C++ Programming

Ch. 6 Branching Statements and Logical Operators

Spring 2014

Myung-Il Roh

Department of Naval Architecture and Ocean Engineering
Seoul National University

Ch. 6 Branching Statements and Logical Operators

Contents

☑ The 'if' Statement

☑ The 'if else' Statement

☑ The 'if else if else' Statement

☑ Logical Expressions

☑ Practice

☑ The 'switch' Statement

☑ The Conditional Operators

☑ Summary

☑ Practice

The ‘if’ Statement

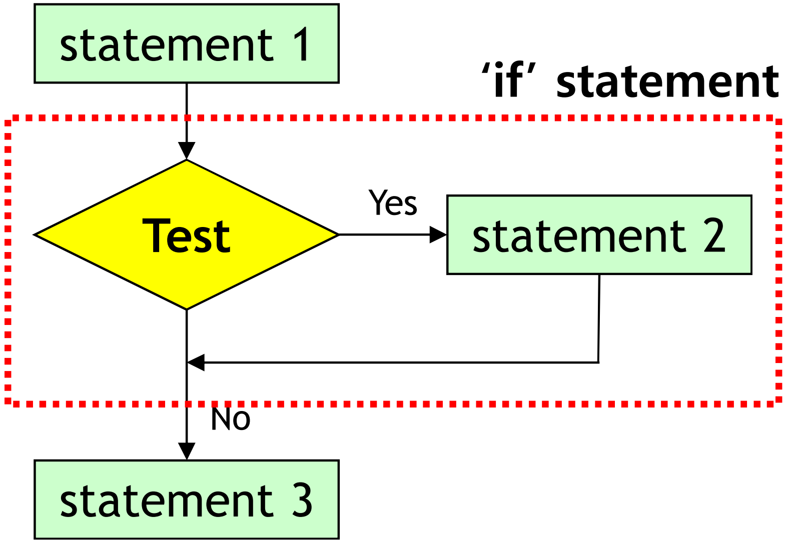
☑ The ‘if’ statement directs a program to

and to

.

☑ Expression

```
statement 1;
if (test-expression)
    statement 2 or body;
statement 3;
```



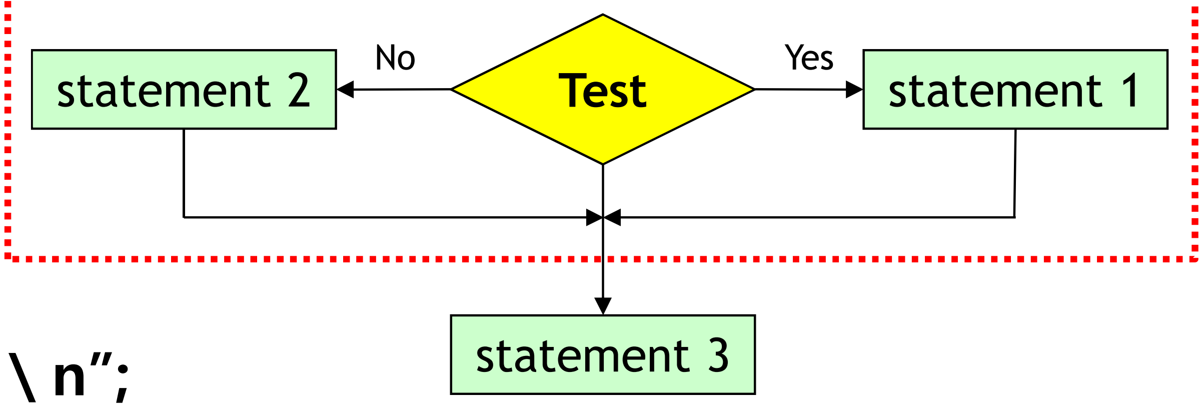
The ‘if else’ Statement

☑ The ‘if else’ statement let a program   . That is, the ‘if else’ statement is used when we want to execute a statement or block according to a test condition or expression.

☑ Expression

```
if (test-expression)
    statement 1;
else
    statement 2;
statement 3;
```

‘if else’ statement



☑ Ex.

```
if (answer == 1492)
    cout << “That’s right! \ n”;
else
    cout << “You’d better review chapter 1 again. \ n”;
```

The ‘if else if else’ Statement

☑ The ‘if else if else’ statement let a program . That is, it is an expansion of the ‘if else’ statement. It is possible to use ‘if else if else if else…’ statement.

☑ Expression

```
if (test-expression)
    statement 1;
else if
    statement 2;
else
    statement 3;
```

Replace the statement B in the ‘if else’ statement as below with another ‘if else’ statement.

```
if (test-expression)
    statement A
else
    statement B
```

☑ Ex.

```
if (ch == ‘A’)
    a_grade++;        // alternative 1
else if (ch == ‘B’)
    b_grade++;        // alternative 2
else
    soso++;           // alternative 3
```

Blocks in the ‘if’ Statement

☑ If the body is more than one statement, construct a compound statement or block.

☑ The block consists of   and the statements they enclose and, for the purposes of syntax,

.

☑ Ex.

```
if (ch == ‘z’)
{
    Kim++;
    cout << “One of the candidates for Mr. Kim. \ n”;
}
else
{
    dull++;
    cout << “He is not the candidates for Mr. Kim. \ n”;
}
```

Logical Expressions (1/3)

☑ When we

in the statements like ‘if’, ‘for’, etc., logical operators can be used to combine or modify existing expressions.

☑ C++ provides three logical operators; logical OR, written ||; logical AND, written &&; and logical NOT, written !.

☑ The value of these operators is bool type; ‘true’ or ‘false’.

☑ **The Logical OR Operator (‘||’)**

- The value of ‘expression 1 || expression 2’

	true	false
true		
false		

⬅ expression 1

⬆ expression 2

Logical Expressions (2/3)

☑ The Logical AND Operator ('&&')

- The value of 'expression 1 && expression 2'

	true	false
true		
false		

⬅ expression 1

⬆ expression 2

☑ The Logical NOT Operator ('!')

- It reverses logical state of its operand.
- Ex.

```
if (!(x > 5))        // 'if (x <= 5)' is more clear.
```

Logical Expressions (3/3)

☑ Caution

- The logical OR and logical AND operators have a precedence than relational operators.
  - Ex.

    ```
    x > 5 && x < 10
    // This is read this way.
    (x > 5) && (x < 10)
    ```

- The logical NOT operator has a precedence than any of the relational or arithmetic operators.
  - Ex.

    ```
    !(x > 5)        // This is it false that x is greater than 5.
    !x > 5          // This is '!x greater than 5'.
    ```

- The logical AND operator has a precedence than the logical OR operator.
  - Ex.

    ```
    age > 30 && age < 45 || weight > 300
    // This is read this way.
    (age > 30 && age < 45) || weight > 300
    ```

The ‘switch’ Statement (1/3)

☑ The ‘switch’ statement let a program

.

☑ Expression

```
switch (integer-expression)
{
    case label 1: statement(s)
        break;
    case label 2: statement(s)
        break;
    …
    default: statement(s)
}
```

☑ Integer Expression

; bool, char,

short, int, and long

The ‘switch’ Statement (2/3)

☑ Variable of ‘char’ Type for Integer Expression

- Ex.

```
char choice;
cin >> choice;
switch (choice)
{
    case ‘a’:
    case ‘A’: cout << “\a\n”;
        break;
    case ‘r’:
    case ‘R’: report();
        break;
    …
    default: cout << “That’s not a choice.\n”;
}
```

The ‘switch’ Statement (3/3)

☑ Caution

.

- Thus, use the ‘break’ statements to confine execution to a particular portion of a ‘switch’ statement.

The ‘if else’ vs. The ‘switch’ Statements (1/2)

☑ Both the ‘if else’ statement and the ‘switch’ statement let a program .

☑ The ‘if else’ Statement

- It is the more versatile of the two and .
  - Ex.

```
if (age > 17 && age <= 35)
    index = 0;
else if (age > 35 && age <= 50)
    index = 1;
else if (age >50 && age < 65)
    index = 2;
else
    index = 3;
```

The ‘if else’ vs. The ‘switch’ Statements (2/2)

☑ The ‘switch’ Statement

- It isn’t designed to handle ranges.
- Each switch case label must be a (which includes char), so a switch statement can’t handle floating-point tests.
- The case label value must be a .
- The switch statement is usually
  .

The ‘break’ and ‘continue’ Statements (1/2)

☑ The ‘break’ and ‘continue’ statements enable a program to

.

☑ The ‘break’ Statement

- It causes program execution to

  .

- That is, it is possible to escape from the loop, and move on to the next statement.

☑ The ‘continue’ Statement

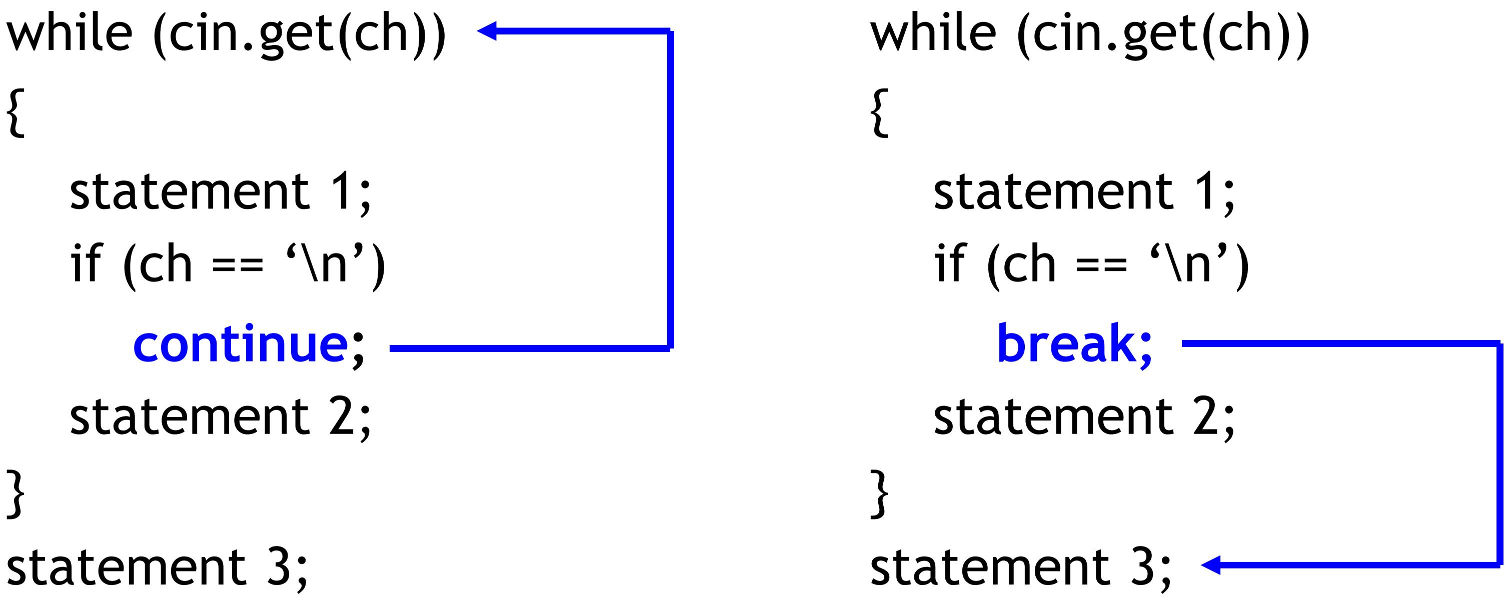
- It is used in loops and causes a program to

  .

- That is, it is possible to move on to the test or update expression.

The ‘break’ and ‘continue’ Statements (2/2)

☑ **Ex.**



```
while (cin.get(ch))
{
    statement 1;
    if (ch == ‘\n’)
        continue;
    statement 2;
}
statement 3;
```

```
while (cin.get(ch))
{
    statement 1;
    if (ch == ‘\n’)
        break;
    statement 2;
}
statement 3;
```

The Conditional Operator '? :'

☑ It be used instead of the 'if else' statement.

☑ It is more simple than the 'if else' statement but not clear.

☑ It requires 3 operands.

☑ Expression
expression 1 ? expression 2 : expression 3

☑ Ex.

```
x = 5 > 3 ? 10 : 12;        // 5>3 is true, so expression value is 10.
y = 3 == 9 ? 25 : 18;       // 3==9 is false, so expression value is
                               18.
```

☑ Useful Expression: "   "

- The same result as the following statements

```
if (a > b)
    c = a;
else
    c = b;
```

Summary (1/2)

☑ C++ provides the ‘if’ statement, the ‘if else’ statement, and the ‘switch’ statement as .

. That is, the program executes the statement or block if a particular condition is met.

. We can append additional ‘if else’ statements to such a statement to present a series of choices.

.

Summary (2/2)

☑ C++ also provides operators to help in decision making. The ‘if’ and ‘if else’ statements typically use relational expressions as test conditions.

☑ By using logical operators (&&, ||, and !), we can combine or modify relational expressions to construct more elaborate tests.

☑ The conditional operator (?:) provides a

.

Practice 1

☑ Make a program that gets all prime numbers between 10 to 100.

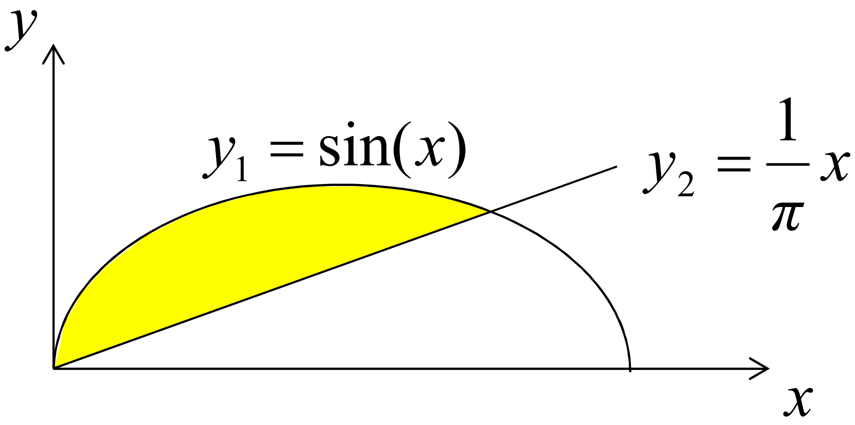
- A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.
- Use nested (double) loops.

```
Preprocessor directives
int main(void)
{
    for i = 10 to 100 {
        int n = i/2;
        while (j <= n) {
            if (i % j == 0) break;
            j = j+1;
        }
        if (j == (n + 1)) then
            print "i is a prime number".
    }
    return 0;
}
```

Practice 2

☑ **Make a program that calculates the area of the yellow region.**

$$y_1 = \sin(x),\ \ y_2 = \frac{1}{\pi}x$$



Practice 3

- ☑ Make a program that calculates the 100, 200, and 300 day-th anniversary when you input the first day of you met your girl or boy friend.
  - ■ The number of the day of each month should be considered by 'switch' statement.
  - ■ Use the following algorithm to consider leap year.
    - ● if year is not divisible by 4 then common year
      else if year is not divisible by 100 then leap year
      else if year is divisible by 400 then leap year
      else common year
  - ■ Calculate the rest of dates to the end of the year from the day of you and your girl or boy friend met
  - ■ If the rest of dates are larger than 100, 200, and 300 days (those anniversaries should be in the same year that you met.), calculate the month first and days later.
  - ■ If the rest of dates are bigger than 0 (those anniversaries may be in the next year.), calculate the month first and days later.