



*Object Oriented Programming by C++*

## Selection & Repetition (1/2)

Conditional execution and Iteration (Loop)

2017. 8.

Sungwon Lee / Professor

Email: [drsungwon@khu.ac.kr](mailto:drsungwon@khu.ac.kr)

Web: <http://mobilelab.khu.ac.kr/>

Modified & taught by Jinwoo Choi  
in 2023 Spring

Email: [jinwoochoi@khu.ac.kr](mailto:jinwoochoi@khu.ac.kr)

Webpage: <https://sites.google.com/site/jchoivision/>

# Contents

---

- Boolean expression
- *if-else* statement
- *while* statement

# Boolean Expression

## True or False

- Conditionally *true* or *false*

In C++,

i) = -> assignment operator

e.g. int x;

x = 10;

ii) == -> equal?

e.g. 10 == 10 -> true

10 == 20 -> false

x == 10

iii) !=

e.g. 10 != 10 -> false

10 != 20 -> true

x != 10 -> false only if x has the value 10

Expression	Value
$10 < 20$	always true
$10 \geq 20$	always false
$x == 10$	true only if x has the value 10
$x \neq y$	true unless x and y have the same values

# Boolean Expression

## Type *bool*

- Special variable type : *bool*
- bool* type variable can have binary value: *true* or *false* (constant in C++)

1            0

```
#include <iostream>
using namespace std;
int main() {
    bool a = true, b = false;
    std::cout << "(a:b)" << "(" << a << ":" << b << ")";
    return 0;
}
```

(a:b)(1:0)

- In C++, *true* outputs 1, and *false* outputs 0
  - If a variable has *non-zero value*, it is translated as *true*

## if-else statement

# Simple if Statement

- Conditional execution
  - Single statement execution
  - Multiple statement execution

```
if ( condition )  
    do something
```

```
if ( condition )  
{  
    do something #1  
    ...  
    do something #n  
}
```

## if-else statement

# Simple if Statement Example

### Listing 5.2: betterdivision.cpp

```
#include <iostream>

int main() {
    int dividend, divisor;

    // Get two integers from the user
    std::cout << "Please enter two integers to divide:";
    std::cin >> dividend >> divisor;
    // If possible, divide them and report the result
    if (divisor != 0)
        std::cout << dividend << "/" << divisor << " = "
            << dividend/divisor << '\n';
}
```

if (divisor != 0) is true:

```
Please enter two integers to divide: 32 8
32/8 = 4
```

if (divisor != 0) is false:

```
Please enter two integers to divide: 32 0
```

## if-else statement

# Simple if Statement Flow-chart

### Listing 5.2: betterdivision.cpp

```
#include <iostream>

int main() {
    int dividend, divisor;

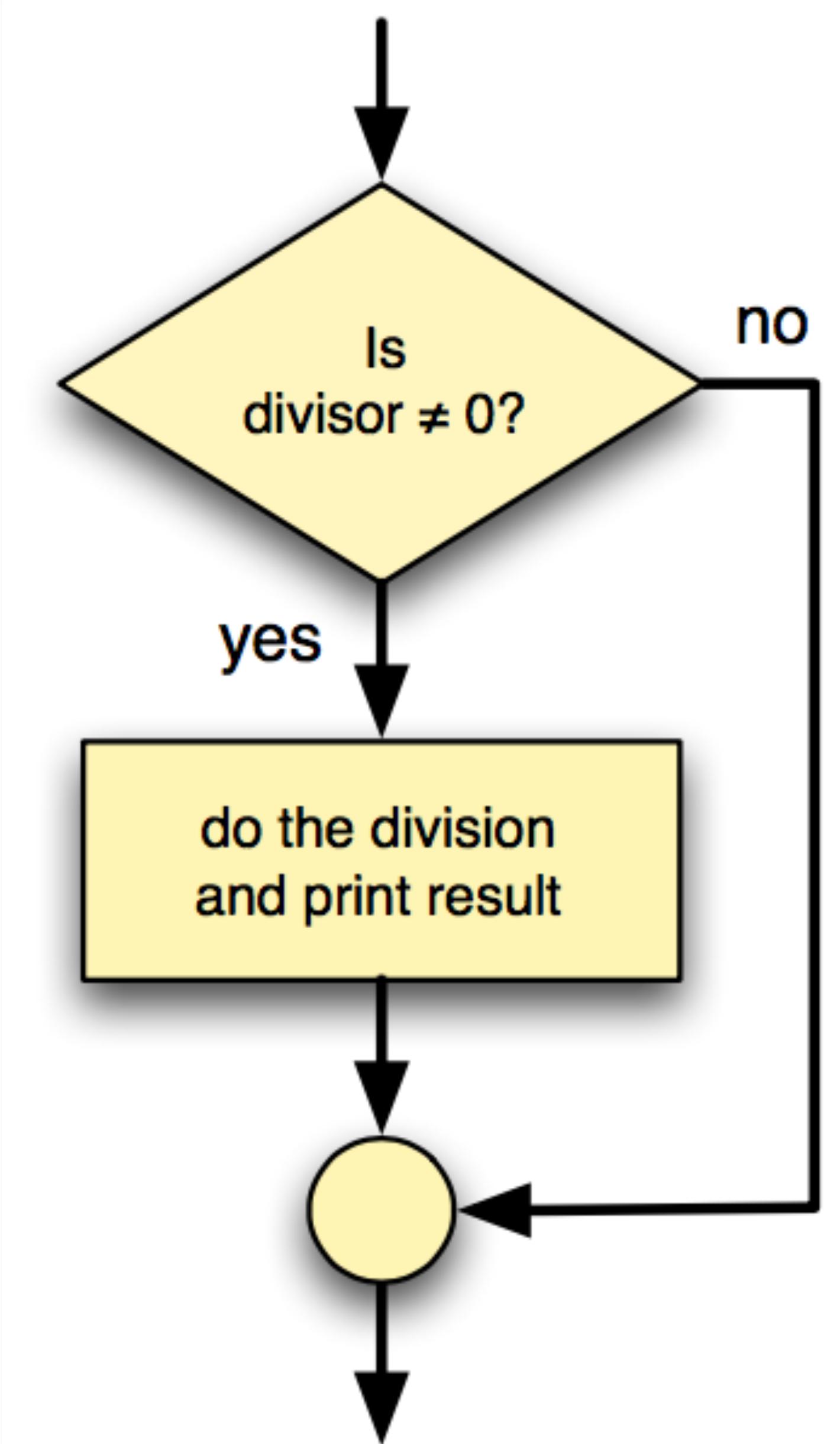
    // Get two integers from the user
    std::cout << "Please enter two integers to divide:";
    std::cin >> dividend >> divisor;
    // If possible, divide them and report the result
    if (divisor != 0)
        std::cout << dividend << "/" << divisor << " = "
            << dividend/divisor << '\n';
}
```

if (divisor != 0) is true:

```
Please enter two integers to divide: 32 8
32/8 = 4
```

if (divisor != 0) is false:

```
Please enter two integers to divide: 32 0
```



## if-else statement

# Simple if Statement Example

### Listing 5.3: alternatedivision.cpp

```
#include <iostream>

int main() {
    int dividend, divisor, quotient;

    // Get two integers from the user
    std::cout << "Please enter two integers to divide:";
    std::cin >> dividend >> divisor;
    // If possible, divide them and report the result
    if (divisor != 0) {
        quotient = dividend / divisor;
        std::cout << dividend << " divided by " << divisor << " is "
            << quotient << '\n';
    }
}
```



```
if (divisor != 0){
    quotient = dividend/divisor;
    std::cout << dividend << " divided by " << divisor << " is " << quotient << '\n';
}
```

Multiple Statement Execution

## if-else statement

# Simple if Statement Writing

- C++ compiler ignores **space** and **endl**; all following are same

```
if(x < 10)
```

```
    y = x;
```

Recommended

```
if(x < 10) y = x;
```

```
if(x < 10)
```

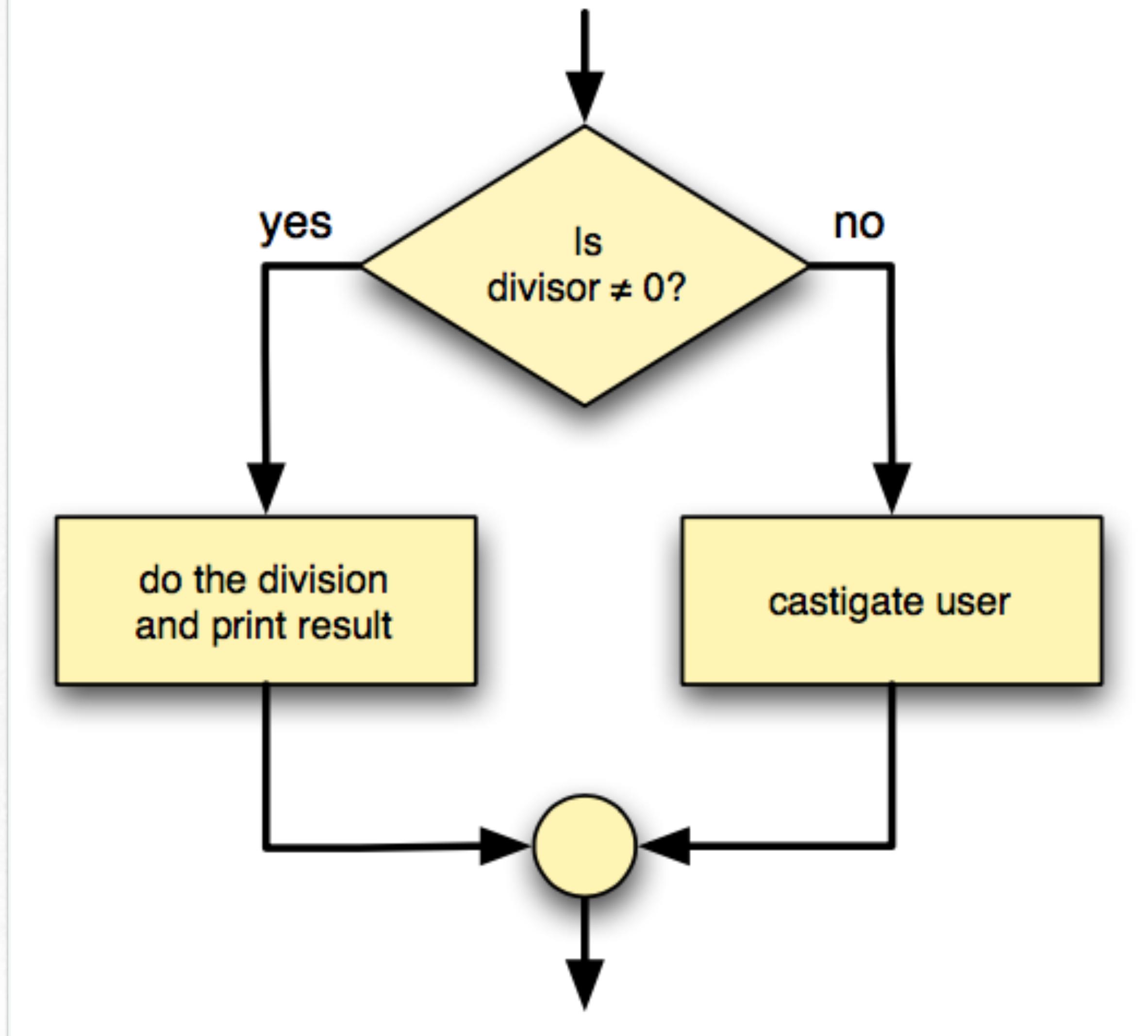
```
    y = x;
```

## if-else statement

# Simple if-else Statement

- “ *if you can fly then fly! else then walk!* ”

```
if ( condition )  
    statement 1  
  
else  
    statement 2
```



## if-else statement

# Simple if-else Statement Example

### **Listing 5.4: betterfeedback.cpp**

```
#include <iostream>

int main() {
    int dividend, divisor;

    // Get two integers from the user
    std::cout << "Please enter two integers to divide:";
    std::cin >> dividend >> divisor;
    // If possible, divide them and report the result
    if (divisor != 0)
        std::cout << dividend << "/" << divisor << " = "
                    << dividend/divisor << '\n';
    else
        std::cout << "Division by zero is not allowed\n";
}
```

if (divisor != 0) is false:

```
Please enter two integers to divide: 32 0
Division by zero is not allowed
```

# if-else statement

## Logical Operator (AND, OR, NOT)

and: &&

or: ||

not: !

bitwise and: &

bitwise or: |

not: !

$e_1$	$e_2$	$e_1 \&\& e_2$	$e_1 \mid\mid e_2$	$!e_1$
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

### Listing 5.7: newcheckrange.cpp

```
#include <iostream>

int main() {
    int value;
    std::cout << "Please enter an integer value in the range 0...10: ";
    std::cin >> value;
    if (value >= 0 && value <= 10)
        std::cout << "In range\n";
}
```

1byte variable ch = 8;

00001000

!ch -> 11110111

ch2 12;

00001100

ch & ch2: and

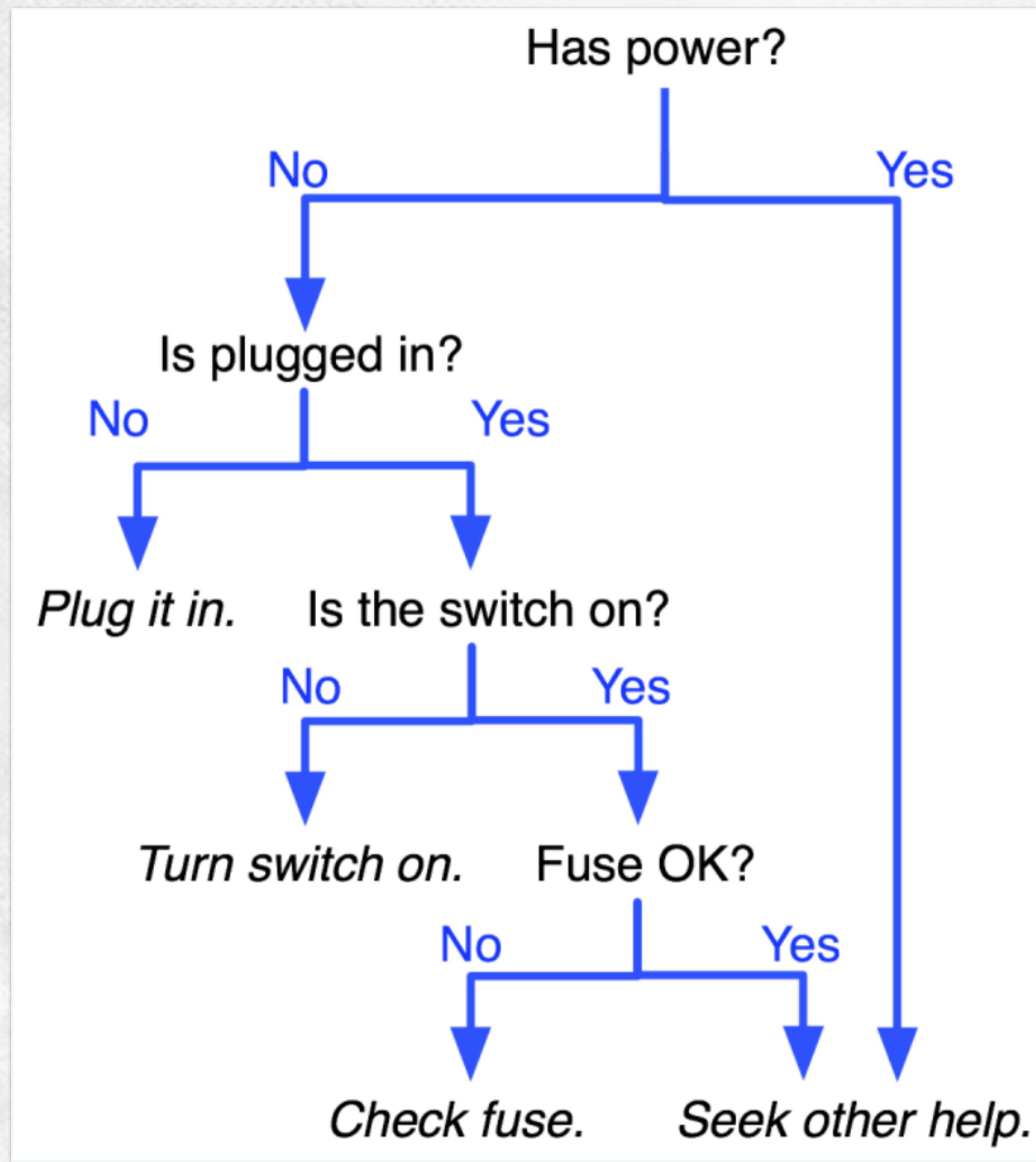
00001000

ch | ch2: or

00001100

# if-else statement

## Nested if-else Statement



```
if ( condition )
{
    do something ...
    if ( condition )
    {
        do something ...
    }
    else if ( condition )
    {
        do something ...
    }
    else
    {
        do something ...
    }
}
```

## if-else statement

# Nested *if-else* Statement Example

---

- Code Review: Listing 5.13 & 5.16 in Textbook

**Let's read together !!**

## while statement

# Statement Description

- “Don’t speak! while you work! ”

- Iteration

- Single statement Iteration
- Multiple statement Iteration

```
while ( condition )  
    do something
```

```
while ( condition )  
{  
    do something #1  
    ...  
    do something #n  
}
```

# while statement

## Statement Example

### Listing 6.1: counttofive.cpp

```
#include <iostream>

int main() {
    std::cout << 1 << '\n';
    std::cout << 2 << '\n';
    std::cout << 3 << '\n';
    std::cout << 4 << '\n';
    std::cout << 5 << '\n';
}
```

Be careful of infinite loop!

### Listing 6.2: iterativecounttofive.cpp

```
#include <iostream>

int main() {
    int count = 1; // Initialize counter
    while (count <= 5) {
        std::cout << count << '\n'; // Display counter, then
        count++; // Increment counter
    }
}
```

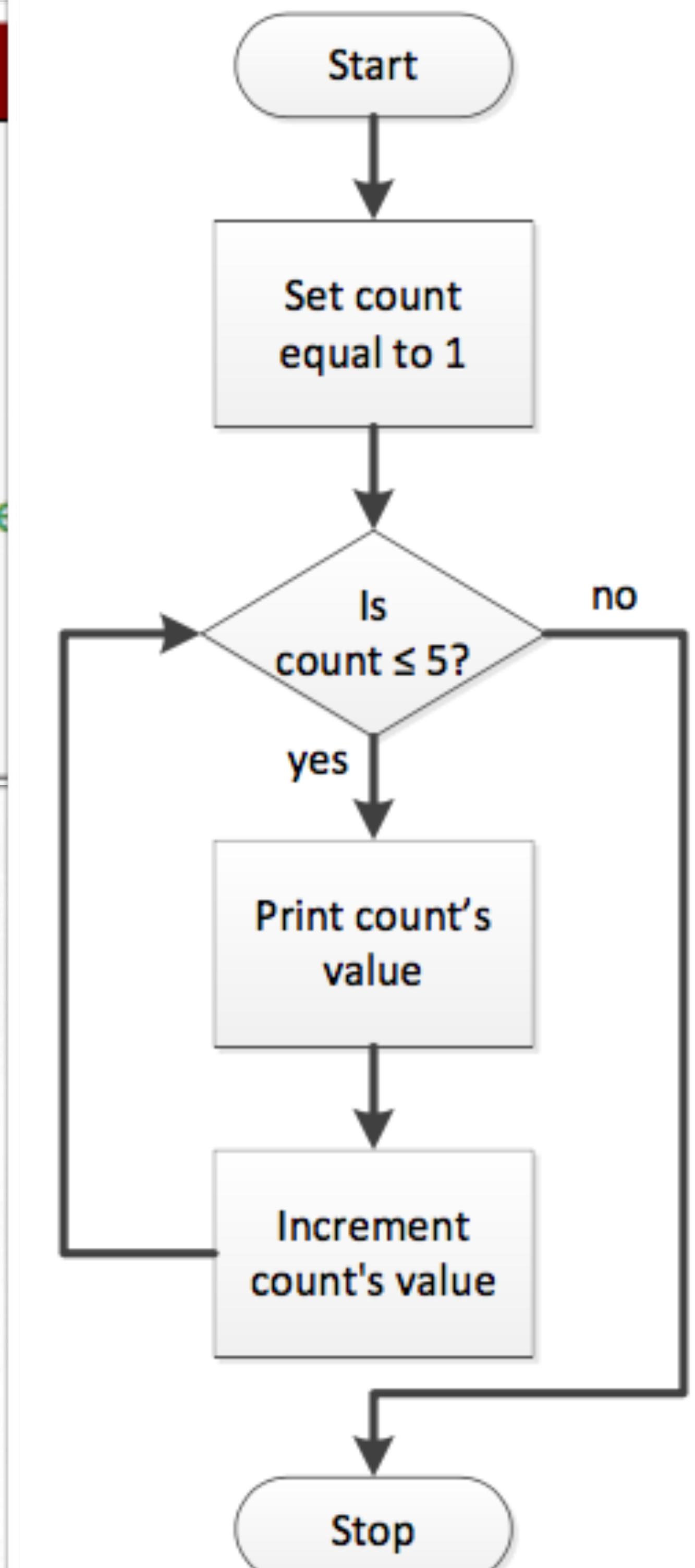
# while statement

## Statement Example

### Listing 6.2: iterativecounttofive.cpp

```
#include <iostream>

int main() {
    int count = 1; // Initialize counter
    while (count <= 5) {
        std::cout << count << '\n'; // Display counter
        count++; // Increment counter
    }
}
```



## while statement

# Mixed with Conditional Statement

### Listing 6.4: addnonnegatives.cpp

```
/*
 * Allow the user to enter a sequence of nonnegative
 * integers. The user ends the list with a negative
 * integer. At the end the sum of the nonnegative
 * integers entered is displayed. The program prints
 * zero if the users no nonnegative integers.
 */

#include <iostream>                                         3 10 7 8 -9

int main() {
    int input = 0,      // Ensure the loop is entered
        sum = 0;       // Initialize sum

    // Request input from the user
    std::cout << "Enter numbers to sum, negative number ends list:";

    while (input >= 0) {      // A negative number exits the loop
        std::cin >> input;   // Get the value
        if (input >= 0)
            sum += input;   // Only add it if it is nonnegative
    }
    std::cout << "Sum = " << sum << '\n'; // Display the sum
}
```

## while statement

# Example with standard functions (1/3)

### Listing 6.6: powersof10.cpp

```
#include <iostream>

int main() {
    int power = 1;
    while (power <= 1000000000) {
        std::cout << power << '\n';
        power *= 10;
    }
}
```

```
1
10
100
1000
10000
100000
1000000
10000000
100000000
1000000000
```

# while statement

## Example with standard functions (2/3)

### Listing 6.7: powersof10justified.cpp

```
#include <iostream>
#include <iomanip>

// Print the powers of 10 from 1 to 1,000,000,000
int main() {
    int power = 1;
    while (power <= 1000000000) {
        // Right justify each number in a field 10 wide
        std::cout << std::setw(10) << power << '\n';
        power *= 10;
    }
}
```

```
1
10
100
1000
10000
100000
1000000
10000000
100000000
1000000000
```

The screenshot shows the cplusplus.com website with the search term "setw" entered. The results page for "setw" is displayed, showing the std::setw function under the "Reference" section. The function is described as setting the field width for output operations. It includes parameters (n), a return value (unspecified), and an example code snippet. The example code is identical to the one in Listing 6.7.

function  
std::setw  
Sets the *field width* to be used on output operations.  
Behaves as if member `width` were called with *n* as argument on the stream on which it is inserted/extracted as a manipulator (it can be inserted/extracted on *input streams* or *output streams*).  
This manipulator is declared in header `<iomanip>`.

**Parameters**  
n Number of characters to be used as field width.

**Return Value**  
Unspecified. This function should only be used as a stream manipulator (see example).

**Example**

```
1 // setw example
2 #include <iostream>      // std::cout, std::endl
3 #include <iomanip>       // std::setw
4
5 int main () {
6     std::cout << std::setw(10);
7     std::cout << 77 << std::endl;
8 }
```

Output: 77

# while statement

## Example with standard functions (3/3)

### Listing 6.8: powersof10withcommas.cpp

```
#include <iostream>
#include <iomanip>
#include <locale>

// Print the powers of 10 from 1 to 1,000,000,000
int main() {
    int power = 1;
    std::cout.imbue(std::locale(""));
    while (power <= 1000000000) {
        // Right justify each number in a field 10 wide
        std::cout << std::setw(13) << power << '\n';
        power *= 10;
    }
}
```

```
1
10
100
1,000
10,000
100,000
1,000,000
10,000,000
100,000,000
1,000,000,000
```

The screenshot shows the Cplusplus.com website documentation for the `std::ios::imbue` function. The page includes a search bar, navigation links for Reference, <ios>, <ios>, and Imbue, and a user login status. The main content area details the function's purpose, parameters, return value, and examples. It also includes a note about floating-point output in non-English locales.

**Google Cloud Platform.**  
Start your first VM for free with Google Compute Engine.  
[cloud.google.com](http://cloud.google.com)

**public member function**  
`std::ios::imbue`

`locale imbue (const locale& loc);`

**Imbue locale**  
Associates `loc` to both the stream and its associated `stream buffer` (if any) as the new locale object to be used with locale-sensitive operations.

This function calls its inherited homonym `ios_base::imbue(loc)` and, if the stream is associated with a `stream buffer`, it also calls `rbdbuf()->pubimbue(loc)`.

All callback functions registered with member `register_callback` are called by `ios_base::imbue`.

**Parameters**  
`loc` locale object to be imbued as the new locale for the stream.

**Return value**  
The locale object associated with the stream before the call.

**Example**

```
1 // imbue example
2 #include <iostream>           // std::cout
3 #include <locale>            // std:::locale
4
5 int main()
6 {
7     std::locale mylocale(""); // get global locale
8     std::cout.imbue(mylocale); // imbue global locale
9     std::cout << 3.14159 << '\n';
10    return 0;
11 }
```

This code writes a floating point number using the global locale given by the environment. For example, in a system configured with a Spanish locale as default, this could write the number using a comma decimal separator:

3,14159

## while statement

# Nested *while* Statement

```
while ( condition )
{
    do something ...

    while ( condition )
    {
        do something ...

    }

    do something ...
}
```

# while statement

## Nested *while* Statement Example

### Listing 6.12: **timestable-3rd-try.cpp**

```
#include <iostream>
#include <iomanip>

int main() {
    int size; // The number of rows and columns in the table
    std::cout << "Please enter the table size: ";
    std::cin >> size;
    // Print a size x size multiplication table
    int row = 1;
    while (row <= size) { // Table has size rows.
        int column = 1; // Reset column for each row.
        while (column <= size) { // Table has size columns.
            int product = row*column; // Compute product
            std::cout << std::setw(4) << product; // Print product
            column++; // Next element in row
        }
        std::cout << '\n'; // Move to next row
        row++; // Next row
    }
}
```

Please enter the table size: 10										
1	2	3	4	5	6	7	8	9	10	
2	4	6	8	10	12	14	16	18	20	
3	6	9	12	15	18	21	24	27	30	
4	8	12	16	20	24	28	32	36	40	
5	10	15	20	25	30	35	40	45	50	
6	12	18	24	30	36	42	48	54	60	
7	14	21	28	35	42	49	56	63	70	
8	16	24	32	40	48	56	64	72	80	
9	18	27	36	45	54	63	72	81	90	
10	20	30	40	50	60	70	80	90	100	



# *Object Oriented Programming by C++*

Sungwon Lee / Professor

Email: [drsungwon@khu.ac.kr](mailto:drsungwon@khu.ac.kr)

Web: <http://mobilelab.khu.ac.kr/>