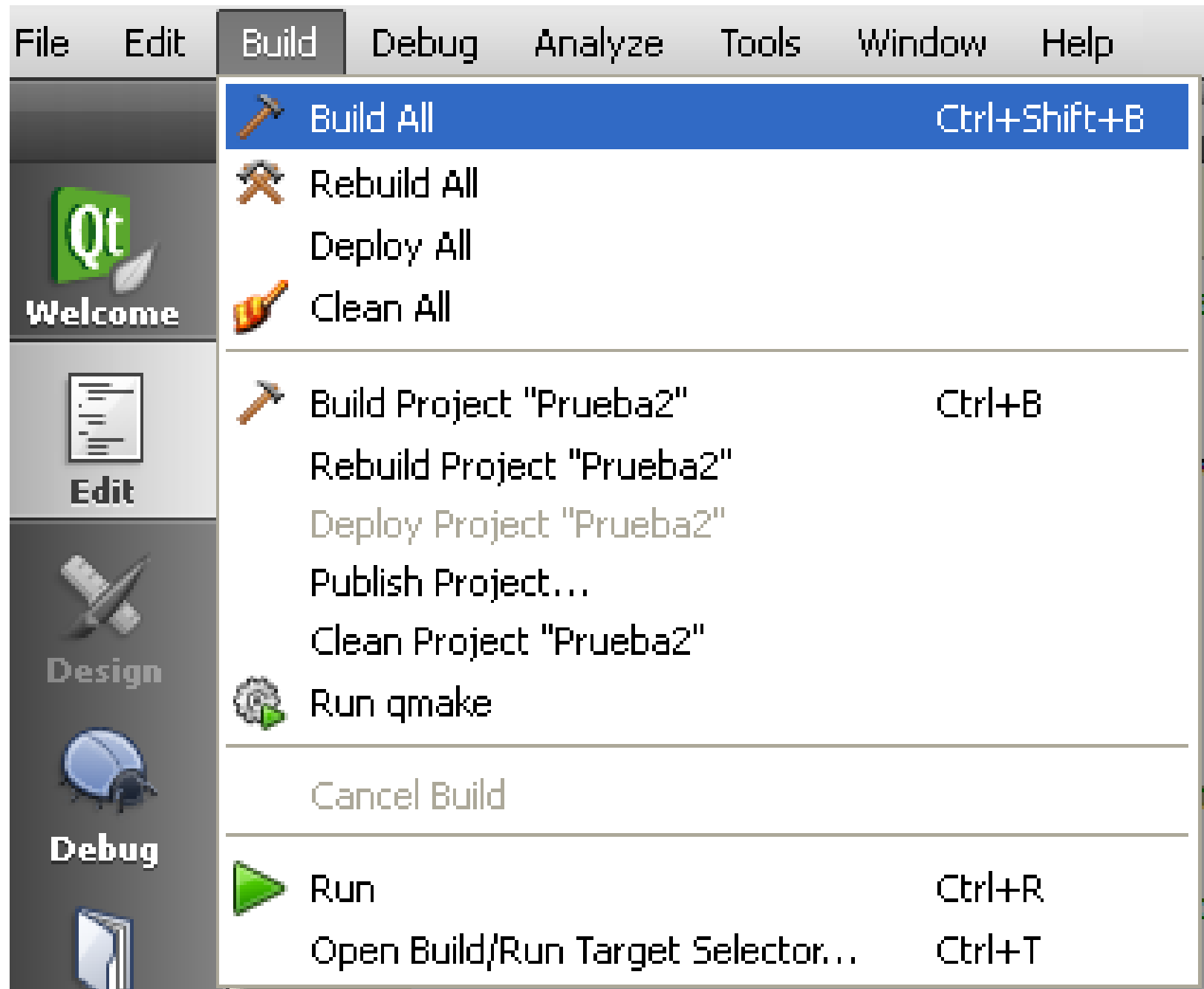
☐  **Qt Simulator**

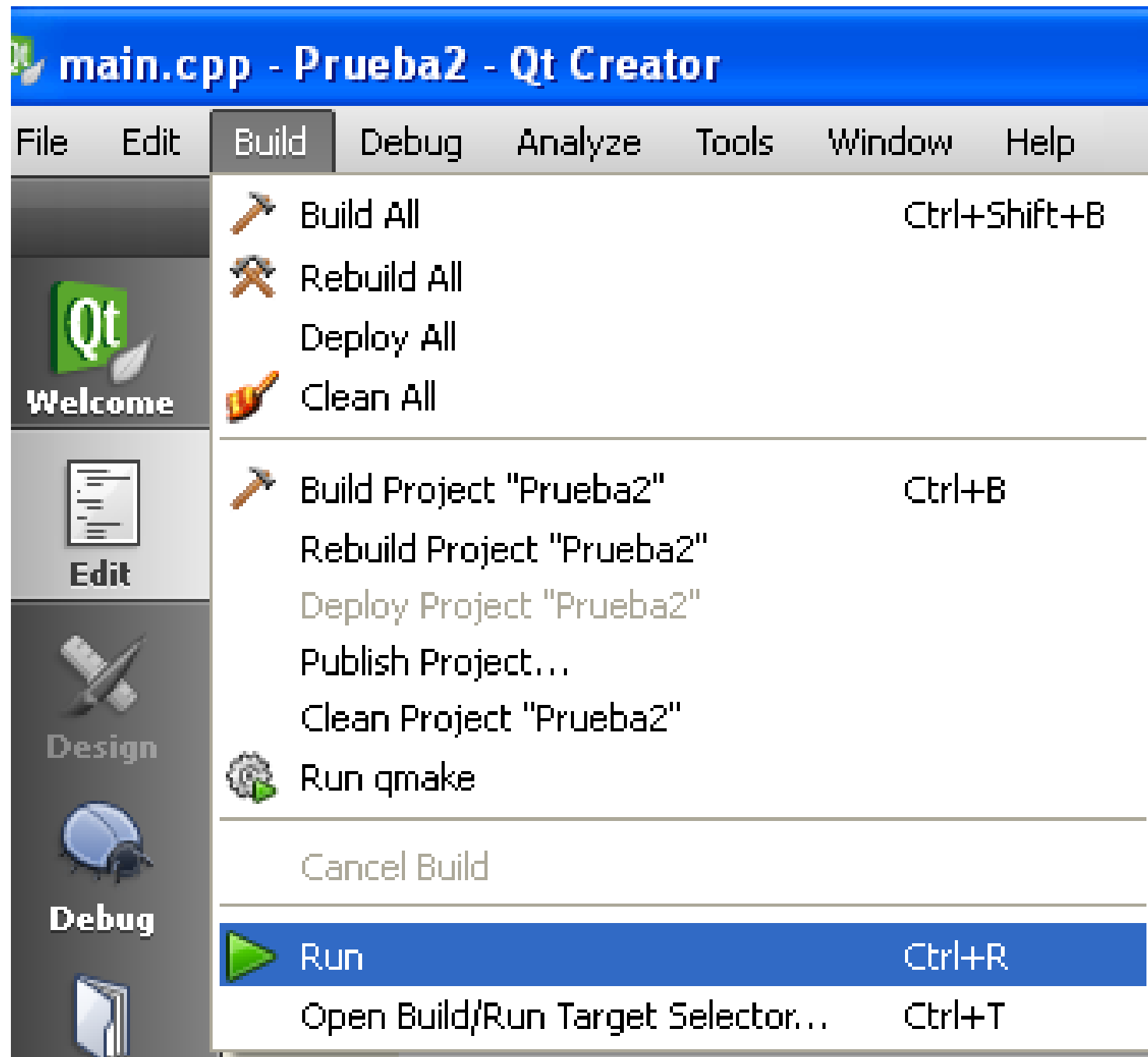
☐  **Harmattan**

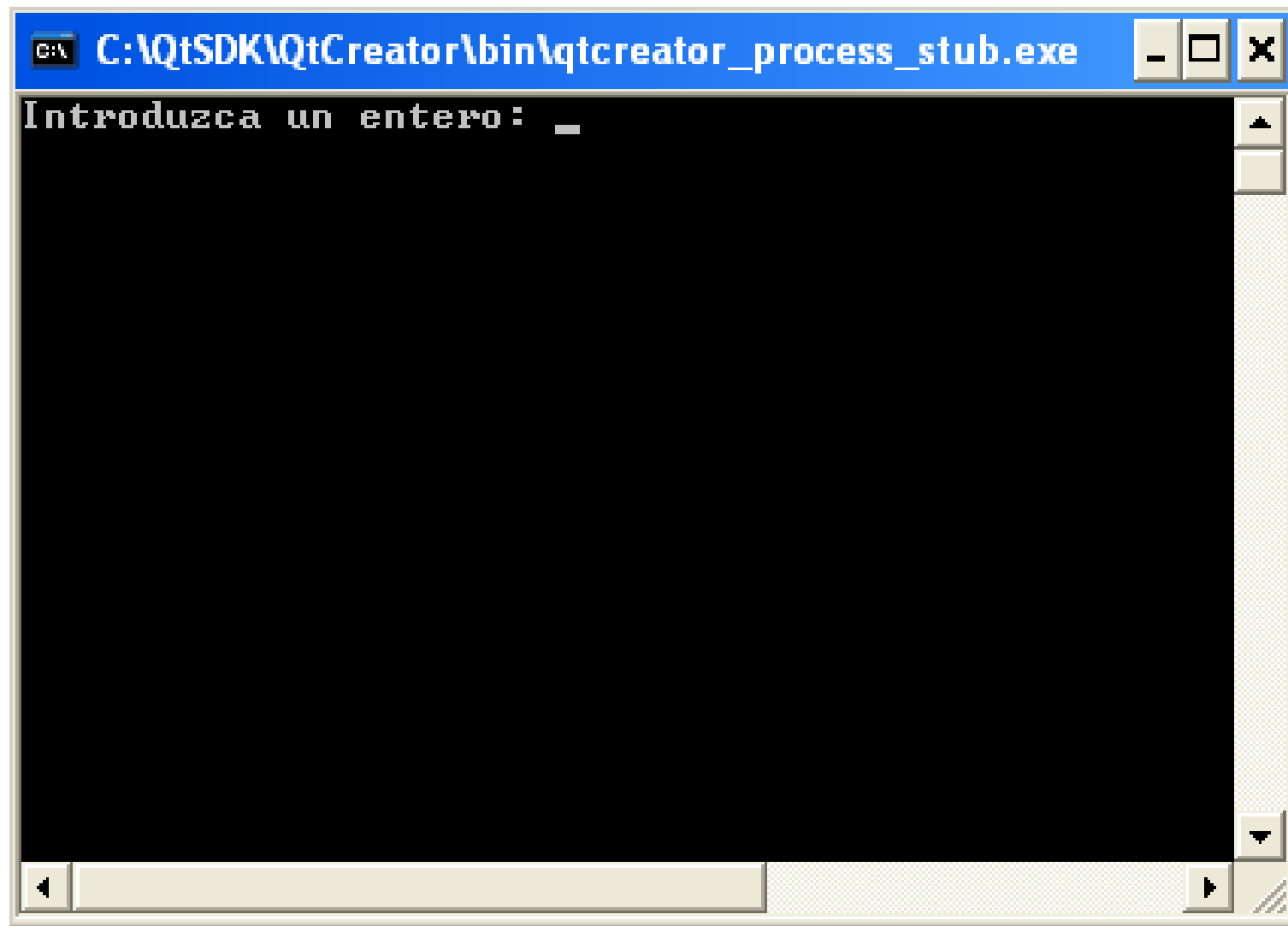


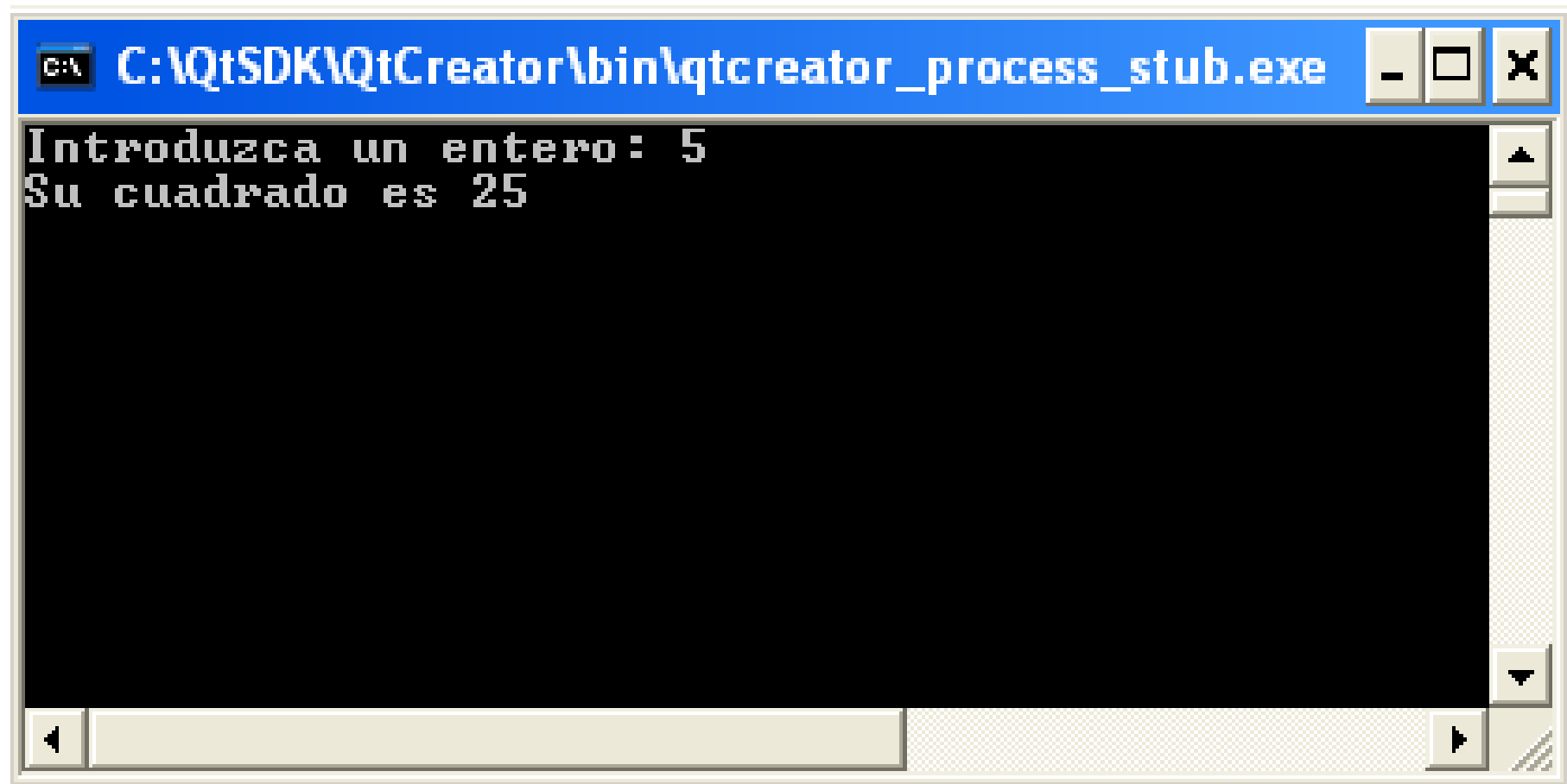












Problemas típicos

- Si dice que “cout” no ha sido declarada, probablemente se se puede haber olvidado incluir la librería “**iostream**” y la línea “**using namespace std;**”.
- Si dice que falta un “;” antes de una función, probablemente se puede haber olvidado ponerlo al final de la anterior.
- Si falta una llave, { ó }, en un sitio, se puede haber olvidado bastante atrás.