

# Introduction to Programming with C++

Chieh-Sen (Jason) Huang

Department of Applied Mathematics

National Sun Yat-sen University

INTRODUCTION TO  
PROGRAMMING  
WITH

The logo for C++ programming language, featuring a large blue 'C' followed by two blue '+' signs.

Third Edition

Contents are based on book by Y. Daniel Liang

# Selections

- The program can decide which statements to execute based on a condition.
- Like all high-level programming languages, C++ provides selection statements: statements that let you choose actions with alternative courses.

```
if (radius < 0)
{
    cout << "Incorrect input" << endl;
}
else
{
    area = radius * radius * PI;
    cout << "The area for the circle of radius " << radius
        << " is " << area << endl;
}
```

- Selection statements use conditions that are Boolean expressions.
- A Boolean expression is an expression that evaluates to a Boolean value: true or false.

# The bool Data Type

- The bool data type declares a variable with the value either true or false.

**TABLE 3.1** Relational Operators

<i>Operator</i>	<i>Mathematics Symbol</i>	<i>Name</i>	<i>Example (radius is 5)</i>	<i>Result</i>
<	<	less than	<code>radius &lt; 0</code>	<code>false</code>
<=	≤	less than or equal to	<code>radius &lt;= 0</code>	<code>false</code>
>	>	greater than	<code>radius &gt; 0</code>	<code>true</code>
>=	≥	greater than or equal to	<code>radius &gt;= 0</code>	<code>true</code>
==	=	equal to	<code>radius == 0</code>	<code>false</code>
!=	≠	not equal to	<code>radius != 0</code>	<code>true</code>

- A variable that holds a Boolean value is known as a Boolean variable.

```
bool lightsOn = true;
```

- True and False are Boolean literals. They are keywords and cannot be used as identifiers in your program.

# The bool Data Type

- Internally, C++ uses 1 to represent true and 0 for false. If you display a bool value to the console, 1 is displayed if the value is true and 0 if it is false.

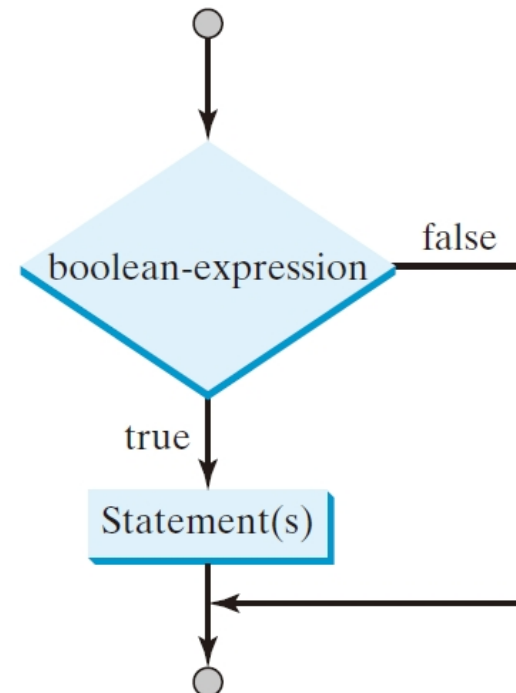
In-Class Exercise: 3.3

# if Statements

- An **if** statement is a construct that enables a program to specify alternative path of execution.
- A one-way if statement executes an action if and only if the condition is true.

```
if (boolean-expression)
{
    statement(s);
}
```

A flowchart is a diagram that describes an algorithm or process.



# SimpleIfDemo.cpp

```
int main()
{
    int number; // List3_1.cpp
    // Prompt the user to enter an integer
    cout << "Enter an integer: ";
    cin >> number;

    if (number % 5 == 0)
        cout << "HiFive" << endl;

    if (number % 2 == 0)
        cout << "HiEven" << endl;

    return 0;
}
```

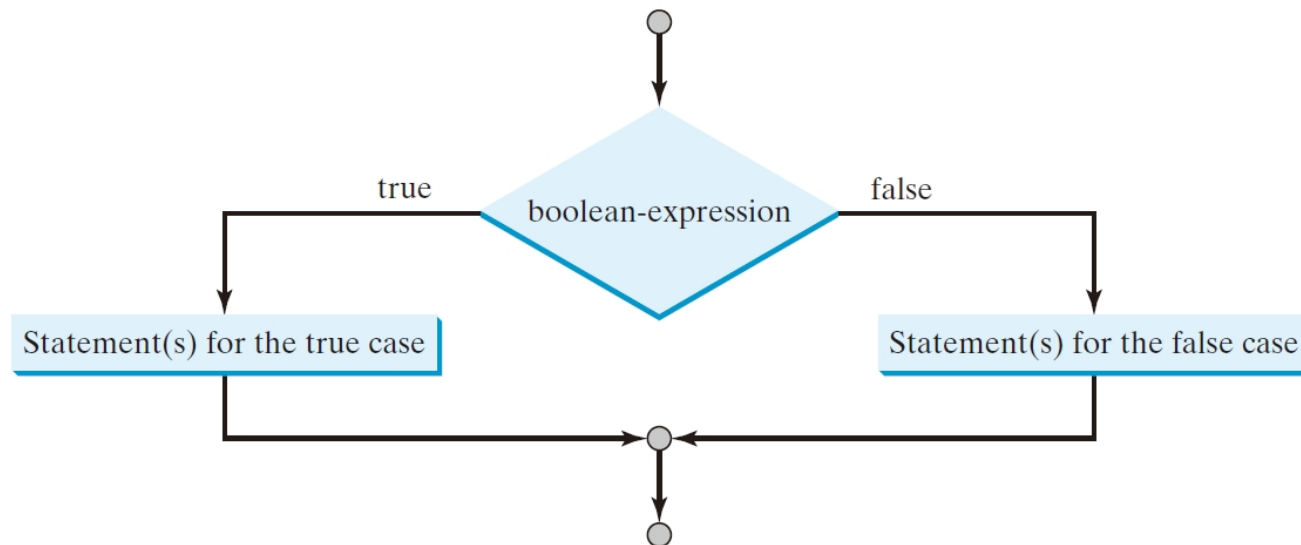
Enter a radius: 4

HiEven

# Two-Way if-else Statements

- An if-else statement decides which statements to execute based on whether the condition is true or false.

```
if (boolean-expression)
{
    statement(s)-for-the-true-case;
}
else
{
    statement(s)-for-the-false-case;
}
```



## Two-Way if-else Statements

```
if (number % 2 == 0)
    cout << number << " is even.";
else
    cout << number << " is odd.";
```

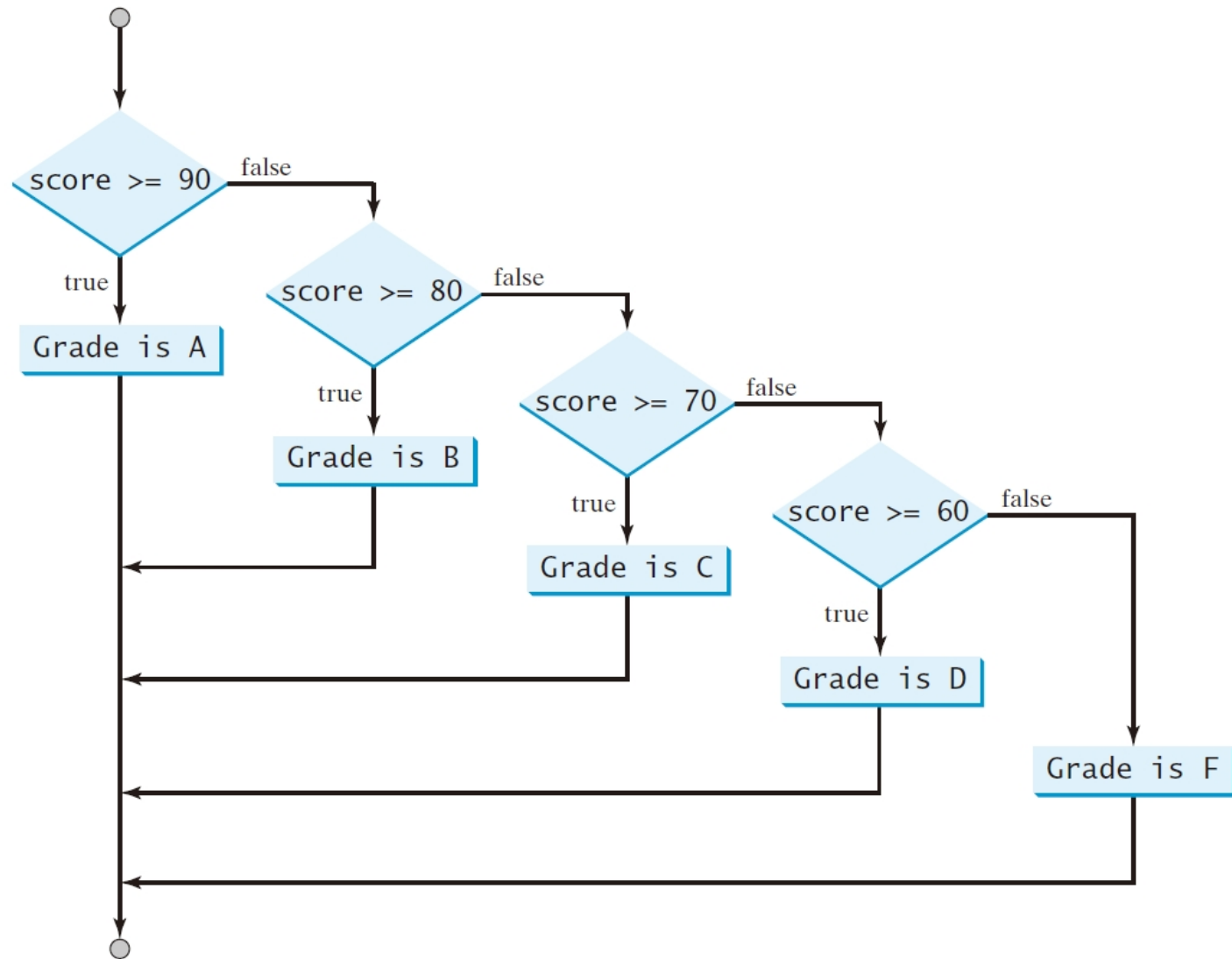
**In-Class Exercise:** Write a code to check whether a number is even or odd.

- An if statement can be inside another if statement to form a nested if statement.

```
if (score >= 90.0)
    cout << "Grade is A";
else if (score >= 80.0)
    cout << "Grade is B";
else if (score >= 70.0)
    cout << "Grade is C";
else if (score >= 60.0)
    cout << "Grade is D";
else
    cout << "Grade is F";
```



# A multi-way if-else statement



**Homework:** Write a code to decide if a year is a leap year or not.

# Logical Operators

The logical operators `!`, `&&`, and `||` can be used to create a compound Boolean expression.

(`&` ampersand, `!` exclamation mark, `|` vertical bar, pipe).

Operator	Name	Description
<code>!</code>	not	logical negation
<code>&amp;&amp;</code>	and	logical conjunction
<code>  </code>	or	logical disjunction

```
if (number % 2 == 0 && number % 3 == 0)
    cout << number << " is divisible by 2 and 3." << endl;
```

```
if (number % 2 == 0 || number % 3 == 0)
    cout << number << " is divisible by 2 or 3." << endl;
```

```
if ((number % 2 == 0 || number % 3 == 0) &&
    !(number % 2 == 0 && number % 3 == 0))
    cout << number << " divisible by 2 or 3, but not both." << endl;
```

**In-Class Exercise:** Write a code to decide the number of days for a given month number. Answer 28 or 29 if 2 (February) is entered.

**Homework:** Check-point 3.23 (page 113). Redo the leap year hw using logical operators.

# Switch Statements

- A switch statement executes statements based on the value of a variable or an expression.
- The switch-expression must yield an integral value and always be enclosed in parentheses.

```
switch (month)
{
    case 2: cout << "month number" << month<< "is 28 or 29 days." << endl;
            break;
    case 4:
    case 6:
    case 9:
    case 11: cout << "month number" << month << "is 30 days." << endl;
            break;
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12: cout << "month number" << month << "is 31 days."<<endl;
            break;
    default: cout << "Error: invalid month number" << endl;
}
```

# Conditional Expressions

- A conditional expression evaluates an expression based on a condition.
- Conditional expressions have a completely different structure and do not include an explicit if.

```
boolean-expression ? expression1 : expression2;
```

-----

```
y = x > 0 ? 1 : -1;
```

-----

```
if (x > 0)
```

```
    y = 1;
```

```
else
```

```
    y = -1;
```

-----


```
cout << (num % 2 == 0 ? "num is even" : "num is odd") << endl;
```

**In-Class Exercise:** Check-point 3.35 (page 122).

# Operator Precedence and Associativity

- Use parentheses to force an evaluation order.

**TABLE 3.7** Operator Precedence Chart

<i>Precedence</i>	<i>Operator</i>
	<b>var++</b> and <b>var--</b> (Postfix)
	<b>+</b> , <b>-</b> (Unary plus and minus), <b>++var</b> and <b>--var</b> (Prefix)
	<b>static_cast</b> <type>(v), (type) (Casting)
	<b>!</b> (Not)
	<b>*</b> , <b>/</b> , <b>%</b> (Multiplication, division, and remainder)
	<b>+</b> , <b>-</b> (Binary addition and subtraction)
	<b>&lt;</b> , <b>&lt;=</b> , <b>&gt;</b> , <b>&gt;=</b> (Relational)
	<b>==</b> , <b>!=</b> (Equality)
	<b>&amp;&amp;</b> (AND)
	<b>  </b> (OR)
	<b>=</b> , <b>+=</b> , <b>-=</b> , <b>*=</b> , <b>/=</b> , <b>%=</b> (Assignment operator)

**Homework:** Programming exercise, 3.1.