

# INFORMÁTICA INDUSTRIAL

## PROGRAMACIÓN BÁSICA C++ (II)

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

**uc3m** | Universidad **Carlos III** de Madrid

# 6. Flujo de Control

# Flujo de Control

Para cada estudiante

Hacer

Estudiar

Hacer el examen

Si no estás de acuerdo con la calificación ➡

Ir a revisión

Mientras la asignatura no esté aprobada

# Operadores para el flujo de control

- Operadores de Relación  
< > <= >=
- Operadores de Igualdad/Desigualdad  
= = (== no es =, que es asignación) !=
- Operadores lógicos
- && || !
- Todos devuelven:
  - 0 si la condición es falsa
  - 1 cuando la condición es válida
- En general, cualquier número diferente de cero significa *“true”* (salida *“1”*)

# if

if(expresión)

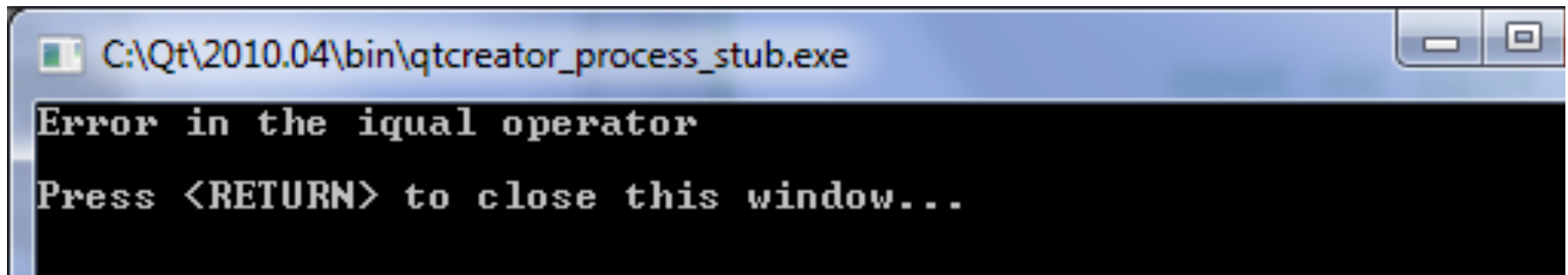
    sentencia1;

siguientes sentencias;

- Si se cumple “expresión” (salida lógica diferente de cero), se ejecuta “sentencia1” (y luego las siguientes sentencias).
- Si no se cumple “expresión” (salida lógica es cero), se ejecutan directamente las siguientes sentencias, pero no “sentencia1”.

# if

```
void main(){  
    int i=0;  
    if(i=1)  
        cout<<"error en el operador de igualdad ";  
}
```



# if

```
void main(){  
    int i=0;  
    if(i==1)  
        cout<<"error en el operador de igualdad ";  
}
```



```
C:\QtSDK\QtCreator\bin\qtcreator_process_stub.exe
```

```
Press <RETURN> to close this window...
```

# if-else

```
if(expresión)
```

```
    sentencia1;
```

```
else
```

```
    sentencia2;
```

```
siguientes sentencias;
```



# if-else

Si “expresión” es verdadero, se ejecuta “sentencia1”.

Si no, se ejecuta sentencia2.

Y siempre después, se ejecutan las siguientes sentencias.

# if-else

```
#include <iostream>
#include <math.h>

using namespace std;
int main(){
    double numero;
    cin>>numero;
    if(numero < 0)
        cout<<"Numero negativo"<<endl;
    else {
        cout<<"Numero positivo"<<endl;
        cout<<"La raíz cuadrada es "<<sqrt(numero)<<endl;
    }
    return 0;
}
```

# If-else anidado

```
#include <iostream>

using namespace std;
int main() {
    int i,j;
    i=3;
    j=-3;
    if(i<0)
        if(j<0)
            cout<<"i j menores que 0"<<endl;
        else
            cout<<"i no es menor que cero"<<endl;
    return 0;
}
```

```
#include <iostream>

using namespace std;
int main() {
    int i,j;
    i=3;
    j=-3;
    if(i<0){
        if(j<0)
            cout<<"i j menores que 0"<<endl;
        }else
            cout<<"i no es menor que cero"<<endl;
    return 0;
}
```

# If-else anidado

```
if(exp1)
{
    if(exp2)
    {
        sentencia1;
    }
    else
    {
        sentencia2;
    }
}
else
{
    sentencia3;
}
```

# If-else anidado

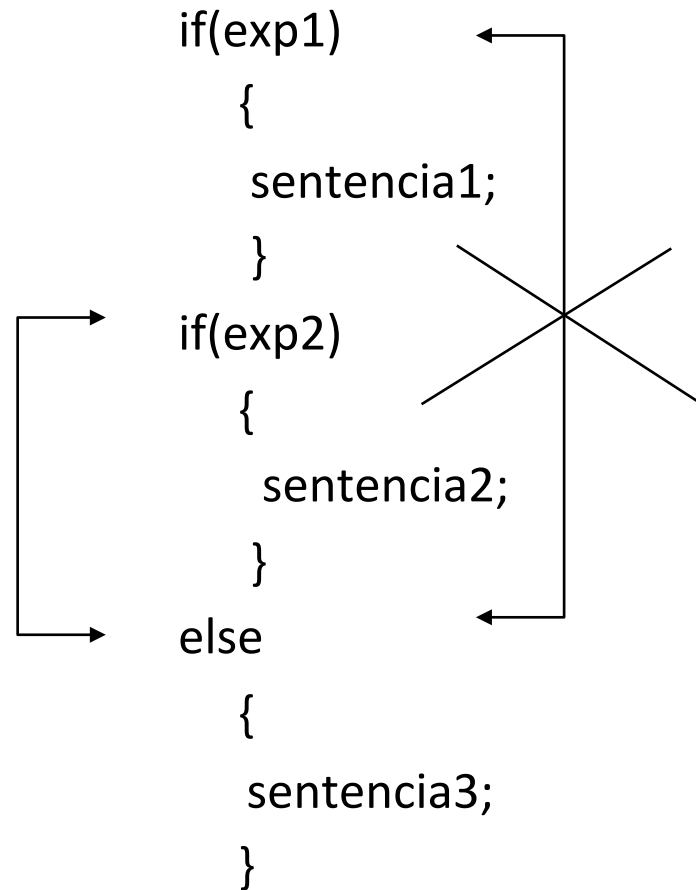
```
#include <iostream>

using namespace std;

int main() {
    int i,j;
    i=3;
    j=-3;
    if(i < 0) {
        if(j < 0)
            cout<<"i j menores que 0"<<endl;
        else
            cout<<"j no es menor que cero, aunque i sí lo sea"<<endl;
    }else
        cout<<"i no es menor que cero"<<endl;
    return 0;
}
```

# If-else anidado

La sentencia “else” está asociada con el “if” más cercano



# while

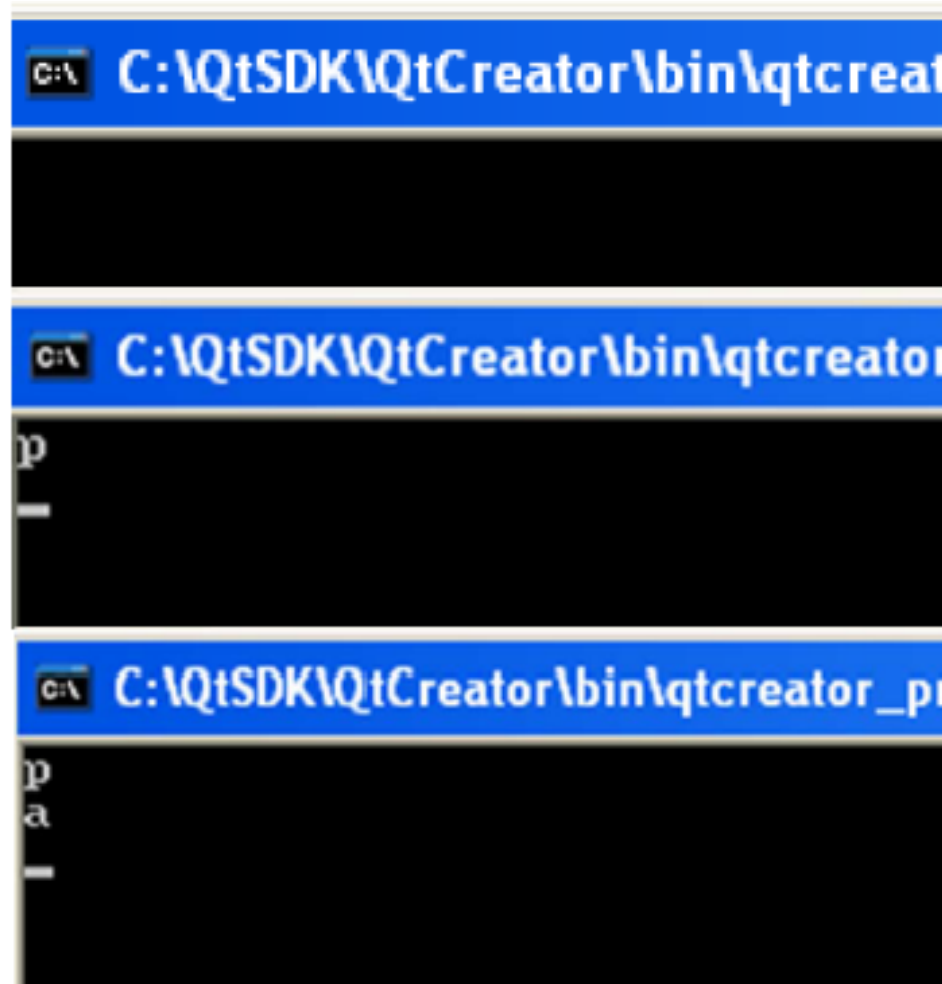
- Si queremos ejecutar sentencias que dependen de una condición

```
while(expresión)
{
    sentencia1;
}
```

# while

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

int main()
{
    char c= '\0' ;
    while (c!= 't')
        cin >> c;
    return 0;
}
```

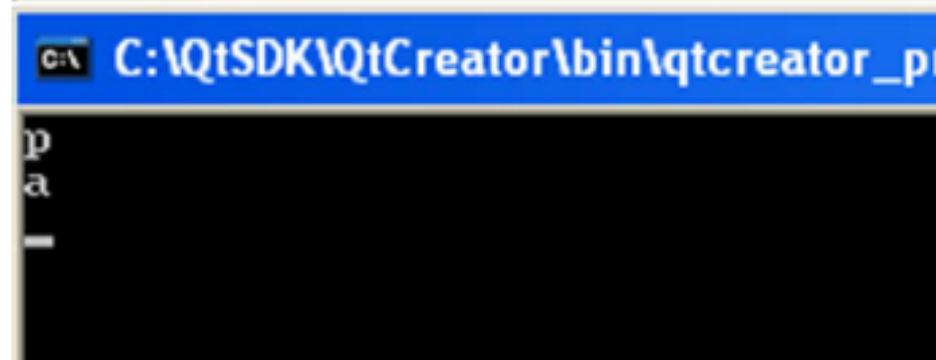




# while

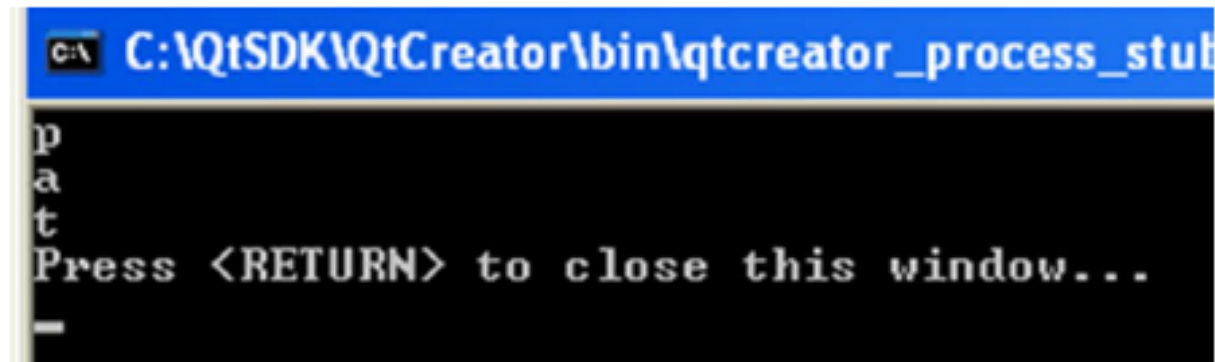
```
#include <iostream>
using namespace std;

int main()
{
    char c= '\0';
    while (c!= 't')
        cin >> c;
    return 0;
}
```



C:\QtSDK\QtCreator\bin\qtcreator\_pr

```
p
a
_
```



C:\QtSDK\QtCreator\bin\qtcreator\_process\_stub

```
p
a
t
Press <RETURN> to close this window...
_
```

# for

Se ejecuta desde el principio

Se ejecuta en cada iteración

```
expresión1;  
while(expresión2)  
{  
    sentencia1;  
    expresión3;  
}
```



```
for(exp1;exp2;exp3)  
{  
    statement1;  
}
```

En cada iteración, se verifica para decidir ejecutar o no una nueva iteración

# for

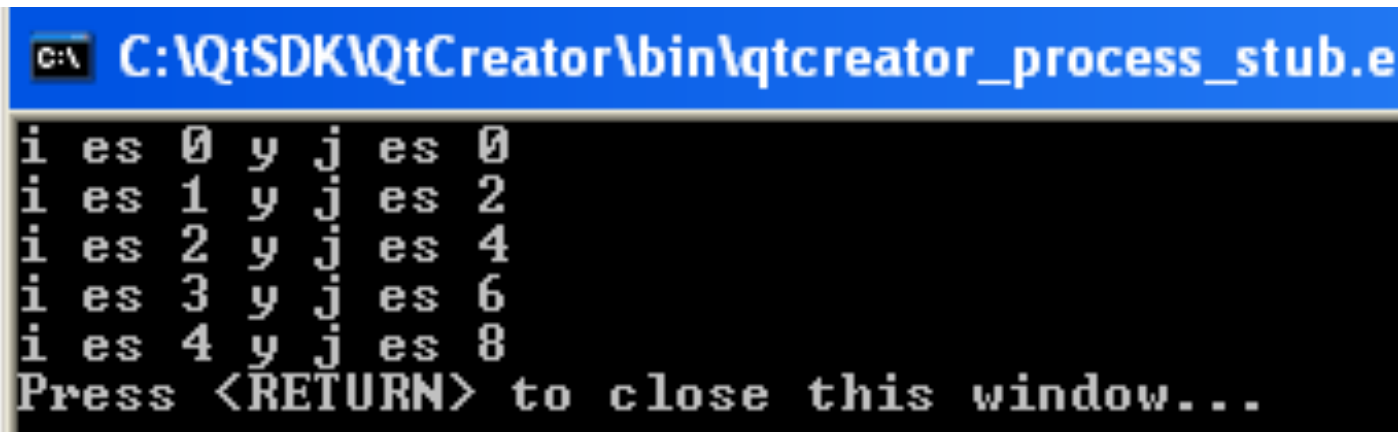
- El uso común de “for” es en bucles con un número conocido de iteraciones

```
int a[10];
for(i=0; i<10; ++i)
{
    cout <<“elemento”<<“ ” <<i<<“es”<<“ ” <<a[i]<<endl;
}
```

The diagram illustrates the components of the for loop header: `i=0` is annotated as the starting value, `i<10` as the test condition, and `++i` as the increment. The body of the loop is annotated as the sentences to be executed.

# for

```
#include <iostream>
using namespace std;
int main() {
    int i,j;
    for(i=0,j=0; i<5 && j<20; i++, j+=2)
        cout<<"i es "<<i<<" j es "<<j<<endl;
    return 0;
}
```



```
C:\QtSDK\QtCreator\bin\qtcreator_process_stub.e
i es 0 y j es 0
i es 1 y j es 2
i es 2 y j es 4
i es 3 y j es 6
i es 4 y j es 8
Press <RETURN> to close this window...
```

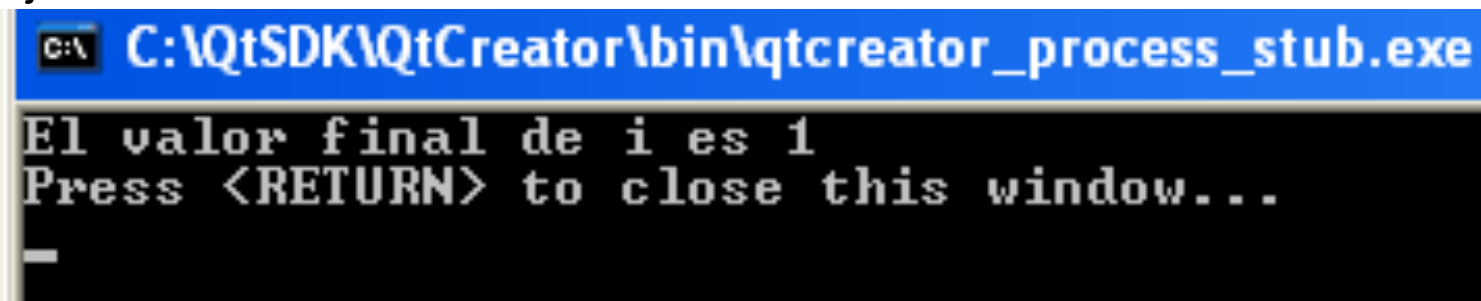
# do-while

- Es similar a “while”, pero se utiliza cuando necesitamos ejecutar un conjunto de instrucciones por lo menos una vez (incluso cuando la expresión es falsa)

```
do
{
sentencia1;
}
while(expresión);
```

# do-while

```
#include <iostream>
using namespace std;
int main() {
    int i=0;
    do{
        i+=1;
    }while(i<0);
    cout<<"El valor final de i es "<<i<<endl;
    return 0;
}
```



The screenshot shows a terminal window with a blue title bar containing the text "C:\QtSDK\QtCreator\bin\qtcreator\_process\_stub.exe". The terminal output displays "El valor final de i es 1" followed by "Press <RETURN> to close this window...". A cursor is visible at the bottom left of the terminal.

# “break” y “continue”

- “**break**” causa la salida de un bucle anidado (**interno**)

```
while(1){  
    cin >> x;  
    if(x<0.0)  
        break;  
    else  
        cout<<"raiz cuadrada"<<sqrt(x)<<endl;  
}
```

# “break” y “continue”

- **continue** interrumpe la iteración actual del bucle a la siguiente iteración

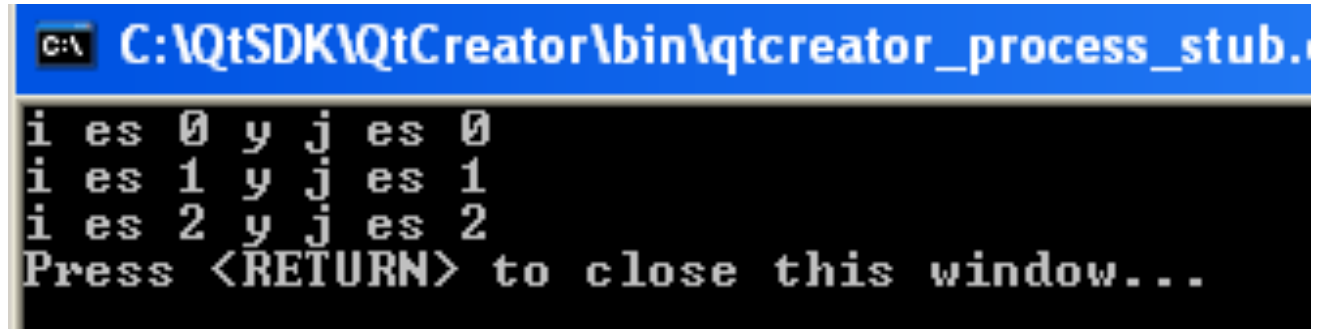
```
for(i=0;i<1000;i++){  
    c=getchar();  
    if('0'<=c && c<='9')  
        continue;  
}
```

- Solo se utiliza con **for**, **while** y **do-while**



# break and continue

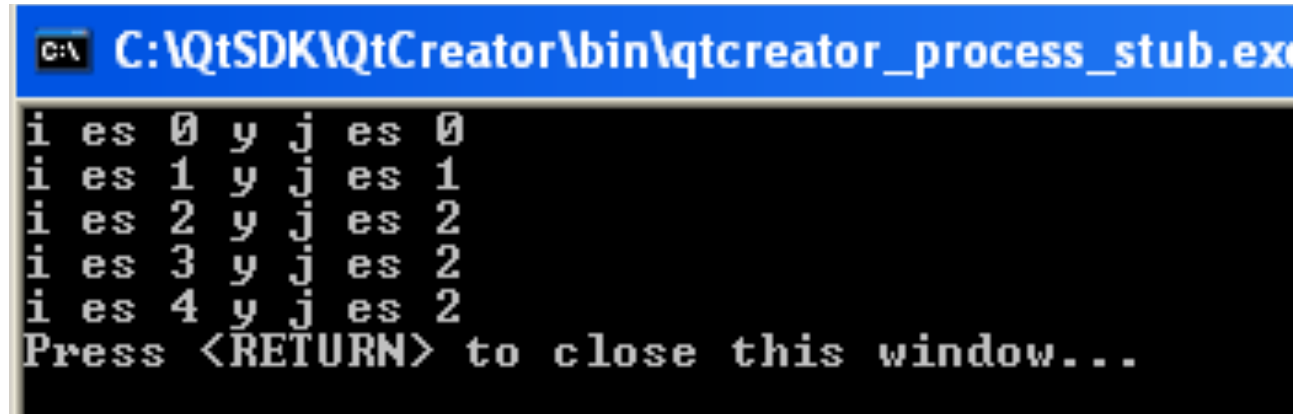
```
#include <iostream>
using namespace std;
int main() {
    int i,j=0;
    for(i=0; i<5; i++){
        cout<<"i es "<<i<<" j es "<<j<<endl;
        if(j>1)
            break;
        j++; //j+=1; o j=j+1;
    }
    return 0;
}
```



```
C:\QtSDK\QtCreator\bin\qtcreator_process_stub...
i es 0 y j es 0
i es 1 y j es 1
i es 2 y j es 2
Press <RETURN> to close this window...
```

# break and continue

```
#include <iostream>
using namespace std;
int main() {
    int i,j=0;
    for(i=0; i<5; i++){
        cout<<"i es "<<i<<" j es "<<j<<endl;
        if(j>1)
            continue;
        j++; //j+=1; o j=j+1;
    }
    return 0;
}
```



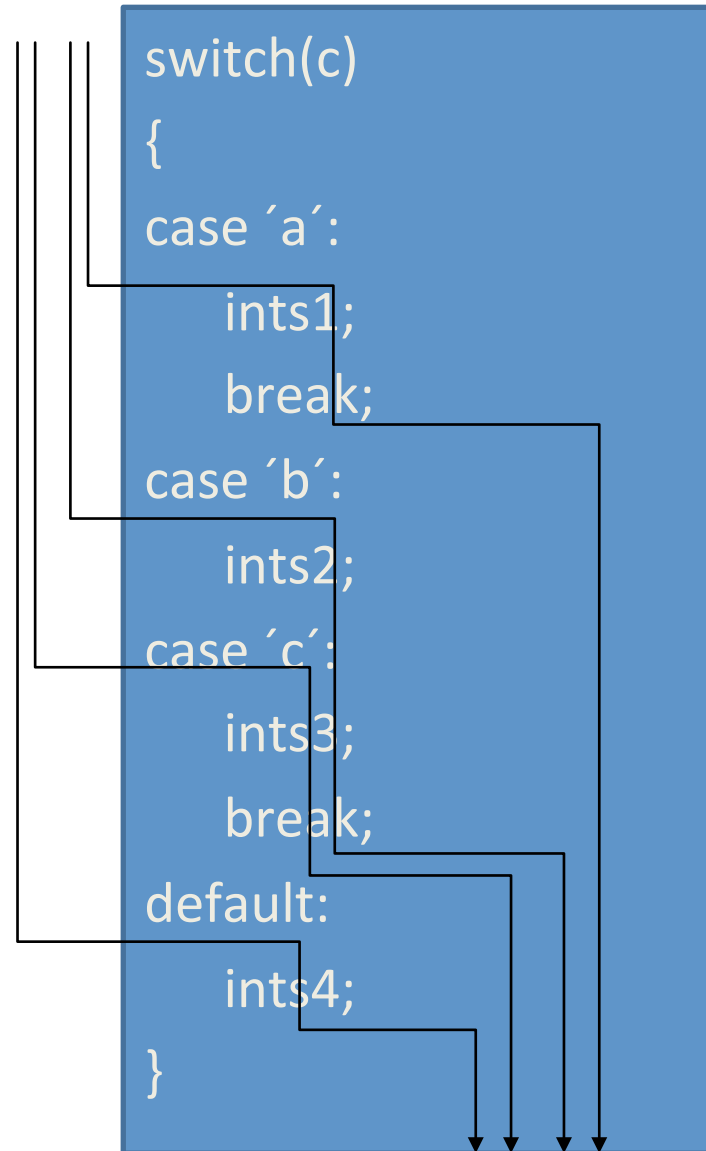
```
C:\QtSDK\QtCreator\bin\qtcreator_process_stub.exe
i es 0 y j es 0
i es 1 y j es 1
i es 2 y j es 2
i es 3 y j es 2
i es 4 y j es 2
Press <RETURN> to close this window...
```

# switch

- Generaliza el uso de múltiples sentencias “if-else”
- Normalmente se utiliza cuando existen varias opciones
- Verifica y chequea el valor de una variable introducida, permitiendo además valores por defecto.

# switch

```
switch(c)
{
case 'a':
    ints1;
    break;
case 'b':
    ints2;
case 'c':
    ints3;
break;
    default:
    ints4;
}
```



# switch

- Es típico poner “break” al final de cada caso para evitar entrar en el siguiente.
- Es muy útil para menús con selección (“Escribe 1 para calcular la suma, 2 para calcular la resta, ...”).

# INFORMÁTICA INDUSTRIAL

## PROGRAMACIÓN BÁSICA C++ (II)

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

**uc3m** | Universidad **Carlos III** de Madrid