Chapter 3: Selections

Sections 3.1–3.16

Textbooks: Y. Daniel Liang, Introduction to Programming with C++, 3rd Edition

These slides were adapted by Prof. Gheith Abandah from the Computer Engineering Department of the University of Jordan for the Course: Computer Skills for Engineers (0907101)
Updated by Dr. Ashraf Suyyagh (Summer 2021)

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Introduction

If you assigned a negative value for **`radius`** in Listing 2.1, ComputeArea.cpp, the program would print an invalid result. If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

The `bool` Type and Operators

Often in a program you need to compare two values, such as whether `i` is greater than `j`. C++ provides six *relational operators* (also known as *comparison operators*):

Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	<	less than	`radius < 0`	`false`
`<=`	≤	less than or equal to	`radius <= 0`	`false`
`>`	>	greater than	`radius > 0`	`true`
`>=`	≥	greater than or equal to	`radius >= 0`	`true`
`==`	=	equal to	`radius == 0`	`false`
`!=`	≠	not equal to	`radius != 0`	`true`

The `bool` Type and Operators

A variable that holds a Boolean value is known as a *Boolean variable*, which holds **`true`** or **`false`**.

```
bool lightsOn = true;
cout << lightsOn; // Displays 1
cout << (4 < 5);  // Displays 1
cout << (4 > 5);  // Displays 0
```

Any nonzero value evaluates to **`true`** and zero value evaluates to **`false`**.

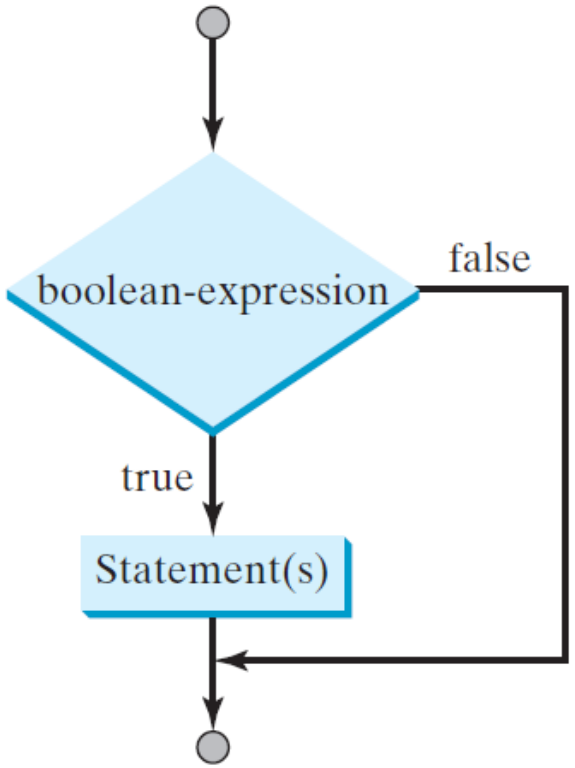
```
bool b1 = -1.5; // ≡ bool b1 = true;
bool b2 = 0;    // ≡ bool b2 = false;
bool b3 = 1.5;  // ≡ bool b3 = true;
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

One-way if Statements

```
if (booleanExpression)
{
  statement(s);
}
```




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

Notes

- The boolean-expression must be enclosed in parentheses.

```
if i > 0
{
  cout << "i is positive" << endl;
}
```

(a) Wrong

```
if (i > 0)
{
  cout << "i is positive" << endl;
}
```

(b) Correct

- The braces can be omitted if they enclose a single statement.

```
if (i > 0)
{
  cout << "i is positive" << endl;
}
```

(a)

Equivalent

```
if (i > 0)
  cout << "i is positive" << endl;
```

(b)

Simple if Demo

A program that prompts the user to enter an integer. If the number is a multiple of **5**, displays **HiFive**. If the number is even, displays **HiEven**.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user to enter an integer
7    int number;
8    cout << "Enter an integer: ";
9    cin >> number;
10
11   if (number % 5 == 0)
12     cout << "HiFive" << endl;
13
14   if (number % 2 == 0)
15     cout << "HiEven" << endl;
16
17   return 0;
18 }
```

SimpleIfDemo

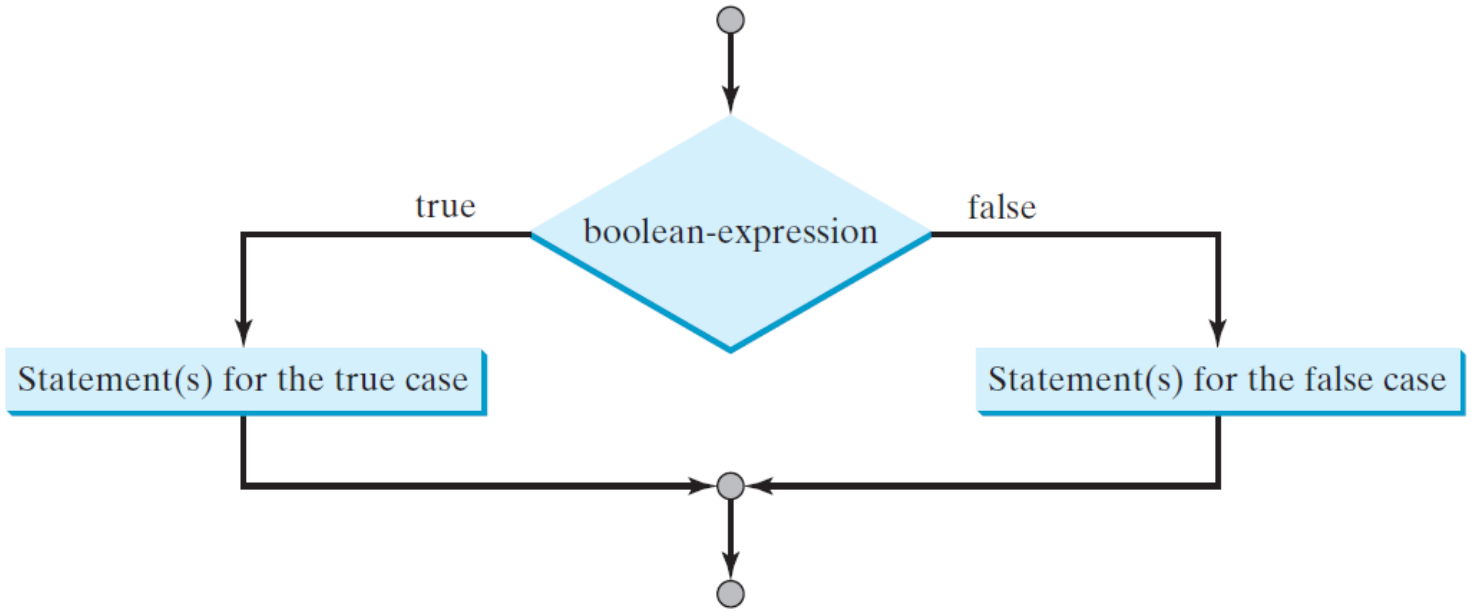
Run

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Two-Way `if-else` Statement

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



Examples

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

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

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Nested `if` Statements

You can nest multiple **`if`** statements

```
if (i > k)
{
  if (j > k)
    cout << "i and j are greater than k";
}
else
  cout << "i is less than or equal to k";
```

Multiple Alternative `if` Statements

```
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)

Equivalent

This is better

```
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";
```

(b)

animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
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";
```

animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
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";
```

animation

Trace if-else statement

Suppose score is 70.0

The condition is true

```
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";
```

animation

Trace if-else statement

Suppose score is 70.0

grade is C

```
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";
```

animation

Trace if-else statement

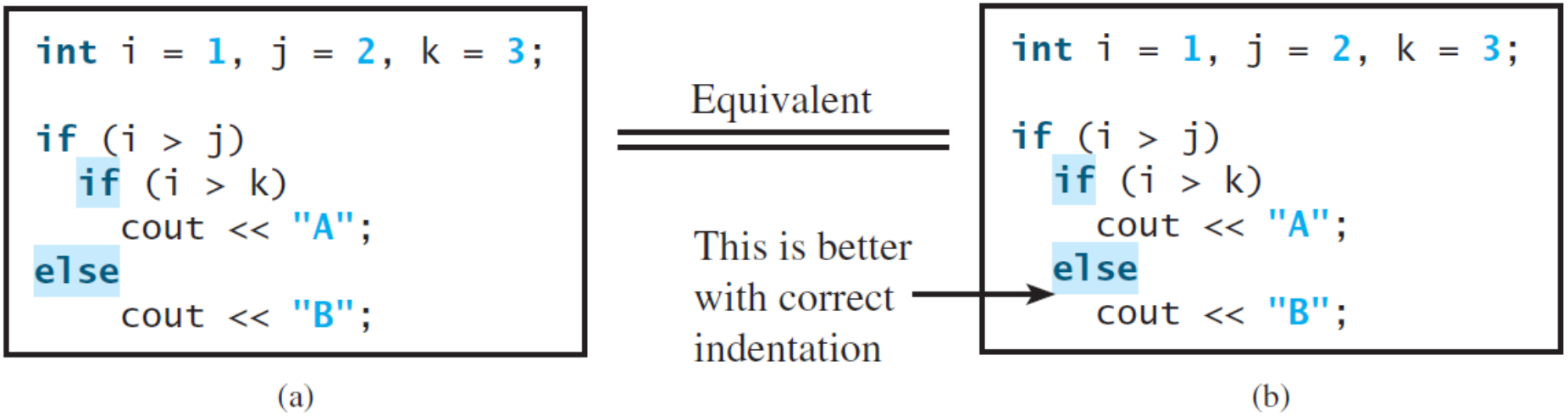
Suppose score is 70.0

Exit the 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";
```

Note

The **`else`** clause matches the most recent **`if`** clause in the same block.



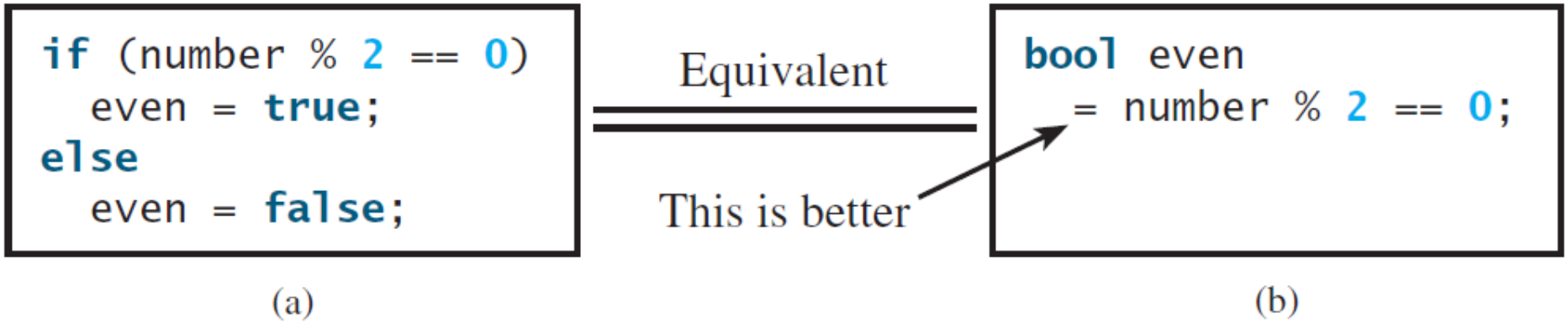
Note, cont.

Nothing is printed from the Statement (a) above. To force the **`else`** clause to match the first **`if`** clause, you must add a pair of braces:

```
int i = 1, j = 2, k = 3;
if (i > j)
{
  if (i > k)
    cout << "A";
}
else
  cout << "B";
```

This statement prints **B**.

TIP


if (number % 2 == 0)
even = true;
else
even = false;
Equivalent
This is better
bool even
= number % 2 == 0;
(a)
(b)

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Common Errors

1: Forgetting Necessary Braces

```
if (radius >= 0)
  area = radius * radius * PI;
  cout << "The area "
    << " is " << area;
```

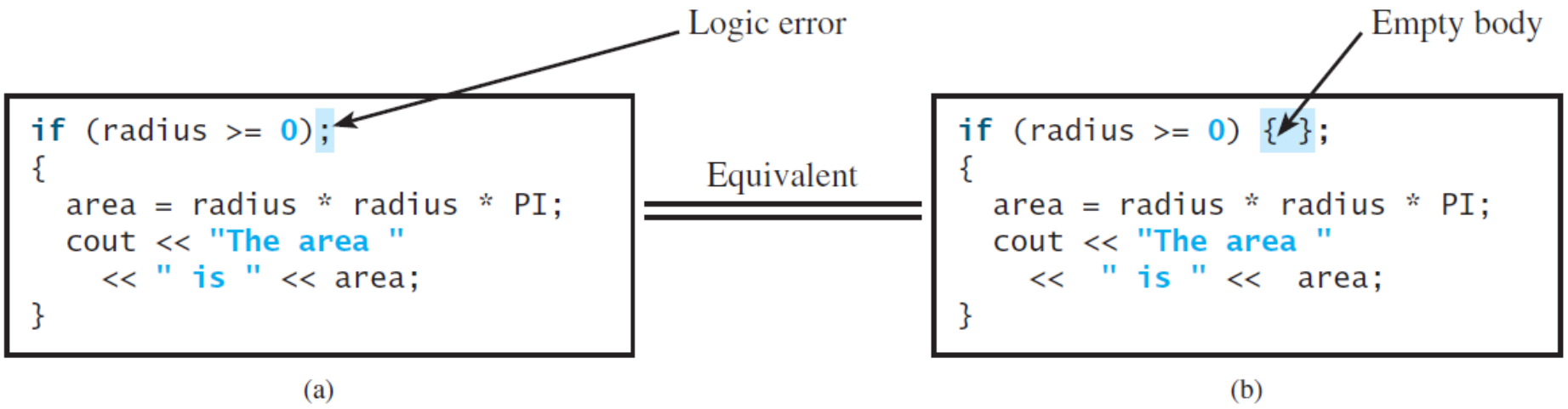
(a) Wrong

```
if (radius >= 0)
{
  area = radius * radius * PI;
  cout << "The area "
    <<  " is " <<  area;
}
```

(b) Correct

Common Errors

2: Wrong Semicolon at the if Line



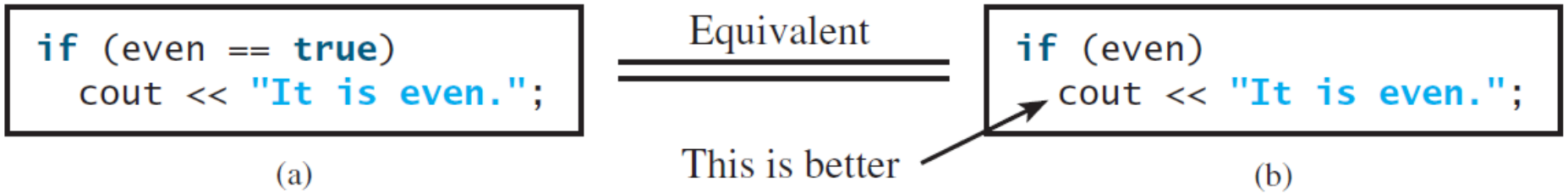
Common Errors

3: Mistakenly Using = for ==

```
if (count = 1)
   cout << "count is zero" << endl;
else
   cout << "count is not zero" << endl;
```

Common Errors

4: Redundant Testing of Boolean Values



Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Case Study: Body Mass Index

The Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters ($BMI = {}^{m}/_{h^2}$). The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
$BMI < 18.5$	Underweight
$18.5 \leq BMI < 25.0$	Normal
$25.0 \leq BMI < 30.0$	Overweight
$30.0 \leq BMI$	Obese

ComputeBMI

Run

Case Study: Body Mass Index

```
double bmi = weightInKilograms /
  (heightInMeters * heightInMeters);

// Display result
cout << "BMI is " << bmi << endl;
if (bmi < 18.5)
  cout << "Underweight" << endl;
else if (bmi < 25)
  cout << "Normal" << endl;
else if (bmi < 30)
  cout << "Overweight" << endl;
else
  cout << "Obese" << endl;
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Self-Study Example: Computing Taxes

The Jordan income tax is calculated based on the filing status and taxable income. In this ***simplified*** example, there are three filing statuses: single filers, married couple filing jointly, and business filing.

The taxable income is calculated as follows:

- Single filers taxable income = total income from all sources – 9000 JDs exemption
- Married filers taxable income = total income from all sources – 18000 JDs exemption – 1000JDs for each child *up to* three children

The businesses are divided into three categories:

1. (Group 1) Banking sector
2. (Group 2) Insurance, communication, mining, power generation companies.
3. (Group 3) Others

SELF-STUDY Examples are not explained in class because they are very long examples. Students are advised to watch the video recorded and provided by the professor

Self-Study Example: Computing Taxes

- The tax rates for 2020 and beyond are shown below:

Tax Bracket	Single, Married Filers
0 – 5,000	5%
5,001 – 10,000	10%
10,001 – 15,000	15%
15,001 – 20,000	25%
20,001- 1,000,000	30%

Business	Tax on net profit
Group 1	35%
Group 2	24%
Group 3	20%

Self-Study Example: Computing Taxes: Skeleton Code

```
if (income > 0){
    if (status == 0)
    {
      // Compute tax for single filers
    }
    else if (status == 1)
    {
      // Compute tax for married file jointly
    }
    else if (status == 2)
    {
      // Compute tax for business
    }
}
```

Self-Study Example: Computing Taxes: First Case Details

```
if (status == 0)
  {
    double taxableIncome = income - 9000;
    // Compute tax for single filers
    if (taxableIncome <= 5000 && taxableIncome >= 0)
      tax = taxableIncome * 0.05;
    else if (taxableIncome <= 10000)
      tax = 5000 * 0.05 + (taxableIncome - 5000) * 0.10;
    else if (taxableIncome <= 15000)
      tax = (5000 * 0.05) + (5000 * 0.10) +
            (taxableIncome - 10000) * 0.15;
    else if (taxableIncome <= 20000)
       …
  }
  else if (status == 1)
```

Self-Study Example: Computing Taxes: Second Case Details

```
if (status == 1)
  {
    int noChild = min(3, children)
    double taxableIncome = income - 18000 - 1000*noChild;
    // Compute tax for married filing jointly
    if (taxableIncome <= 5000 && taxableIncome >= 0)
      tax = taxableIncome * 0.05;
    else if (taxableIncome <= 10000)
      tax = 5000 * 0.05 + (taxableIncome - 5000) * 0.10;
    else if (taxableIncome <= 15000)
      tax = (5000 * 0.05) + (5000 * 0.10) +
            (taxableIncome - 10000) * 0.15;
    else if (taxableIncome <= 20000)
      ...
  }
  else if (status == 2)
```

Self-Study Example: Computing Taxes: Third Case Details

```
if (status == 2)
   {
       //Divide into three business groups
       if (group == 1)
          tax = profit * 0.35;
       else if (group == 2)
          tax = profit * 0.24;
       else if (group == 3)
          tax = profit * 0.20;
    }
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Generating Random Numbers

- You can use the **`rand()`** function to obtain a random integer.
- This function returns a random integer between 0 and RAND_MAX (32,767 in Visual C++).
- To start with a different seed at each execution, use

```
srand(time(0));
```

- To obtain a random integer between 0 and 9, use

```
rand() % 10
```

Example: A Simple Math Learning Tool

- This example creates a program for a first grader to practice subtractions.
- The program randomly generates two single-digit integers **`number1`** and **`number2`** with **`number1 >= number2`** and displays a question such as **"What is 9 – 2?"** to the student.
- After the student types the answer, the program displays a message to indicate whether the answer is correct.

SubtractionQuiz Run

SubtractQuiz.cpp 1/2

```
#include <iostream>
#include <ctime> // for time function
#include <cstdlib> // for rand and srand functions
using namespace std;

int main()
{
  // 1. Generate two random single-digit integers
  srand(time(0));
  int number1 = rand() % 10;
  int number2 = rand() % 10;

  // 2. If number1 < number2, swap number1 with number2
  if (number1 < number2)
  {
    int temp = number1;
    number1 = number2;
    number2 = temp;
  }
```

SubtractQuiz.cpp 2/2

```
  // 3. Ask the student “what is number1 – number2?”
  cout << "What is " << number1 << " - " << number2 << "? ";
  int answer;
  cin >> answer;

  // 4. Grade the answer and display the result
  if (number1 - number2 == answer)
    cout << "You are correct!";
  else
    cout << "Your answer is wrong.\n“
         << number1 << " - “ << number2
         << " should be " << (number1 - number2) << endl;

  return 0;
}
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Logical Operators

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

TABLE 3.3 Boolean Operators

Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction

TABLE 3.4 Truth Table for Operator `!`

p	*!p*	*Example (assume age = 24, weight = 140)*
`true`	`false`	`!(age > 18)` is `false`, because `(age > 18)` is `true`.
`false`	`true`	`!(weight == 150)` is `true`, because `(weight == 150)` is `false`.

TABLE 3.5 Truth Table for Operator `&&`

p1	*p2*	*p1 && p2*	*Example (assume age = 24, weight = 140)*
`false`	`false`	`false`	`(age > 18) && (weight <= 140)` is `true`, because
`false`	`true`	`false`	`(age > 18)` and `(weight <= 140)` are both `true`.
`true`	`false`	`false`	`(age > 18) && (weight > 140)` is `false`, because
`true`	`true`	`true`	`(weight > 140)` is `false`.

TABLE 3.6 Truth Table for Operator `||`

p1	*p2*	*p1 || p2*	*Example (assume age = 24, weight = 140)*
`false`	`false`	`false`	`(age > 34) || (weight <= 140)` is `true`, because
`false`	`true`	`true`	`(weight <= 140)` is `true`.
`true`	`false`	`true`	`(age > 34) || (weight >= 150)` is `false`, because
`true`	`true`	`true`	`(age > 34)` and `(weight >= 150)` are both `false`.

Examples

A program that checks whether a number is divisible by 2 and 3, whether a number is divisible by 2 or 3, and whether a number is divisible by 2 or 3 but not both:

TestBooleanOperators

Run

TestBooleanOperators.cpp

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

int main()
{
  int number;
  cout << "Enter an integer: ";
  cin >> number;

  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;

  return(0);
}
```

Short-Circuit Operator

- When evaluating **`p1 && p2`**, C++ first evaluates **`p1`** and then evaluates **`p2`** if **`p1`** is **`true`**; if **`p1`** is **`false`**, it does not evaluate **`p2`**.
- When evaluating **`p1 || p2`**, C++ first evaluates **`p1`** and then evaluates **`p2`** if **`p1`** is **`false`**; if **`p1`** is **`true`**, it does not evaluate **`p2`**.
- Therefore, **`&&`** is referred to as the *conditional* or *short-circuit AND* operator, and **`||`** is referred to as the *conditional* or *short-circuit OR* operator.

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Case Study: Determining Leap Year

A program that lets the user enter a year and checks whether it is a leap year.

A year is a *leap year* if it is divisible by 4 but not by 100 or if it is divisible by 400. So you can use the following Boolean expression to check whether a year is a leap year:

```
(year % 4 == 0 && year % 100 != 0) ||
(year % 400 == 0)
```

LeapYear Run

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Case Study: Lottery

Randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:

- If the user input matches the lottery in exact order, the award is $10,000.
- If the user input matches the lottery, the award is $3,000.
- If one digit in the user input matches a digit in the lottery, the award is $1,000.

Lottery Run

Lottery.cpp 1/2

```
#include <iostream>
#include <ctime>   // for time function
#include <cstdlib> // for rand and srand functions
using namespace std;

int main()
{
  // Generate a lottery
  srand(time(0));
  int lottery = rand() % 100;

  // Prompt the user to enter a guess
  cout << "Enter your lottery pick (two digits): ";
  int guess;
  cin >> guess;
```

Lottery.cpp 1/2

```
  // Check the guess
  if (guess  == lottery)
    cout << "Exact match: you win $10,000" << endl;
  else if (guess % 10 == lottery / 10
      && guess / 10 == lottery % 10)
    cout << "Match all digits: you win $3,000" << endl;
  else if (guess % 10 == lottery / 10
        || guess % 10 == lottery % 10
        || guess / 10 == lottery / 10
        || guess / 10 == lottery % 10)
    cout << "Match one digit: you win $1,000" << endl;
  else
    cout << "Sorry, no match" << endl;

  return 0;
}
```

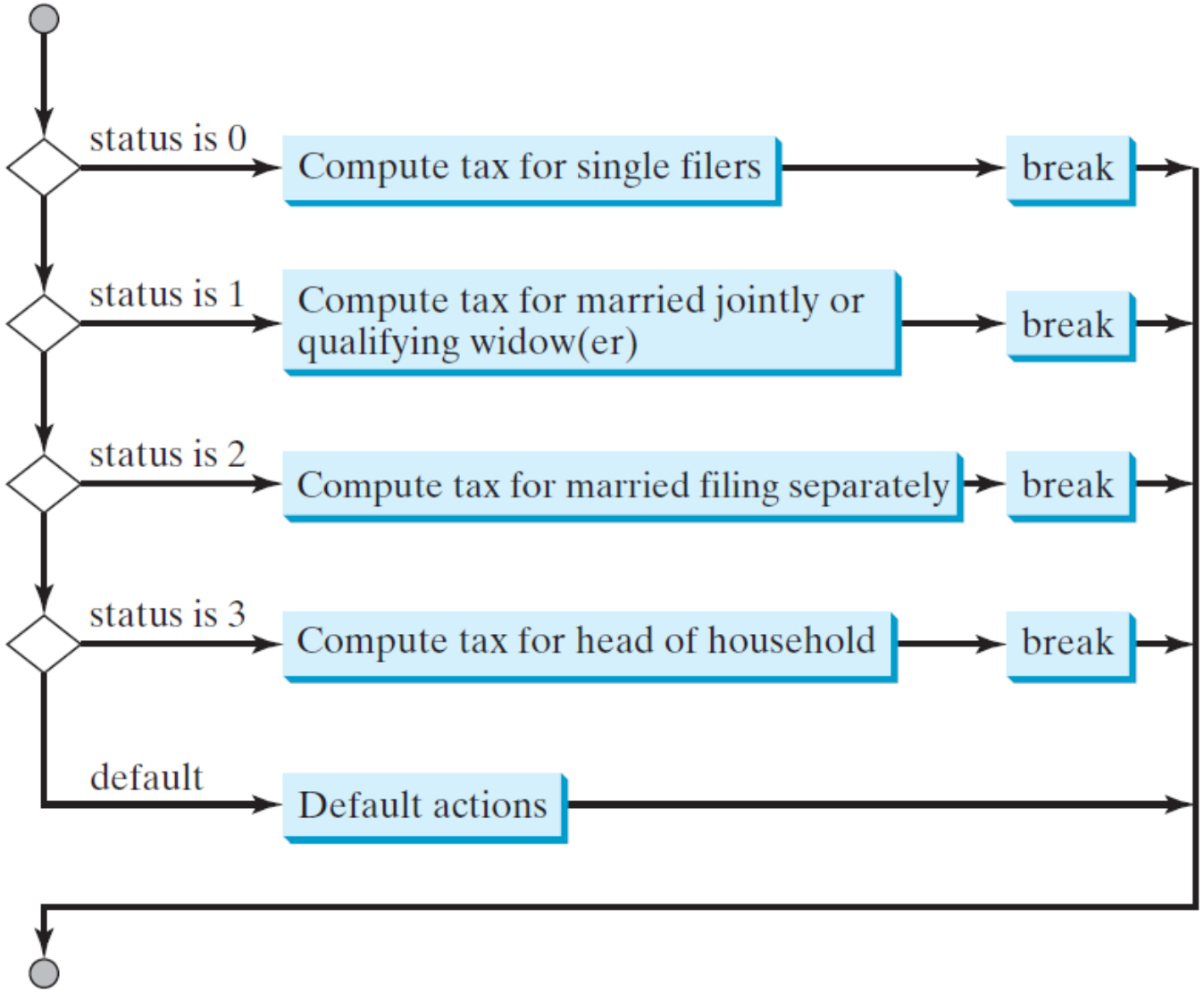
Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

`switch` Statements

```
switch (status)
{
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: cout << "Errors: invalid status" << endl;
}
```

`switch` Statement Flow Chart



`switch` Statement Rules

The switch-expression must yield a integral value and must always be enclosed in parentheses.

```
switch (switch-expression)
{
  case value1: statement(s)1;
               break;
  case value2: statement(s)2;
               break;
  ...
  case valueN: statement(s)N;
               break;
  default:     statement(s)-for-default;
}
```

The case values must be integral constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

`switch` Statement Rules

The break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement.

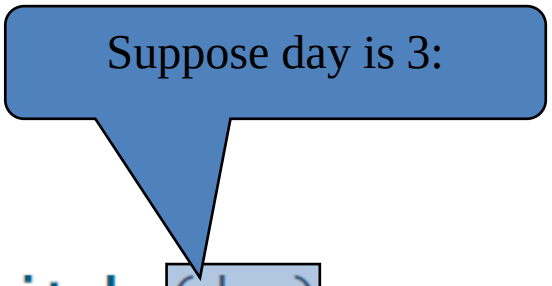
```
switch (switch-expression)
{
  case value1: statement(s)1;
               break;
  case value2: statement(s)2;
               break;
  ...
  case valueN: statement(s)N;
               break;
  default:     statement(s)-for-default;
}
```

The default case, which is optional, can be used to perform actions when none of the specified cases is executed.

When the value in a case statement matches the value of the switch-expression, the statements starting from this case are executed until either a break statement or the end of the switch statement is reached.

animation

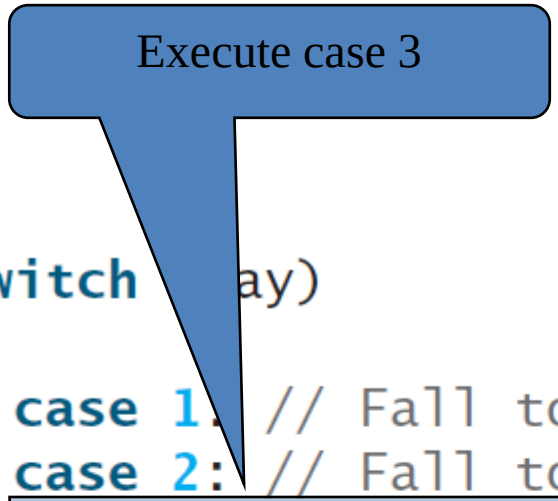
Trace `switch` statement




```
switch (day)
{
   case 1: // Fall to through to the next case
   case 2: // Fall to through to the next case
   case 3: // Fall to through to the next case
   case 4: // Fall to through to the next case
   case 5: cout << "Weekday"; break;
   case 0: // Fall to through to the next case
   case 6: cout << "Weekend";
}
```

animation

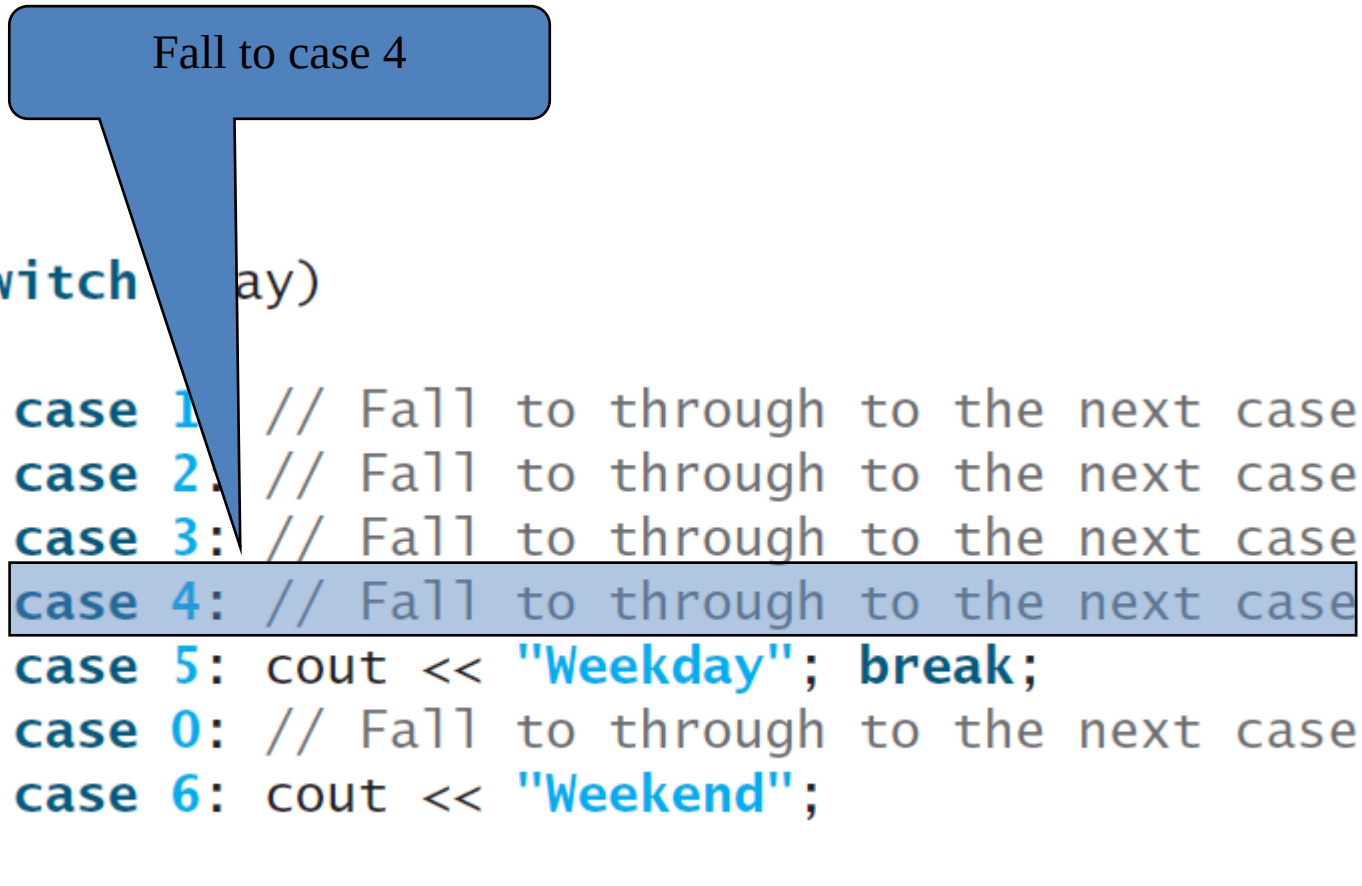
Trace `switch` statement




```
switch (day)
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```

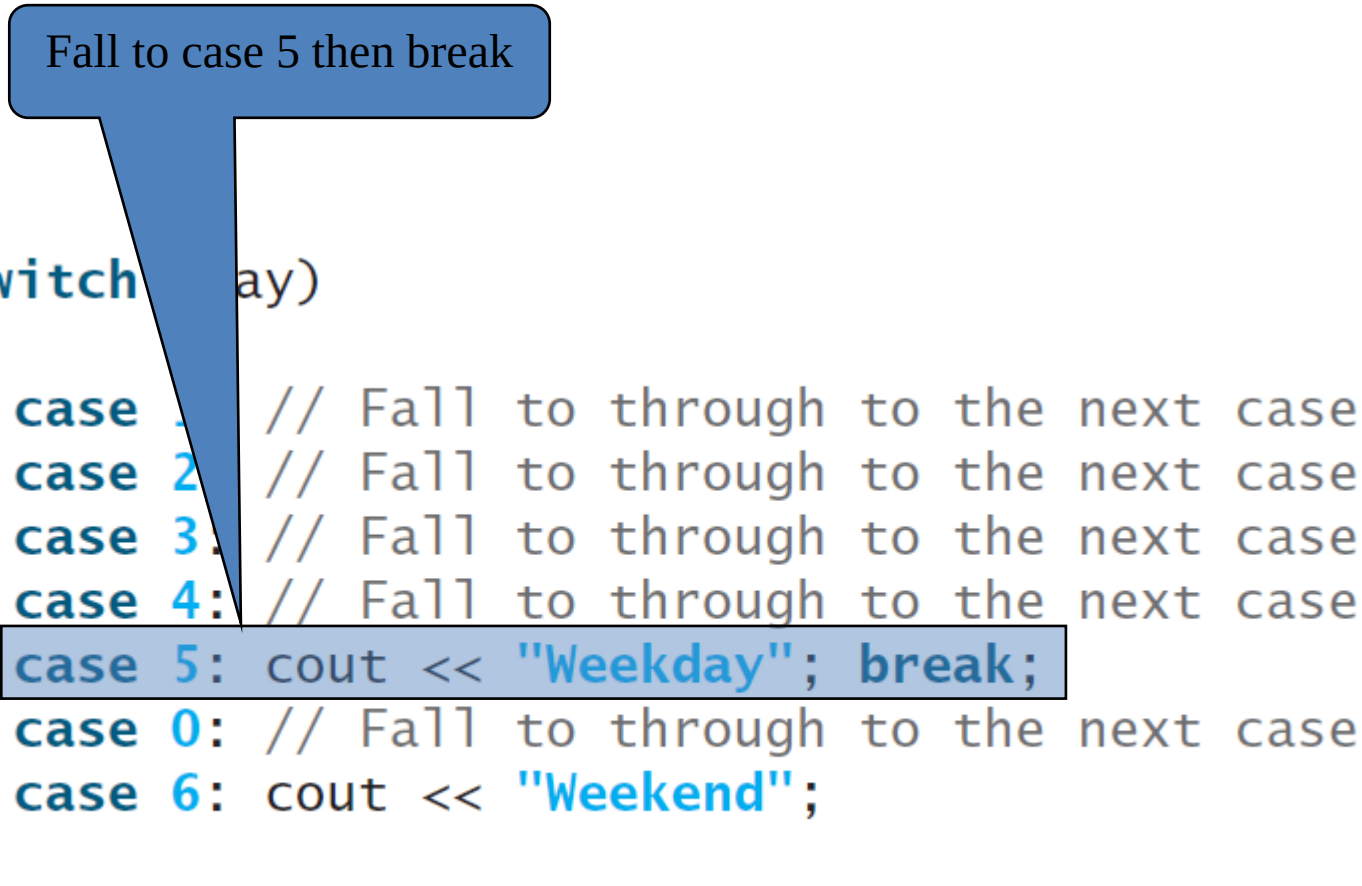
animation

Trace `switch` statement



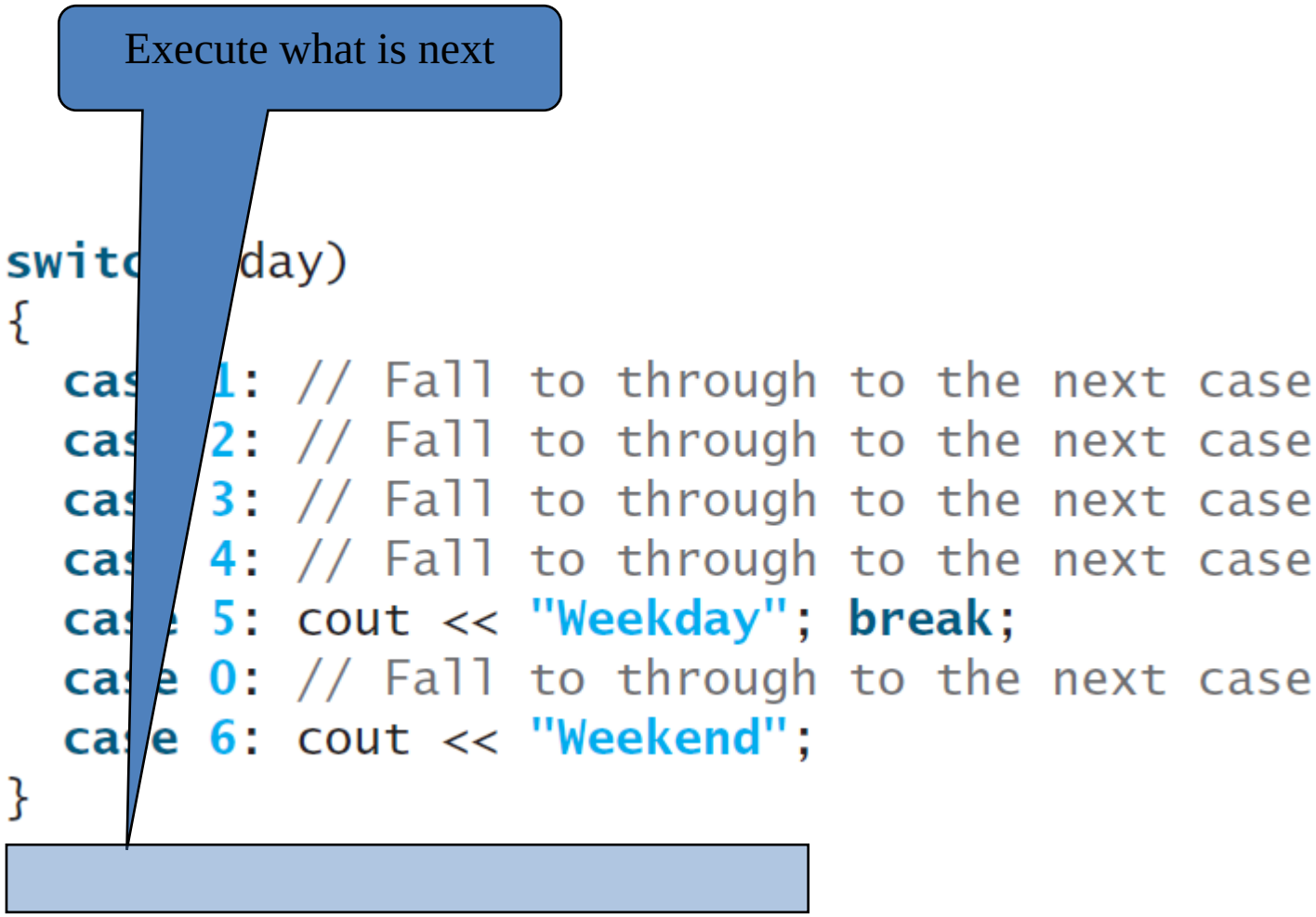
animation

Trace `switch` statement



animation

Trace `switch` statement



Example: Chinese Zodiac

A program that prompts the user to enter a year and displays the animal for the year.

year % 12 =

0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep

ChineseZodiac

Run

ChineseZodiac.cpp

```
 8    cin >> year;
 9
10    switch (year % 12)
11    {
12      case 0: cout << "monkey" << endl; break;
13      case 1: cout << "rooster" << endl; break;
14      case 2: cout << "dog" << endl; break;
15      case 3: cout << "pig" << endl; break;
16      case 4: cout << "rat" << endl; break;
17      case 5: cout << "ox" << endl; break;
18      case 6: cout << "tiger" << endl; break;
19      case 7: cout << "rabbit" << endl; break;
20      case 8: cout << "dragon" << endl; break;
21      case 9: cout << "snake" << endl; break;
22      case 10: cout << "horse" << endl; break;
23      case 11: cout << "sheep" << endl; break;
24    }
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Conditional Expressions

A conditional expression evaluates an expression based on a condition.

Syntax:
(booleanExpression) ? expression1 : expression2

The result of this conditional expression is expression1 if boolean-expression is true; otherwise, the result is expression2.

Examples

- Equivalent statements:

```
if (x > 0)
  y = 1;
else
  y = -1;
```

≡

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

- Finding the max:

```
max = num1 > num2 ? num1 : num2;
```

- Odd of even:

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

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Operator Precedence and Associativity

Operator precedence and associativity determine the order in which operators are evaluated.

How to evaluate 3 + 4 * 4 > 5 * (4 + 3) – 1?

false?

3 + 4 * 4 > 5 * (4 + 3) – 1 && (4 – 3 > 5)?

false?

Operator Precedence

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

Operator Associativity

- All binary operators except assignment operators are *left associative*.
- Assignment operators are *right associative*.

```
a - b + c - d   is equivalent to   ((a - b) + c) - d

a = b += c = 5   is equivalent to   a = (b += (c = 5))
```

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging

Debugging

- Debugging is the process of finding and fixing errors in a program.
- Visual Studio supports debugging:
  - Executing a single statement at a time
  - Tracing into or stepping over a function
  - Setting breakpoints
  - Displaying variables
  - Displaying call stacks
  - Modifying variables
- Show demo on Visual Studio 2019.

Outline

- Introduction
- The bool Data Type
- if Statements
- Two-Way if-else Statements
- Nested if and Multi-Way if-else Statements
- Common Errors and Pitfalls
- Case Study: Computing Body Mass Index
- Case Study: Computing Taxes
- Generating Random Numbers
- Logical Operators
- Case Study: Determining Leap Year
- Case Study: Lottery
- switch Statements
- Conditional Expressions
- Operator Precedence and Associativity
- Debugging