INFORMÁTICA INDUSTRIAL

PROGRAMACIÓN BÁSICA C++

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid

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

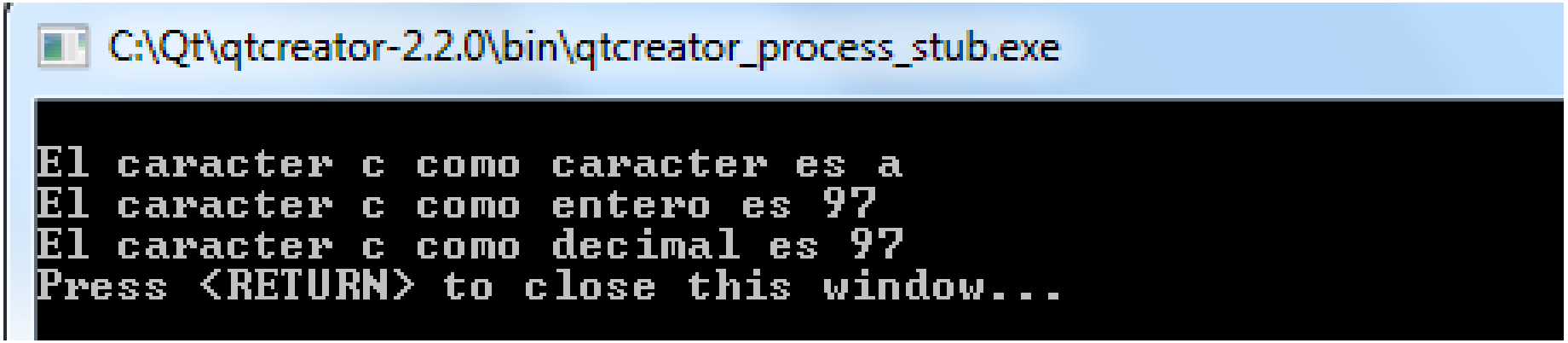
- Es importante elegir bien el tipo de variable: un int no almacena decimales y un float no permite resto entero.
- Es importante usar para cada variable, su código de conversión correspondiente en las funciones.

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;
 }
```



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

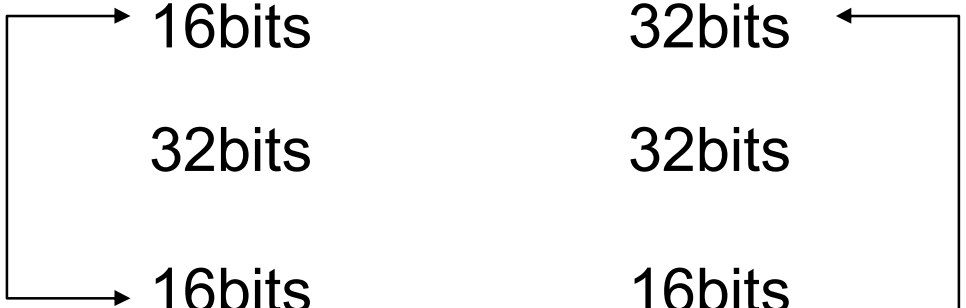
```
El caracter c como caracter es b
El caracter c como entero es 98
El caracter c como decimal es 98
Press <RETURN> to close this window...
```

Tipos básicos

Tamaños de los tipos

- bool
- char
- int
- float
- short
- long
- double

	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

Tipo	bits	Rango / Tipo de uso
unsigned char	8	0 <= X <= 255 Números pequeños y juego caracteres del PC.
char (signed)	8	-128 <= X <= 127 Números muy pequeños y juego de caracteres ASCII
short (signed)	16	-32,768 <= X <= 32,767 Números muy pequeños, control de bucles pequeños
unsigned short	16	0 <= X <= 65,535 Números muy pequeños, control de bucles pequeños
unsigned (int)	32	0 <= X <= 4,294,967,295. Números grandes
int (signed)	32	-2,147,483,648 <= X <= 2,147,483,647 Números pequeños, control de bucles
unsigned long	32	0 <= X <= 4,294,967,295 Distancias astronómicas
enum	32	-2,147,483,648 <= X <= 2,147,483,647 Conjuntos de valores ordenados
long (int)	32	-2,147,483,648 <= X <= 2,147,483,647 Números grandes
float	32	1.18e-38 <= |X| <= 3.40e38 Precisión científica (7-dígitos)
double	64	2.23e-308 <= |X| <= 1.79e308 Precisión científica (15-dígitos)
long double	80	3.37e-4932 <= |X| <= 1.18e4932 Precisión científica (18-dígitos)

- 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

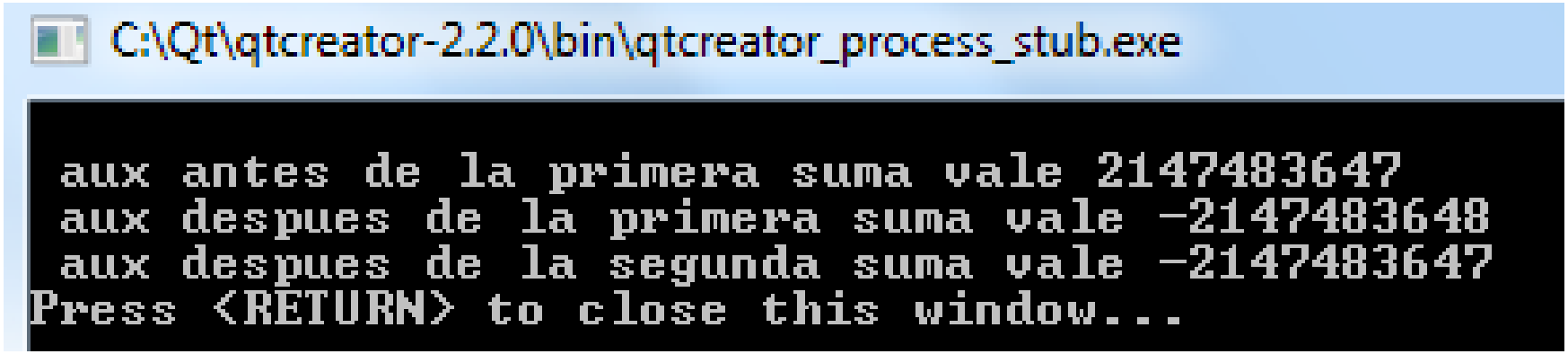
- Dependen del Sistema Operativo y el compilador

```
#include <limits>
#include <iostream>

int main()
{
    std::cout   << "el mayor float es ”
                <<std::numeric_limits<float>::max()
                << ",el menor float es “
                <<std::numeric_limits<float>::min()
                << ",los char tienen signo?“
                <<std::numeric_limits<char>::is_signed
                << "\n";
}
```

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;
}
```

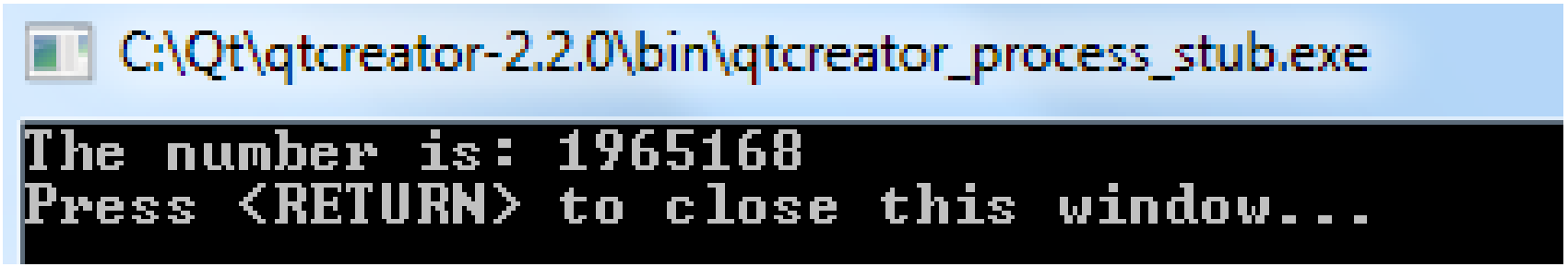


Rangos de los tipos

```
#include<iostream>

using namespace std;

int main()
{
        int num, Num;
        num = 1 ;
        cout << "The number is: " << Num << endl ;
        return 0;
}
```



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*8+2.
- Si empieza por 0x =hexadecimal
  - 0x42 es realmente 66: 4*16+2.

Sizeof

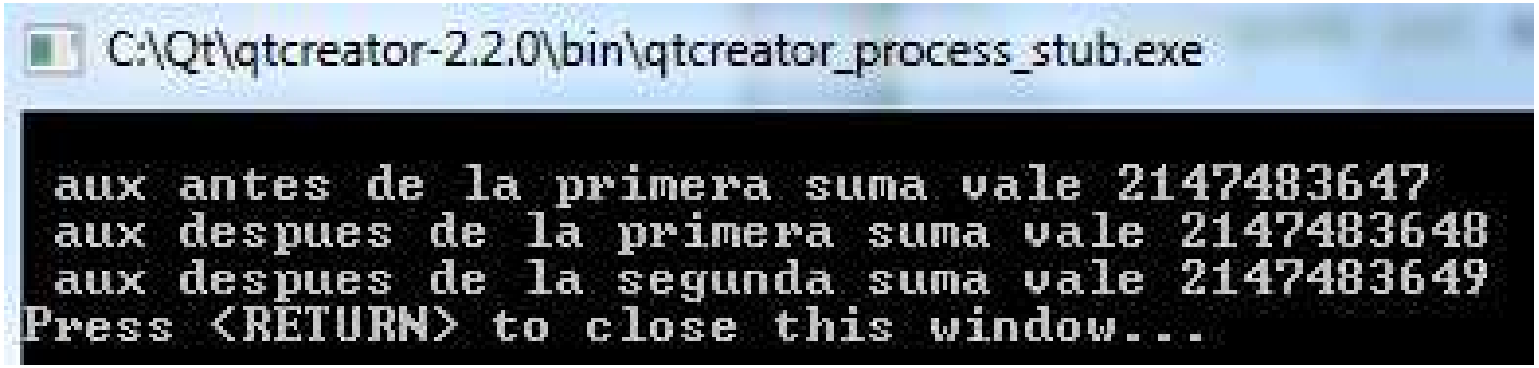
- Los tamaños de los enteros dependen:
  - Del Sistema Operativo
  - Del compilador
- La función sizeof() devuelve tamaño de la variable o el objeto en bytes
  - sizeof(int)

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;
}
```



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 de convierte al tipo de la variable izda

  Si *i* es variable entera, si i=3/9.0 entonces i vale 0

- Un *cast* es una conversión explícita

  Aunque f sea un float, si f=3/9 entonces f=0

  Al hacer cast, f=3/**(float)**9 entonces 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

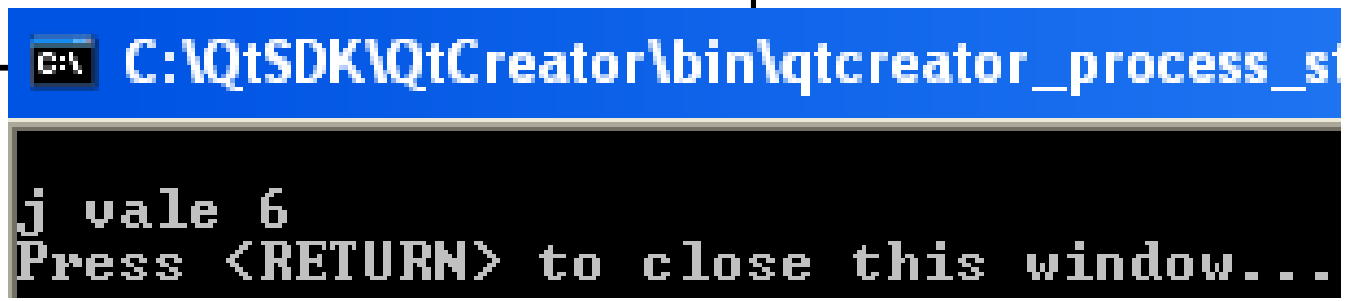
```
int a;
int number=2, Number=3;
float Number2;
Number2=3/9;
char letra;
letra='a';
char letra2='v';
```

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;
}
```



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();
   }
}
```

¿Cuál es el resultado?

Ejemplo de variable *static*

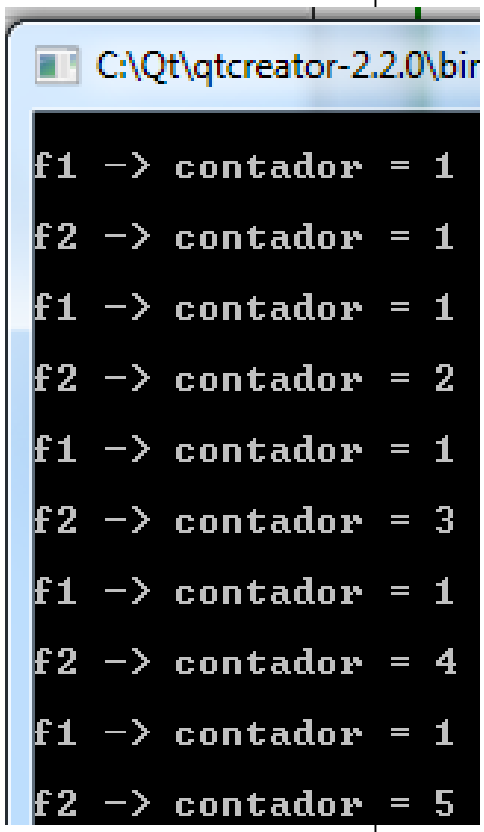
```
#include <iostream>
 using namespace std;
void f1(void)
{
   int contador=0;
   cout << "\nf1 -> contador = " << ++contador << endl;
}

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

…
```

```
…

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



¿Cuál es el resultado?

INFORMÁTICA INDUSTRIAL

PROGRAMACIÓN BÁSICA C++

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid