Making Decisions

Motivations

- If you assigned a negative value for radius in **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?

The boolean Type and Operators

- Often in a program you need to compare two values, such as

  - whether i > j or not?
  - whether radius > 0 or not?

- C++ provides six *comparison operators* (also known as *relational operators*) that can be used to compare two values.

4.1

Relational Operators

Relational Operators

- Used to compare numbers to determine relative order
- Operators:

$(x > y)$

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Relational Operators

Operator	Name	Example	Result
<	less than	1 < 2	true
<=	less than or equal to	1 <= 2	true
>	greater than	1 > 2	false
>=	greater than or equal to	1 >= 2	false
==	equal to	1 == 2	false
!=	not equal to	1 != 2	true

Relational Expressions

- Boolean expressions – `true` or `false`

- Examples:

  `12 > 5` is `true`

  `7 <= 5` is `false`

  if `x` is 10, then

  `x == 10` is `true`,

  `x != 8` is `true`, and

  `x == 8` is `false`

Relational Expressions

- Can be assigned to a variable:

```
result = x <= y;
```

- Assigns `0` for `false`, `1` for `true`

x = 10;

- Do not confuse = and ==

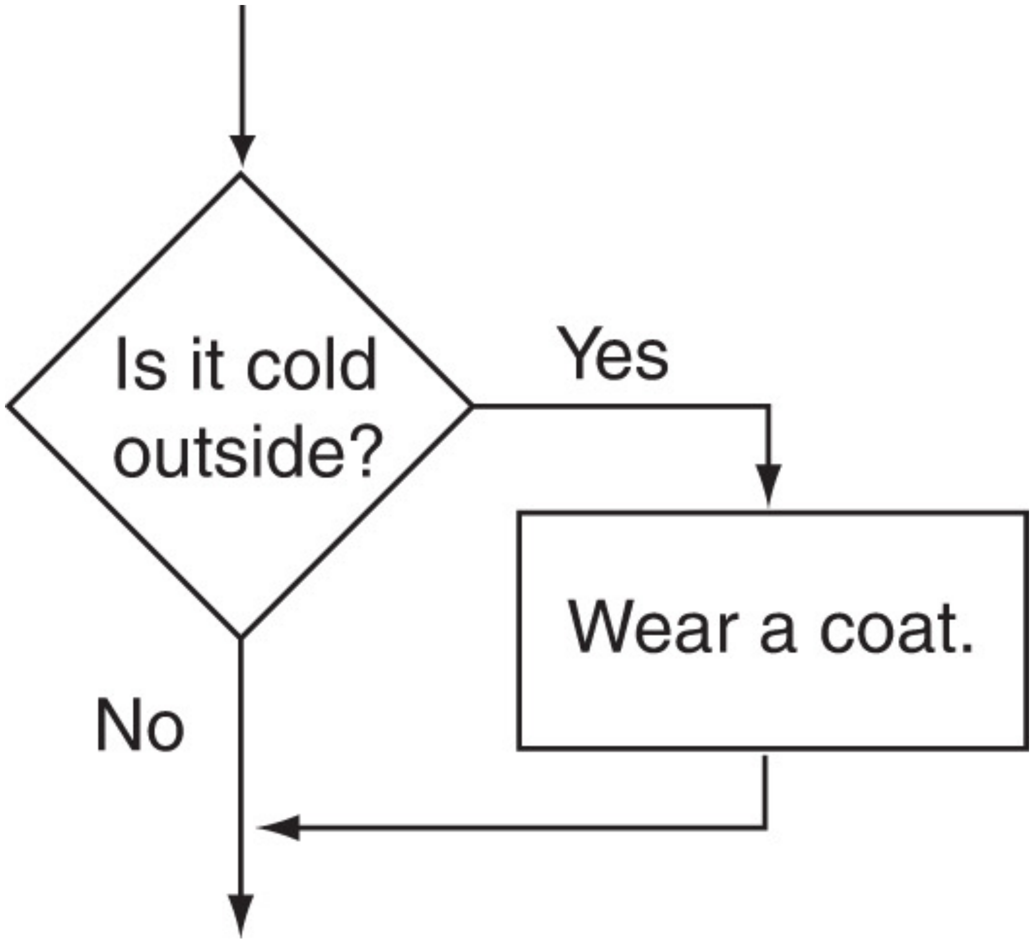
result = x

result == x

4.2

The `if` Statement

Flowchart for Evaluating a Decision

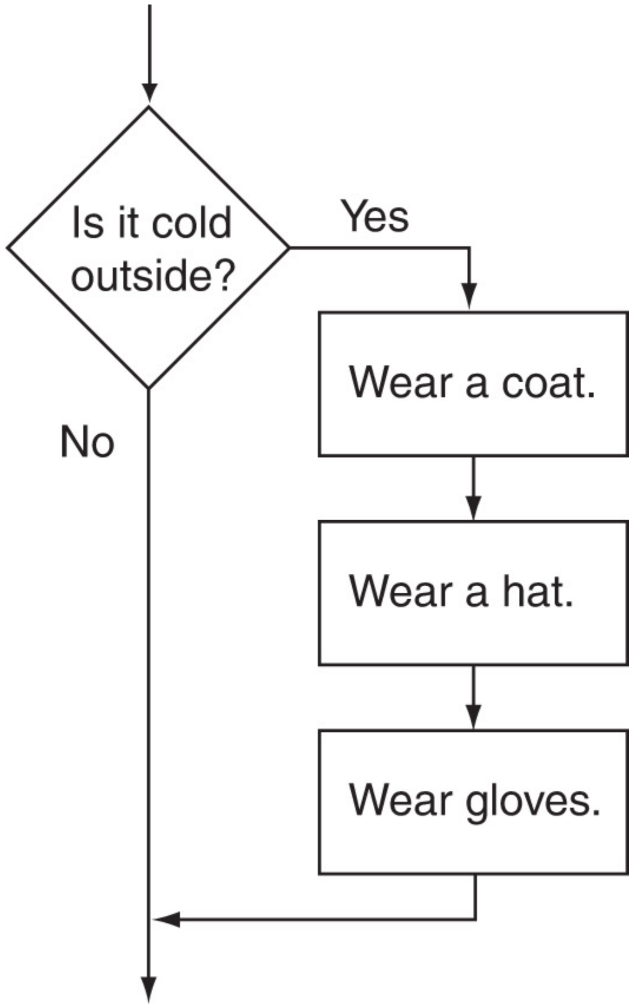


The `if` Statement

- Allows statements to be conditionally executed or skipped over

- Models the way we mentally evaluate situations:
  - "If it is raining, take an umbrella."
  - "If it is cold outside, wear a coat."

Flowchart for Evaluating a Decision



The `if` Statement

- General Format:

```
if (expression)
      statement;
```

```
if( expn)
{
   S1;
   S2;
   S3;
}
```

The if Statement-What Happens

To evaluate:

```
    if (expression)
        statement;
```

- If the `expression` is `true`, then `statement` is executed.
- If the `expression` is `false`, then `statement` is skipped.

if Statement in Program 4-2

Program 4-2

```
// This program averages three test scores
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int score1, score2, score3;   // To hold three test scores
   double average;               // To hold the average score

```

Continued…

`if` Statement in Program 4-2

Program 4-2 *(continued)*

```
    // Get the three test scores.
    cout << "Enter 3 test scores and I will average them: ";
    cin >> score1 >> score2 >> score3;

    // Calculate and display the average score.
    average = (score1 + score2 + score3) / 3.0;
    cout << fixed << showpoint << setprecision(1);
    cout << "Your average is " << average << endl;

    // If the average is greater than 95, congratulate the user.
    if (average > 95)
       cout << "Congratulations! That's a high score!\n";
    return 0;
}
```

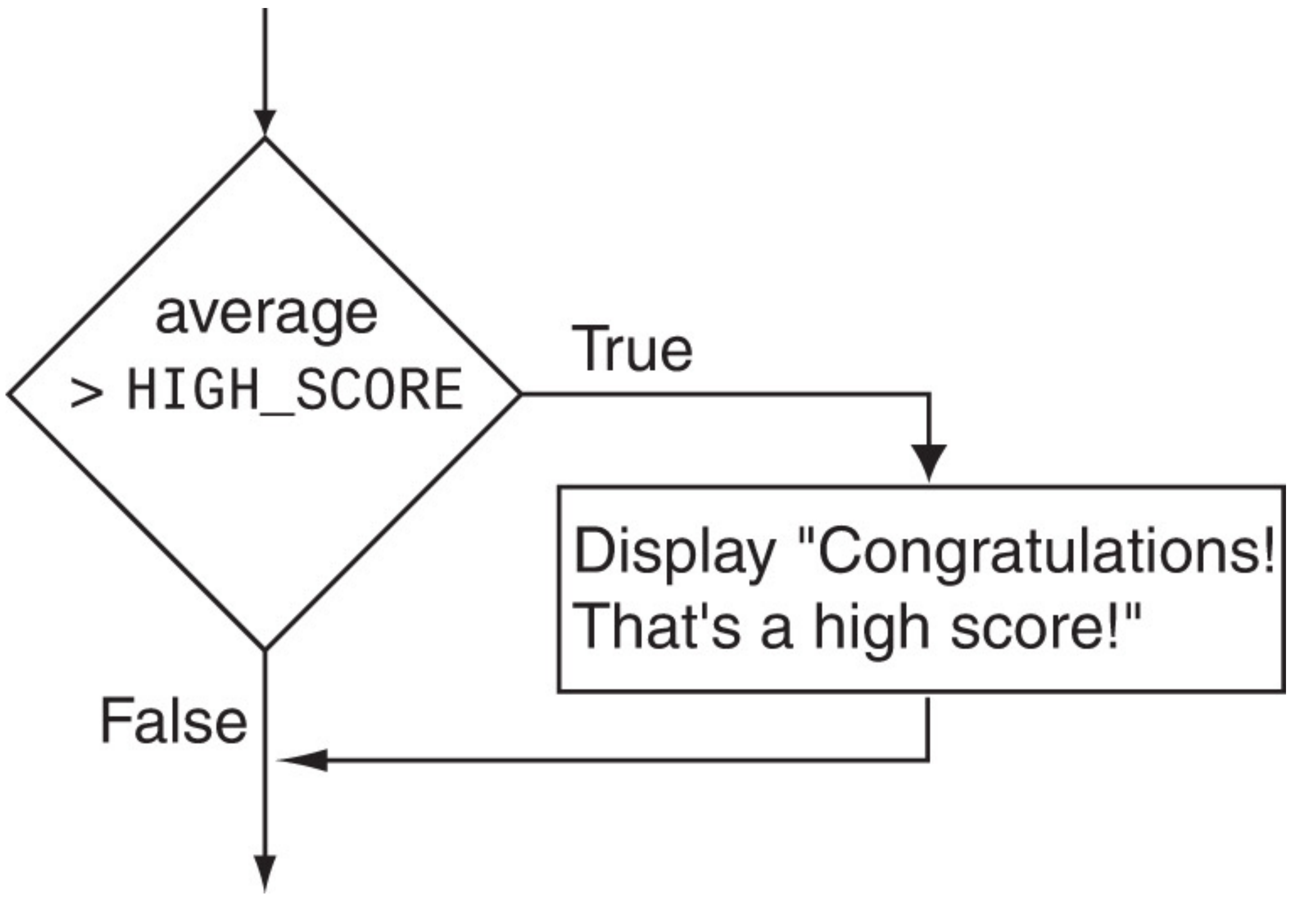
Program Output with Example Input Shown in Bold

```
Enter 3 test scores and I will average them: 80 90 70 [Enter]
Your average is 80.0
```

Program Output with Other Example Input Shown in Bold

```
Enter 3 test scores and I will average them: 100 100 100 [Enter]
Your average is 100.0
Congratulations! That's a high score!
```

Flowchart for Program 4-2 Lines 21 and 22



`if` Statement Notes

- Do not place `;` after `(`*`expression`*`)`

- Place *`statement;`* on a separate line after *`expression`*`),` indented:

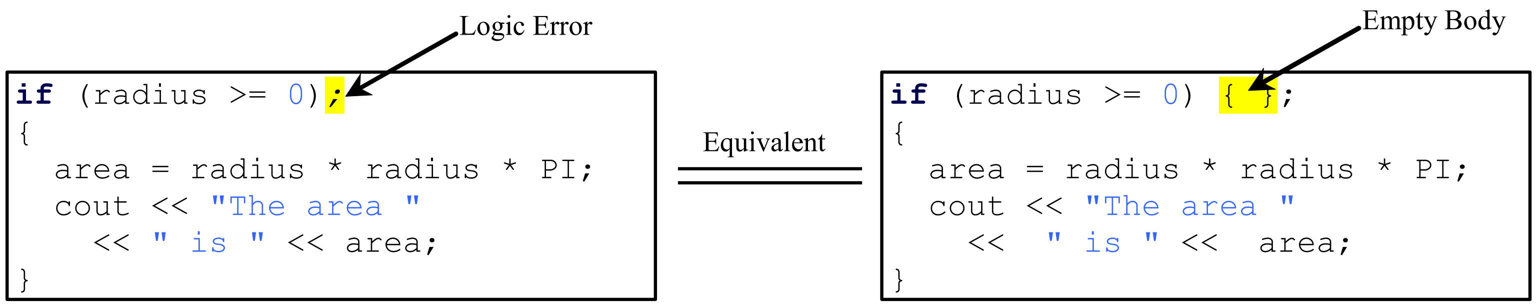
```
    if (score > 90)
        grade = 'A';
```

- Be careful testing `float`s and `double`s for equality

- `0` is `false`; any other value is `true`

Caution

- Adding a semicolon at the end of an if clause is a common mistake.



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

(a)

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

(b)

- This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.
- This error often occurs when you use the next-line block style.

Note

Outer parentheses required

```
if ((i > 0) && (i < 10))
{
  cout << "i is an " <<
    "integer between 0 and 10";
}
```

(a)

Equivalent

Braces can be omitted if the block contains a single statement

```
if ((i > 0) && (i < 10))
  cout << "i is an " <<
    "integer between 0 and 10";
```

(b)

4.3

Expanding the `if` Statement

Expanding the `if` Statement

- To execute more than one statement as part of an `if` statement, enclose them in `{ }`:

```
if (score > 90)
{
    grade = 'A';
    cout << "Good Job!\n";
}
```

- `{ }` creates a <u>block</u> of code

- 4.3 Misplaced semicolon
- 4.4 Floating point comparison
- 4.5 Assignment operator
- 4.6 Several statements enclosing in {}
- 4.7 What if your forget {}

Classroom Exercise

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

What should be the result if the input is 4?

What should be the result if the input is 30?

4.4

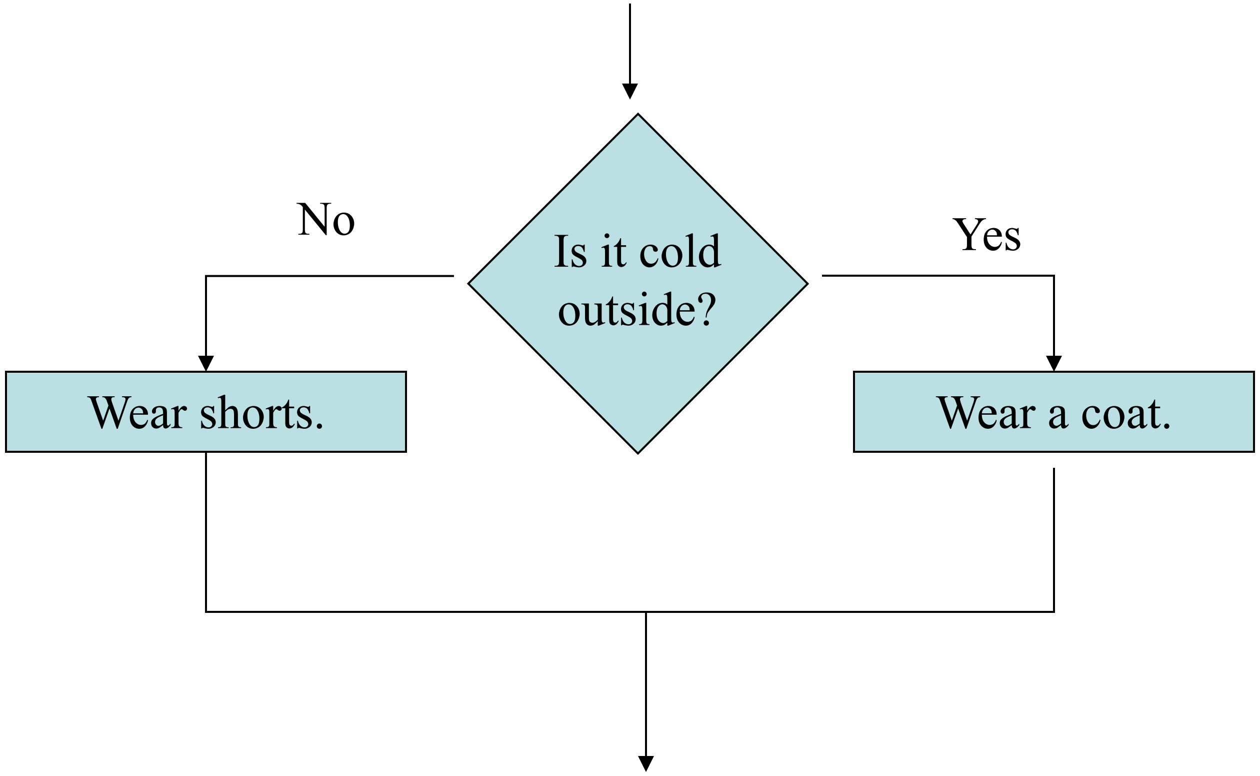
The `if/else` Statement

The `if/else` statement

- Provides two possible paths of execution

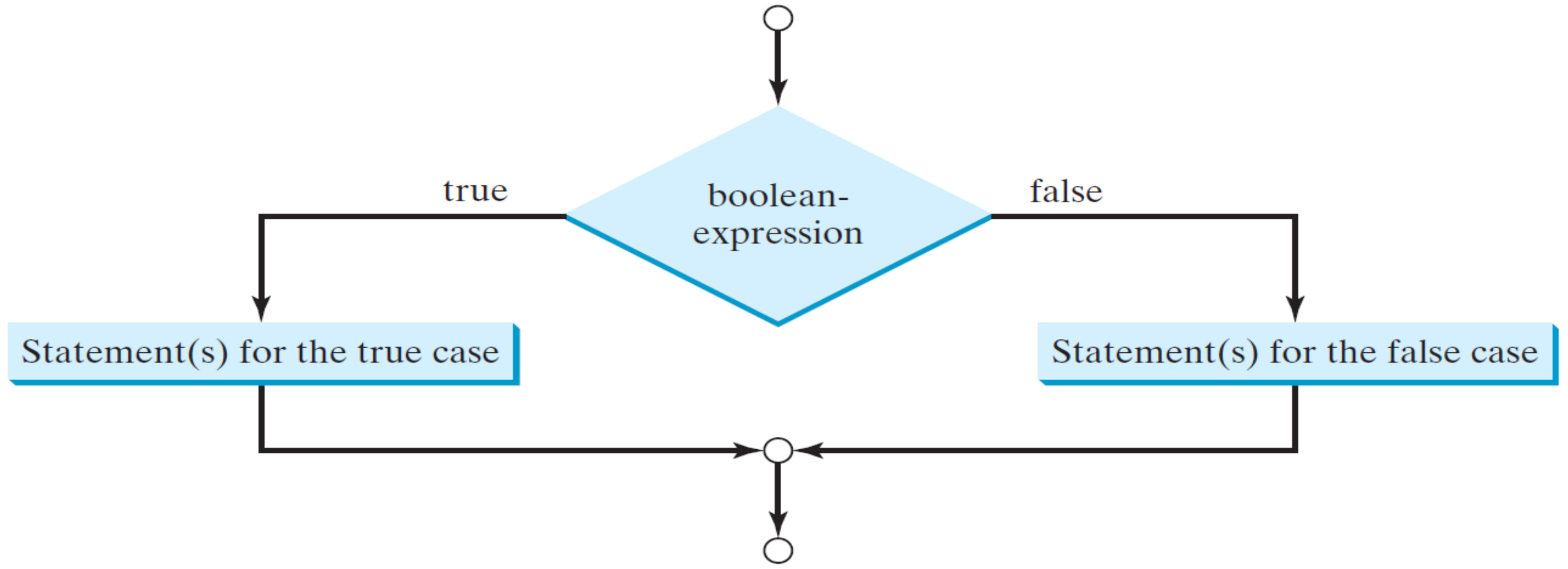
- Performs one statement or block if the *`expression`* is true, otherwise performs another statement or block.

if-else Statement Flowcharts



The Two-way if Statement

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


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

The `if/else` statement

- General Format:

```
if (expression)
   statement1;   // or block
else
   statement2;   // or block
```

`if/else`-What Happens

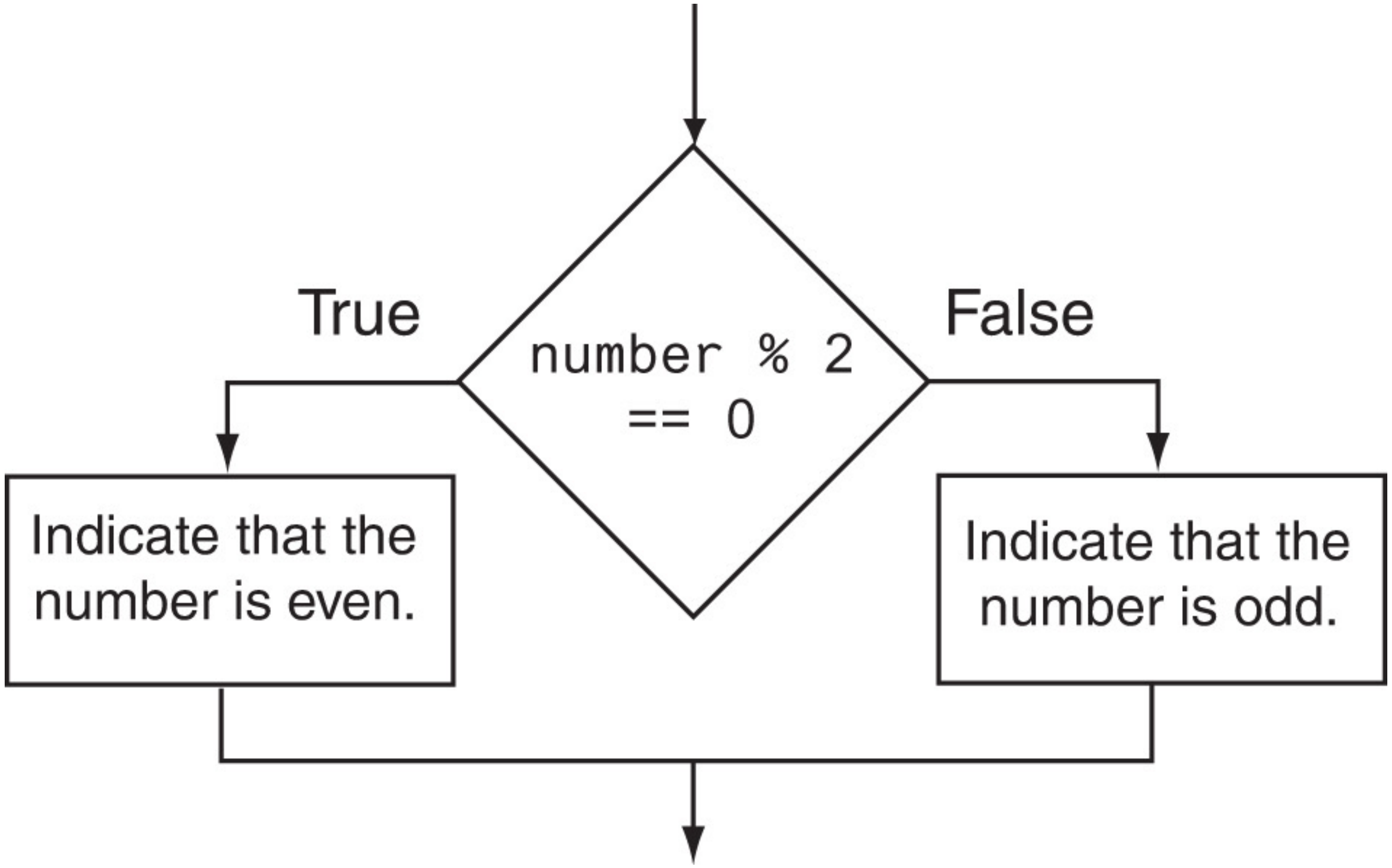
To evaluate:

```
if (expression)
    statement1;
else
    statement2;
```

- If the `expression` is `true`, then `statement1` is executed and `statement2` is skipped.

- If the `expression` is `false`, then `statement1` is skipped and `statement2` is executed.

Flowchart for Program 4-8 Lines 14 through 18


True
False
number % 2 == 0
Indicate that the number is even.
Indicate that the number is odd.

The if/else statement and Modulus Operator in Program 4-8

Program 4-8

```
// This program uses the modulus operator to determine
// if a number is odd or even. If the number is evenly divisible
// by 2, it is an even number. A remainder indicates it is odd.
#include <iostream>
using namespace std;

int main()
{
   int number;

   cout << "Enter an integer and I will tell you if it\n";
   cout << "is odd or even. ";
   cin >> number;
   if (number % 2 == 0)
      cout << number << " is even.\n";
   else
      cout << number << " is odd.\n";
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter an integer and I will tell you if it
is odd or even. 17 [Enter]
17 is odd.
```

Testing the Divisor in Program 4-9

Program 4-9

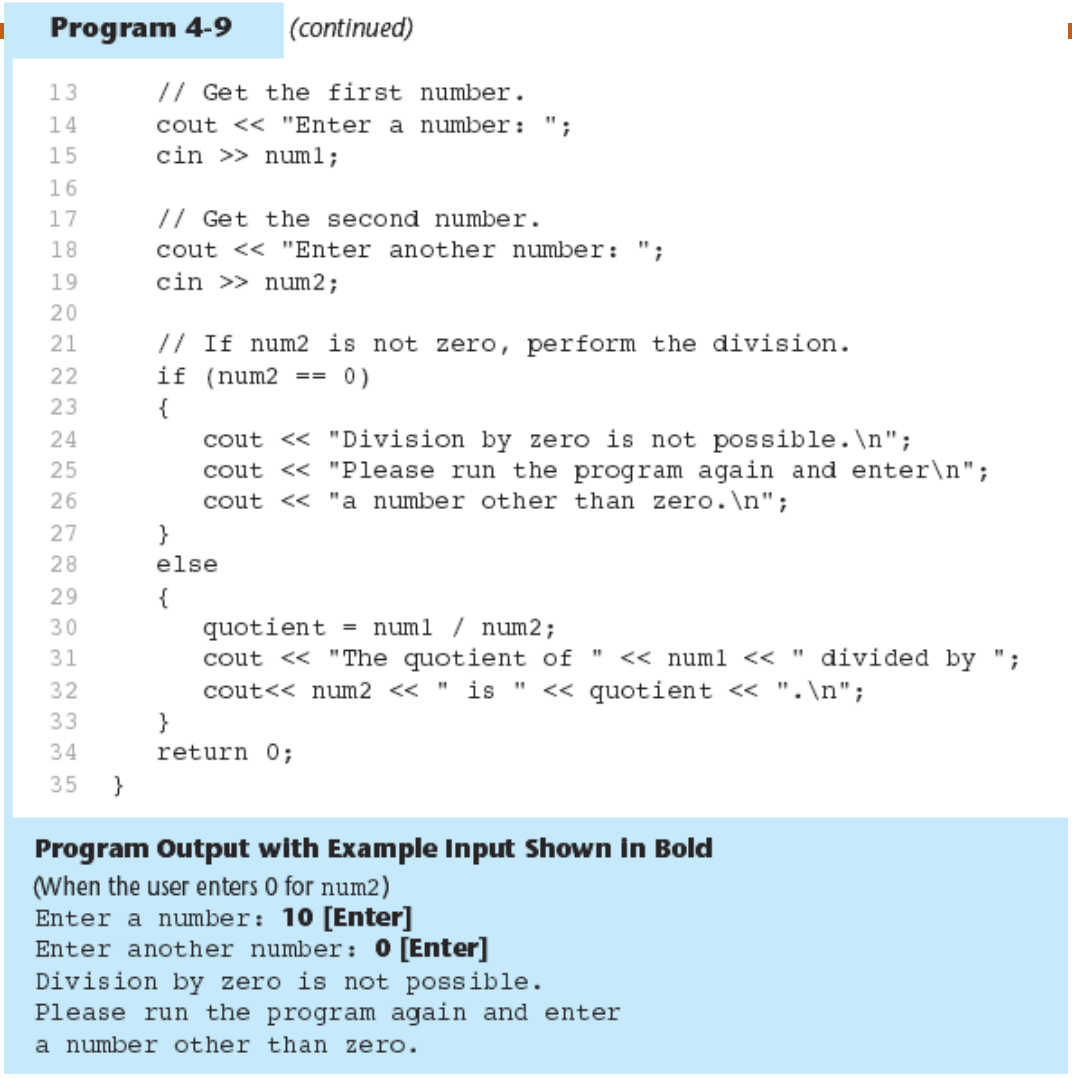
```
// This program asks the user for two numbers, num1 and num2.
// num1 is divided by num2 and the result is displayed.
// Before the division operation, however, num2 is tested
// for the value 0. If it contains 0, the division does not
// take place.
#include <iostream>
using namespace std;

int main()
{
   double num1, num2, quotient;

```

Continued…

Testing the Divisor in Program 4-9



Program 4-9 *(continued)*

```cpp
    // Get the first number.
    cout << "Enter a number: ";
    cin >> num1;

    // Get the second number.
    cout << "Enter another number: ";
    cin >> num2;

    // If num2 is not zero, perform the division.
    if (num2 == 0)
    {
       cout << "Division by zero is not possible.\n";
       cout << "Please run the program again and enter\n";
       cout << "a number other than zero.\n";
    }
    else
    {
       quotient = num1 / num2;
       cout << "The quotient of " << num1 << " divided by ";
       cout<< num2 << " is " << quotient << ".\n";
    }
    return 0;
}
```

Program Output with Example Input Shown in Bold

(When the user enters 0 for num2)

```
Enter a number: 10 [Enter]
Enter another number: 0 [Enter]
Division by zero is not possible.
Please run the program again and enter
a number other than zero.
```

if-else Example

```
if (radius >= 0) {
   area = radius * radius * 3.14159;

   cout <<  "The area for the circle of radius" << radius  << " is "
        << area;
}
else {
    cout << "Negative input";
}
```

CircleArea.cpp

Classroom Exercise

- Create a program for a first grader to practice subtractions. The program randomly generates two single-digit integers number1 and number2**.** Ask the user to input the difference of the two numbers with always the greater number as the first operand. If the user inputs wrong answer, print a message, “Your answer is wrong. The correct answer is ” with the correct answer. Otherwise print “Your answer is correct.”

- Hint:
  1. Check if number1 < number2, otherwise swap them
  2. Ask the user what is “number1 – number2”?
  3. Print the message according to the user input

Writing a Program

Step 1: Designing An Algorithm

1. Generate two random numbers.
2. Check if number1 < number2, otherwise swap them
3. Prompt the student to answer "what is number1 – number2?"
4. Grade the answer and display the result

- Subtraction.cpp

4.5

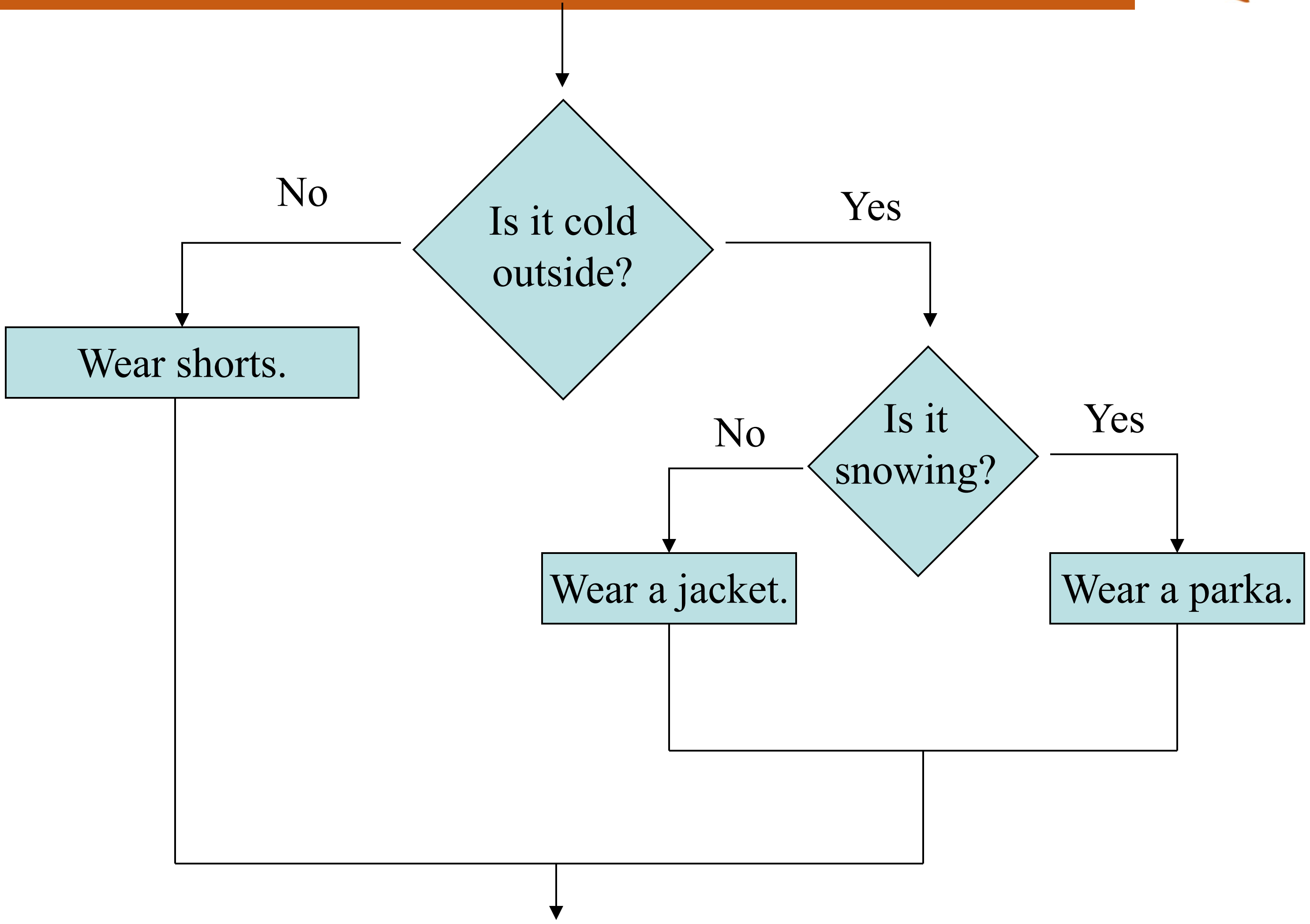
Nested `if` Statements

Nested `if` Statements

- An `if` statement that is nested inside another `if` statement

- Nested `if` statements can be used to test more than one condition

Nested if Statement Flowcharts



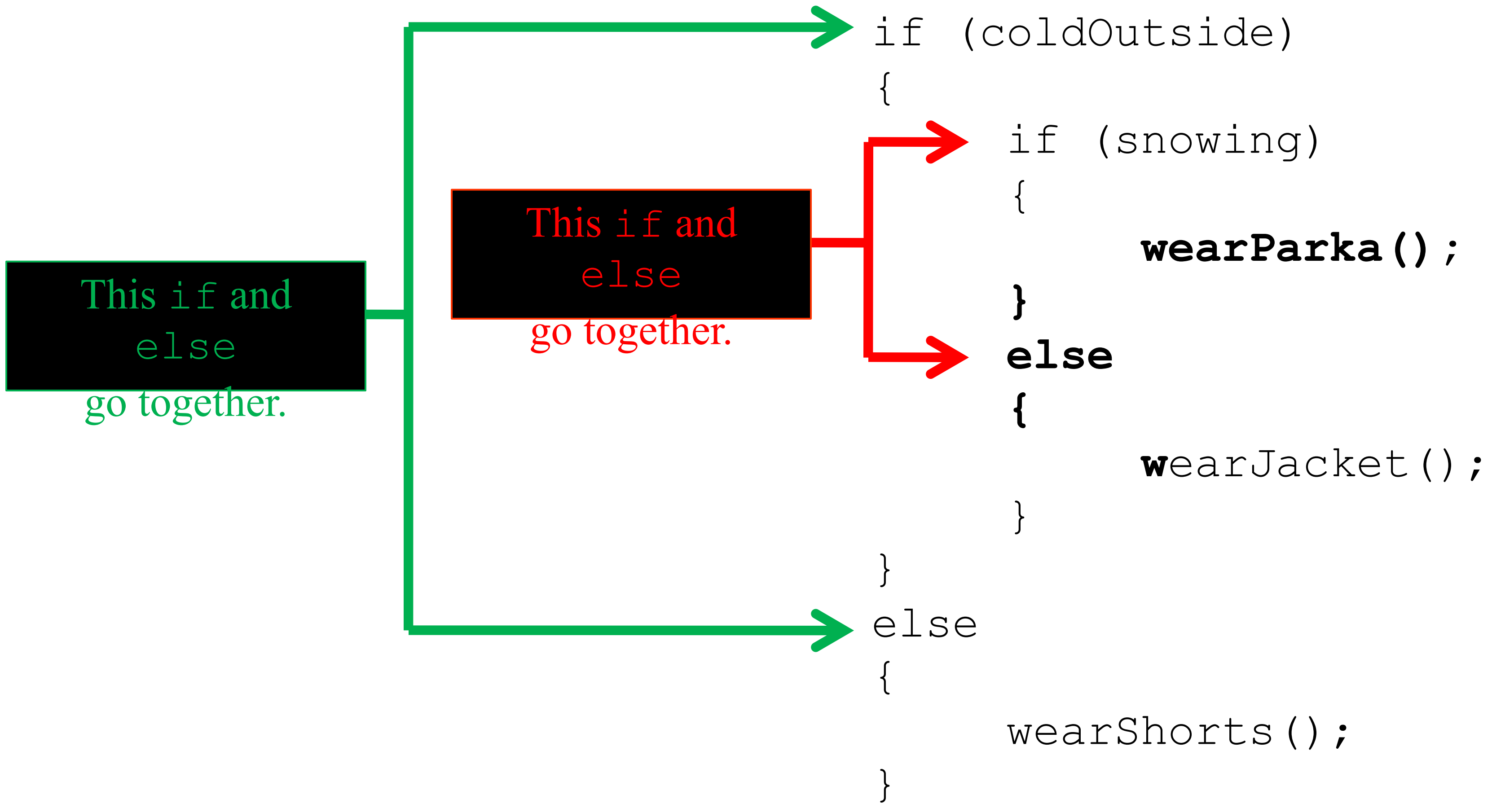
Nested if Statement Flowcharts

```
if (coldOutside)
{
     if (snowing)
     {
          wearParka();
     }
     else
     {
          wearJacket();
     }
}
else
{
     wearShorts();
}
```

Nested if Statement Flowcharts

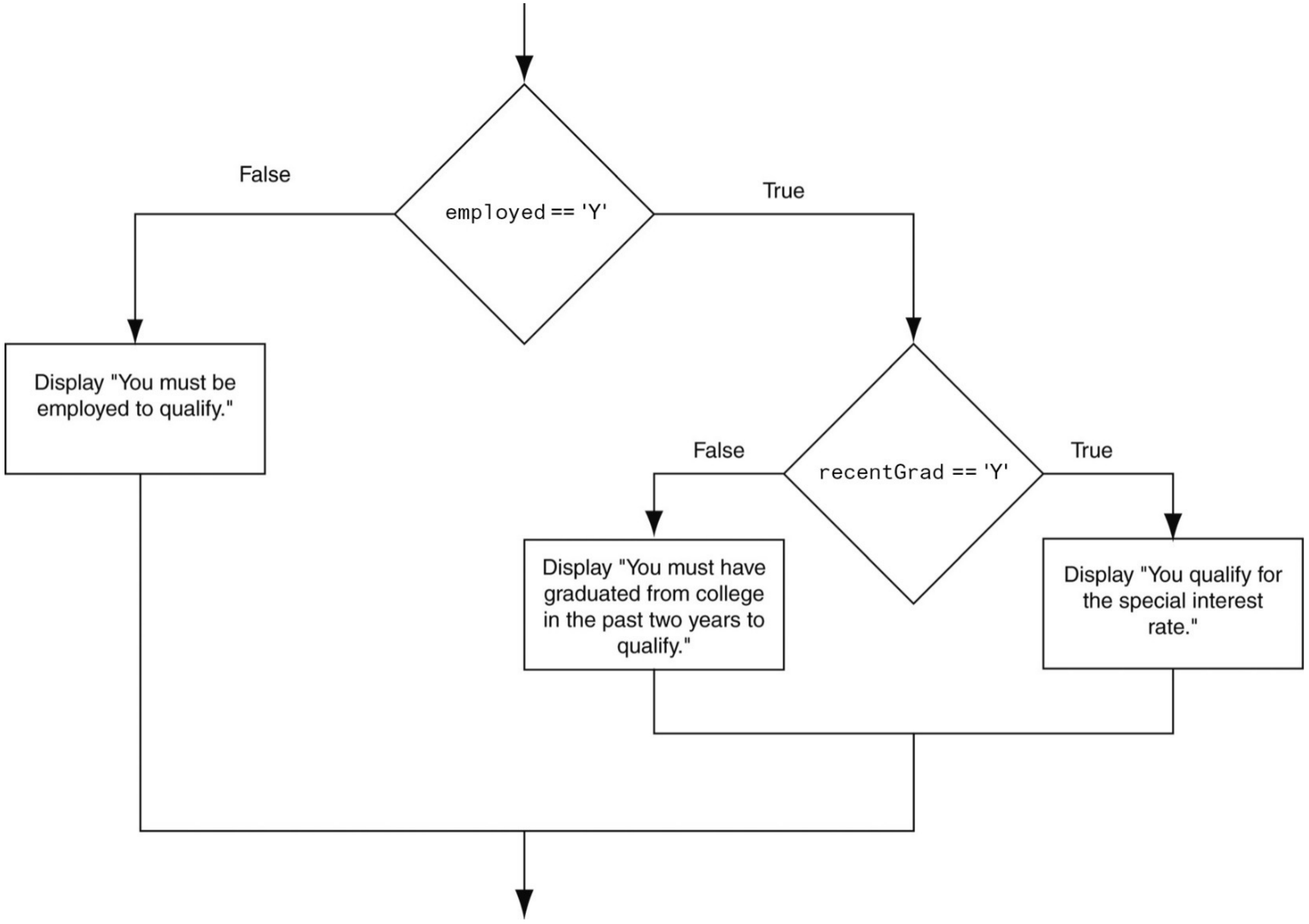
- Curly brace use is not required if there is only one statement to be conditionally executed.
- However, sometimes curly braces can help make the program more readable.
- Additionally, proper indentation makes it much easier to match up else statements with their corresponding if statement.

Alignment and Nested if Statements



```
if (coldOutside)
{
    if (snowing)
    {
        wearParka();
    }
    else
    {
        wearJacket();
    }
}
else
{
    wearShorts();
}
```

Another Flowchart for a Nested `if` Statement


employed == 'Y'
False
True
Display "You must be employed to qualify."
recentGrad == 'Y'
False
True
Display "You must have graduated from college in the past two years to qualify."
Display "You qualify for the special interest rate."

Nested `if` Statements

- From Program 4-10

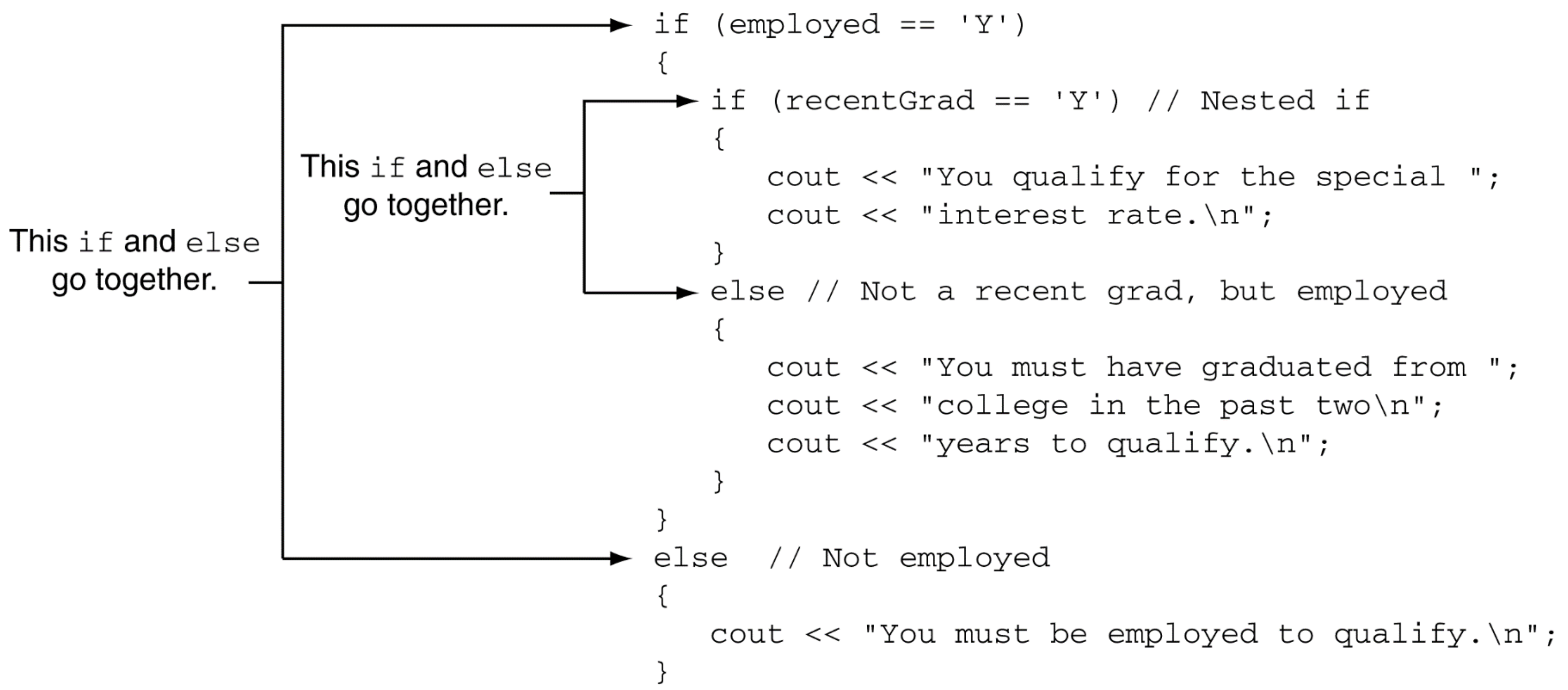
```
    // Determine the user's loan qualifications.
    if (employed == 'Y')
    {
        if (recentGrad == 'Y') //Nested if
        {
            cout << "You qualify for the special ";
            cout << "interest rate.\n";
        }
    }
```

Nested `if` Statements

- Another example, from Program 4-11

```
    // Determine the user's loan qualifications.
    if (employed == 'Y')
    {
       if (recentGrad == 'Y') // Nested if
       {
          cout << "You qualify for the special ";
          cout << "interest rate.\n";
       }
       else // Not a recent grad, but employed
       {
          cout << "You must have graduated from ";
          cout << "college in the past two\n";
          cout << "years to qualify.\n";
       }
    }
    else // Not employed
    {
       cout << "You must be employed to qualify.\n";
    }
```

Use Proper Indentation!



```
if (employed == 'Y')
{
   if (recentGrad == 'Y') // Nested if
   {
      cout << "You qualify for the special ";
      cout << "interest rate.\n";
   }
   else // Not a recent grad, but employed
   {
      cout << "You must have graduated from ";
      cout << "college in the past two\n";
      cout << "years to qualify.\n";
   }
}
else  // Not employed
{
   cout << "You must be employed to qualify.\n";
}
```

4.6

The `if/else if` Statement

The `if/else if` Statement

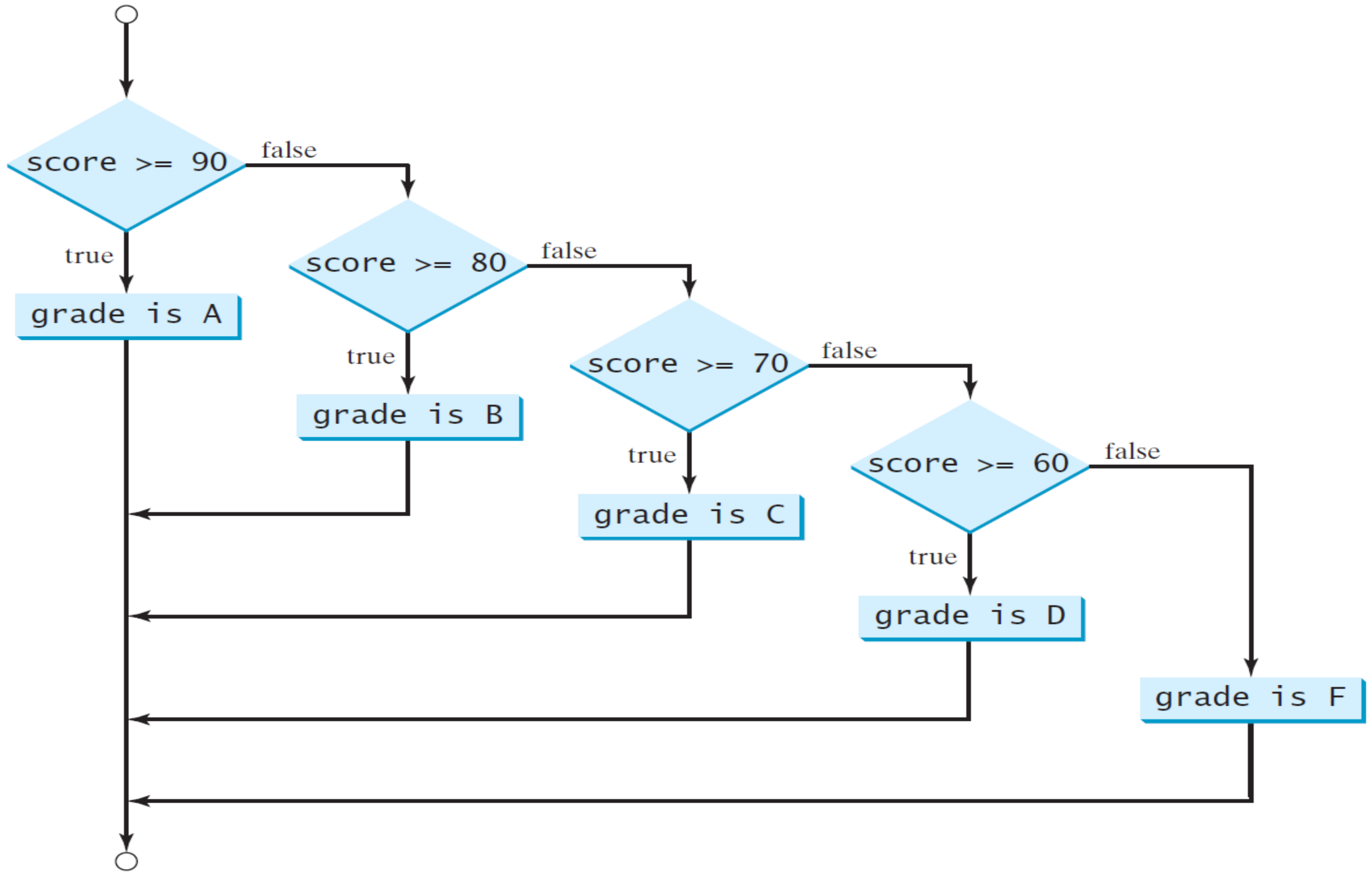
- Tests a series of conditions until one is found to be true
- Often simpler than using nested `if/else` statements
- Can be used to model thought processes such as:

  "If it is raining, take an umbrella,
  else, if it is windy, take a hat,
  else, take sunglasses”

if/else if Format

```
if (expression)
      statement1;   // or block
else if (expression)
      statement2;   // or block
      .
      . // other else ifs                 .
else if (expression)
      statementn;   // or block
```

Multi-Way if-else Statements



The `if/else if` Statement in Program 4-13

```
// Determine the letter grade.
if (testScore >= A_SCORE)
   cout << "Your grade is A.\n";
else if (testScore >= B_SCORE)
   cout << "Your grade is B.\n";
else if (testScore >= C_SCORE)
   cout << "Your grade is C.\n";
else if (testScore >= D_SCORE)
   cout << "Your grade is D.\n";
else
   cout << "Your grade is F.\n";
```

Using a Trailing `else` to Catch Errors in Program 4-14

- The trailing `else` clause is optional, but it is best used to catch errors.

```
    // Determine the letter grade.
    if (testScore >= A_SCORE)
        cout << "Your grade is A.\n";
    else if (testScore >= B_SCORE)
        cout << "Your grade is B.\n";
    else if (testScore >= C_SCORE)
        cout << "Your grade is C.\n";
    else if (testScore >= D_SCORE)
        cout << "Your grade is D.\n";
    else if (testScore >= 0)
        cout << "Your grade is F.\n";
    else
        cout << "Invalid test score.\n";
```

This trailing `else` catches invalid test scores

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";
```

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";
```

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";
```

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";
```

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";
```

Problem: Body Mass Index

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. The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
below 16	serious underweight
16-18	underweight
18-24	normal weight
24-29	overweight
29-35	seriously overweight
above 35	gravely overweight

BagelShop

- Do the bagel problem using if and else if

Day2

Pick a card

- Write a program that simulates picking a card from a deck of 52 cards. Your program should display the rank (Ace,2,3 --- J, Q,K) and suit(clubs, D, H and Spades

- Output:
  - The card you picked is Queen of Hearts

Pick a card - Psuedocode

- Use a random generator and get a number. Since 52 cards are there , the range will be 52

- **To determine the rank**
  - If (number % 13 == 0) it is ace
    - = 10 it is jack
    - = 11 it is Queen
    - = 12 it is king
    - The rest Number(1-9) will number plus 1
- **To determine the suit**
  - If the card picked divided by 13 is
    - 0 let it be Clubs
    - 1 – Spades
    - 2 – Diamond
    - 3 – Hearts

4.7

Flags

Flags

- Variable that signals a condition
- Usually implemented as a `bool` variable
- Can also be an integer
  - The value `0` is considered `false`
  - Any nonzero value is considered `true`
- As with other variables in functions, must be assigned an initial value before it is used
- validTriangle.cpp

4.8

Logical Operators

Logical Operators

- Used to create relational expressions from other relational expressions
- Operators, meaning, and explanation:

`&&`	AND	New relational expression is true if both expressions are true
`||`	OR	New relational expression is true if either expression is true
`!`	NOT	Reverses the value of an expression – true expression becomes false, and false becomes true

Logical Operators

Operator	Name	Description
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

The && Operator

- The logical AND operator (&&) takes two operands that must both be boolean expressions.
- The resulting combined expression is true if (and *only* if) both operands are true.
- See example: LogicalAnd.java w5

Expression 1	Expression 2	Expression1 && Expression2
true	false	false
false	true	false
false	false	false
true	true	true

Truth Table for Operator &&

p_1	p_2	p_1 && p_2	Example (assume age = 24, weight = 140)
false	false	false	(age <= 18) && (weight < 140) is false, because (age > 18) and (weight <= 140) are both false.
false	true	false	(age < 18) && (weight >=140) is false, because (age < 18) is false but (weight >) is true.
true	false	false	(age > 18) && (weight > 140) is false, because (weight > 140) is false.
true	true	true	(age > 18) && (weight >= 140) is true, because both (age > 18) and (weight >= 140) are true.

The logical && operator in Program 4-15

```
// Determine the user's loan qualifications.
if (employed == 'Y' && recentGrad == 'Y')
{
   cout << "You qualify for the special "
        << "interest rate.\n";
}
else
{
   cout << "You must be employed and have\n"
        << "graduated from college in the\n"
        << "past two years to qualify.\n";
}
```

The || Operator

- The logical OR operator (||) takes two operands that must both be boolean expressions.
- The resulting combined expression is false if (and *only* if) both operands are false.
- Example: LogicalOr.java

Expression 1	Expression 2	Expression1 || Expression2
true	false	true
false	true	true
false	false	false
true	true	true

Truth Table for Operator ||

p_1	p_2	p_1 || p_2	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) || (weight >140)
false	true	true	(age > 34) || (weight <= 140) is true, because (age > 34) is false, but (weight <= 140) is true.
true	false	true	(age > 14) || (weight >= 150) is false, because (age > 14) is true.
true	true	true	

The logical || Operator in Program 4-16

```
    // Determine the user's loan qualifications.
    if (income >= MIN_INCOME || years > MIN_YEARS)
       cout << "You qualify.\n";
    else
    {
       cout << "You must earn at least $"
            << MIN_INCOME << " or have been "
            << "employed more than " << MIN_YEARS
            << " years.\n";
    }
```

The ! Operator

- The ! operator performs a logical NOT operation.

- If an *expression* is true, !*expression* will be false.

```
if (!(temperature > 100))
 cout << “Below the maximum temperature.";
```

- If **temperature > 100** evaluates to false, then the output statement will be run.

Expression 1	!Expression1
true	false
false	true

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.

The logical ! Operator in Program 4-17

```
// Determine the user's loan qualifications.
if (!(income >= MIN_INCOME || years > MIN_YEARS))
{
   cout << "You must earn at least $"
        << MIN_INCOME << " or have been "
        << "employed more than " << MIN_YEARS
        << " years.\n";
}
else
   cout << "You qualify.\n";
```

Truth Table for Operator ^

p_1	p_2	p_1 ^ p_2	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) ^ (weight > 140) is false, because (age > 34) is false and (weight > 140) is false.
false	true	true	(age > 34) ^ (weight >= 140) is true, because (age > 34) is false but (weight >= 140) is true.
true	false	true	(age > 14) ^ (weight > 140) is true, because (age > 14) is true and (weight > 140) is false.
true	true	false	

Short Circuiting

- Logical AND and logical OR operations perform *short-circuit evaluation* of expressions.

- Logical AND will evaluate to false as soon as it sees that one of its operands is a false expression.

- Logical OR will evaluate to true as soon as it sees that one of its operands is a true expression.

- ShortCircuiting.cpp

Order of Precedence (1 of 2)

- The ! operator has a higher order of precedence than the && and || operators.
- The && and || operators have a lower precedence than relational operators like
  < and >.
- Parenthesis can be used to force the precedence to be changed.

Order of Precedence (2 of 2)

Order of Precedence	Operators	Description
1	(unary negation) `!`	Unary negation, logical NOT
2	`* / %`	Multiplication, Division, Modulus
3	`+ -`	Addition, Subtraction
4	`< > <= >=`	Less-than, Greater-than, Less-than or equal to, Greater-than or equal to
5	`== !=`	Is equal to, Is not equal to
6	`&&`	Logical AND
7	`||`	Logical OR
8	`= += -=` `*= /= %=`	Assignment and combined assignment operators.

Logical Operators-Examples

```
int x = 12, y = 5, z = -4;
```

`(x > y) && (y > z)`	`true`
`(x > y) && (z > y)`	`false`
`(x <= z) || (y == z)`	`false`
`(x <= z) || (y != z)`	`true`
`!(x >= z)`	`false`

Divisible by 5, 6

- Here is a program that checks
  - whether a number is divisible by 5 and 6,
  - whether a number is divisible by 5 or 6,
  - whether a number is divisible by 5 or 6 but not both
  - and whether a number is not divisible by both 5 and 6 w5

Example

```
Enter an integer: 24
24 is divisible by 5 or 6.
24 is divisible by 5 or 6, but not both.
24 not divisible by both 5 and 6.
```

```
Enter an integer: 30
30 is divisible by 5 and 6.
30 is divisible by 5 or 6.
```

```
Enter an integer: 23
23 not divisible by both 5 and 6.
```

Leap Year - Class exercise

Write 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. w5

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

Problem: Lottery - Class exercise

Write a program that 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: P5

- ❖ 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.
- ❖ Else no award