COMPREHENSIVE EDITION

PROGRAMMING AND PROBLEM SOLVING WITH C++

SIXTH EDITION

Nell Dale and Chip Weems

Chapter 5
Conditions, Logical Expressions, and Selection Control Structures

Chapter 5 Topics

- Data Type `bool`
- Using Relational and Logical Operators to Construct and Evaluate Logical Expressions
- If-Then-Else Statements

Chapter 5 Topics

- **If-Then Statements**
- **Nested If Statements for Multi-way Branching**
- **Testing the State of an I/O Stream**
- **Testing a C++ Program**

Flow of Control

Flow of Control is the order in which program statements are executed

What are the possibilities?

Flow of Control

- **Sequential** unless a "control structure" is used to change the order

- Two general types of control structures

  Selection (also called branching)

  Repetition (also called looping)

bool Data Type

- Type `bool` is a built-in type consisting of just two values, the constants **true** and **false**

- We can declare variables of type **bool**

```
bool hasFever; // true if has high temperature

bool isSenior; // true if age is at least 55
```

C++ Control Structures

- **Selection**

  if

  if . . . else

  switch

- **Repetition**

  for loop

  while loop

  do . . . while loop

Expressions

Control structures use logical expressions to make choices, which may include:

6 Relational Operators

< <= > >= == !=

3 Logical Operators

! && ||

6 Relational Operators

are used in expressions of form:

ExpressionA	*Operator*	*ExpressionB*

temperature	>	humidity
rain	>=	average
B * B - 4.0 * A * C	<	0.0
hours	<=	40
abs (number)	==	35
initial	!=	‘Q’

Given

```
int  x, y;
x = 4;
y = 6;
```

Expression	Value
x < y	true
x + 2 < y	false
x != y	true
x + 3 >= y	true
y == x	false
y == x+2	true
y = x + 3	7 (true)

Comparing Strings

- Two objects of type string (or a string object and a C string) can be compared using the relational operators

- A character-by-character comparison is made using the ASCII character set values

- If all the characters are equal, then the 2 strings are equal. Otherwise, the string with the character with smaller ASCII value is the “lesser” string

```
string   myState;
string   yourState;

myState = "Texas";
yourState = "Maryland";
```

Expression	Value
myState == yourState	false
myState > yourState	true
myState == "Texas"	true
myState < "texas"	true

Operator	Meaning	Associativity
!	NOT	Right
*, / , %	Multiplication, Division, Modulus	Left
+ , -	Addition, Subtraction	Left
<	Less than	Left
<=	Less than or equal to	Left
>	Greater than	Left
>=	Greater than or equal to	Left
==	Is equal to	Left
!=	Is not equal to	Left
&&	AND	Left
||	OR	Left
=	Assignment	Right

Logical Expression	Meaning	Description
! p	NOT p	! p is false if p is true ! p is true if p is false
p && q	p AND q	p && q is true if both p and q are true. It is false otherwise.
p || q	p OR q	p || q is true if either p or q or both are true. It is false otherwise.

```
int   age;
bool  isSenior,  hasFever;
float temperature;

age = 20;
temperature = 102.0;
isSenior = (age >= 55); // isSenior is  false
hasFever = (temperature > 98.6);
// hasFever is  true
```

Expression	Value
isSenior && hasFever	false
isSenior || hasFever	true
! isSenior	true
! hasFever	false

What is the value?

```
int age, height;
age = 25;
height = 70;
```

Expression	Value
!(age < 10)	?
!(height > 60)	?

"Short-Circuit" Evaluation

- C++ uses short circuit evaluation of logical expressions
- This means logical expressions are evaluated left to right and evaluation stops as soon as the final truth value can be determined

Short-Circuit Example

```
int age, height;

age = 25;
height = 70;
```

Expression

(age > 50) && (height > 60)

false

Evaluation can stop now because result of && is only true when both sides are true; thus it is already determined the expression will be false

More Short-Circuiting

```
int   age, height;

age = 25;
height = 70;
```

Expression

(height > 60) || (age > 40)

true

Evaluation can stop now because result of || is true if either side is true; thus it is already determined that the expression will be true

What happens?

```
int age, weight;
age = 25;
weight = 145;
```

Expression

(weight < 180) && (age >= 20)

true

Must still be evaluated because truth value of entire expression is not yet known

(Why?)

What happens?

```
int   age, height;

age = 25;
height = 70;
```

___Expression_____

! (height > 60) || (age > 50)

true

false

Does this part need to be evaluated?

Write an expression for each

- taxRate is over 25% and income is less than $20,000
- temperature is less than or equal to 75° or humidity is less than 70%
- age is over 21 and age is less than 60
- age is 21 or 22

Some Answers

```
(taxRate > .25) && (income < 20000)

(temperature <= 75) || (humidity < .70)

(age > 21) && (age < 60)

(age == 21) || (age == 22)
```

Use Precedence Chart

```
int     number;
float   x;

        number != 0 && x < 1 / number
```

/	**has highest priority**
<	**next priority**
!=	**next priority**
&&	**next priority**

What happens if Number has value 0?

Run Time Error (Division by zero) occurs

Short-Circuit Benefits

- One Boolean expression can be placed first to "guard" a potentially unsafe operation in a second Boolean expression

- Time is saved in evaluation of complex expressions using operators || and &&

Our Example Revisited

```
int     number;
float   x;

(number  !=  0)  &&  (x  <  1 / number)
```

is evaluated first and has value false

Because operator is &&, the entire expression will have value false; because of short-circuiting, the right side is not evaluated in C++

Warning About Expression in C++

- "Boolean expression" means an expression whose value is true or false
- An expression is any valid combination of operators and operands

Warning About Expression in C++

- Each expression has a value, which can lead to unexpected results
- Construct your expressions carefully
  - use precedence chart to determine order
  - use parentheses for clarification (and safety)

What went wrong?

This is only supposed to display "HEALTHY AIR" if the air quality index is between 50 and 80.

But when you tested it, it displayed "HEALTHY AIR" when the index was 35.

```
int  AQIndex;
AQIndex = 35;

if (50 < AQIndex < 80)
        cout << "HEALTHY AIR";
```

Analysis of Situation

AQIndex = 35;

According to the precedence chart, the expression

(50 < AQIndex < 80) ***means***

(50 < AQIndex) < 80 ***because < is Left Associative***

(50 < AQIndex) is false ***(has value 0)***

(0 < 80) is true.

Corrected Version

```
int  AQIndex;
AQIndex = 35;

if ((50 < AQIndex) && (AQIndex < 80))

    cout << "HEALTHY AIR";
```

Comparing Real Values

Do not compare floating point values for equality, compare them for near-equality.

```
float myNumber;
float yourNumber;

cin >> myNumber;
cin >> yourNumber;

if (fabs (myNumber - yourNumber) <
0.00001)
    cout << "They are close enough!"
         << endl;
```

Flow of Control

Flow of control is the order in which program statements are executed

THE 3 POSSIBILITIES ARE:

- **Sequential**
- **Selection Control Structure**
- **Loop Control Structure**

Selection Statements

Selection statements are statements used to choose an action, depending on the current status of your program as it is running

Expressions

Control structure use logical expressions which may include

6 Relational Operators

< <= > >= == !=

3 Logical Operators

! && ||

What can go wrong here?

```
float   average;
float   total;
int     howMany;
  .
  .
  .
average = total / howMany;
```

Improved Version

```
float  average,
float  total;
int     howMany;

if (howMany >  0)
{
    average = total  / howMany;
    cout <<  average;
}
else
    cout << "No prices were entered";
```

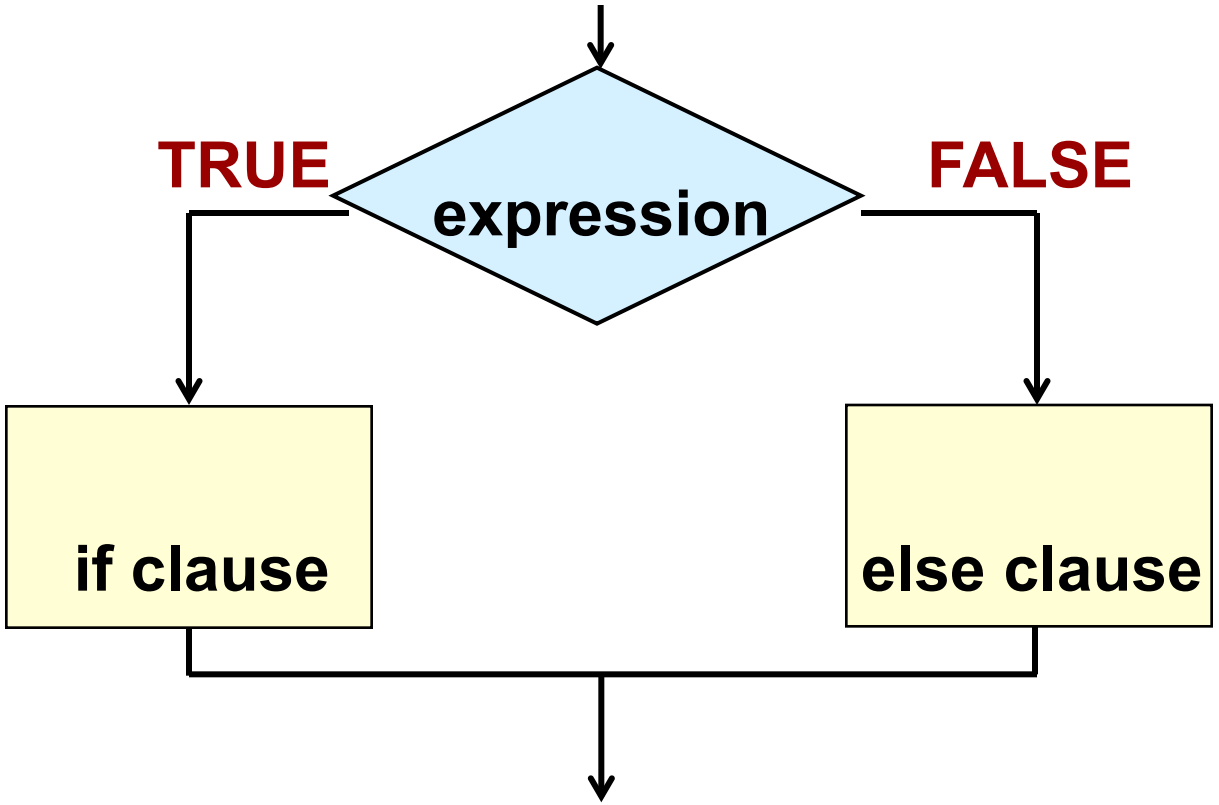
If-Then-Else Syntax

```
if (Expression)
        StatementA
else
        StatementB
```

NOTE: StatementA and StatementB each can be a single statement, a null statement, or a block

if .. else provides two-way selection

between executing one of 2 clauses (the if clause or the else clause)



Blocks Recommended

```
if (Expression)
{

}
else
{

}
```

"if clause"

"else clause"

```
int    carDoors, driverAge;
  float  premium,  monthlyPayment;
  .  .  .
  if  ((carDoors == 4)  &&  (driverAge > 24))
  {
     premium = 650.00;
     cout << " LOW RISK ";
  }
  else
  {
     premium = 1200.00;
     cout << " HIGH RISK ";
  }

  monthlyPayment = premium / 12.0 + 5.00;
```

What happens if you omit braces?

```
if  ((carDoors == 4)  &&  (driverAge >24))
      premium = 650.00;
      cout << " LOW RISK ";
else
      premium = 1200.00;
      cout << " HIGH RISK ";
monthlyPayment = premium / 12.0 + 5.00;
```

Compile error occurs: The "if clause" is the single statement following the if

Omitting Braces

Braces can be omitted only when a clause is a single statement

```
if (lastInitial  <= 'K')

    volume = 1;

else
    volume = 2;

cout   <<  "Look it up in volume # "
       <<  volume  <<  " of NYC phone book";
```

Example

```
// Where is first 'A' found in a string?
string   myString;
string::size_type   pos;
   .  .  .
pos  =  myString.find('A');

if  (pos == string::npos)
    cout  <<  "No 'A' was found" <<  endl;
else
    cout  <<  "An 'A' was found in position "
          <<  pos   <<  endl;
```

Example

Assign value .25 to discountRate and assign value 10.00 to shipCost if purchase is over 100.00

Otherwise, assign value .15 to discountRate and assign value 5.00 to shipCost

Either way, calculate totalBill

Example

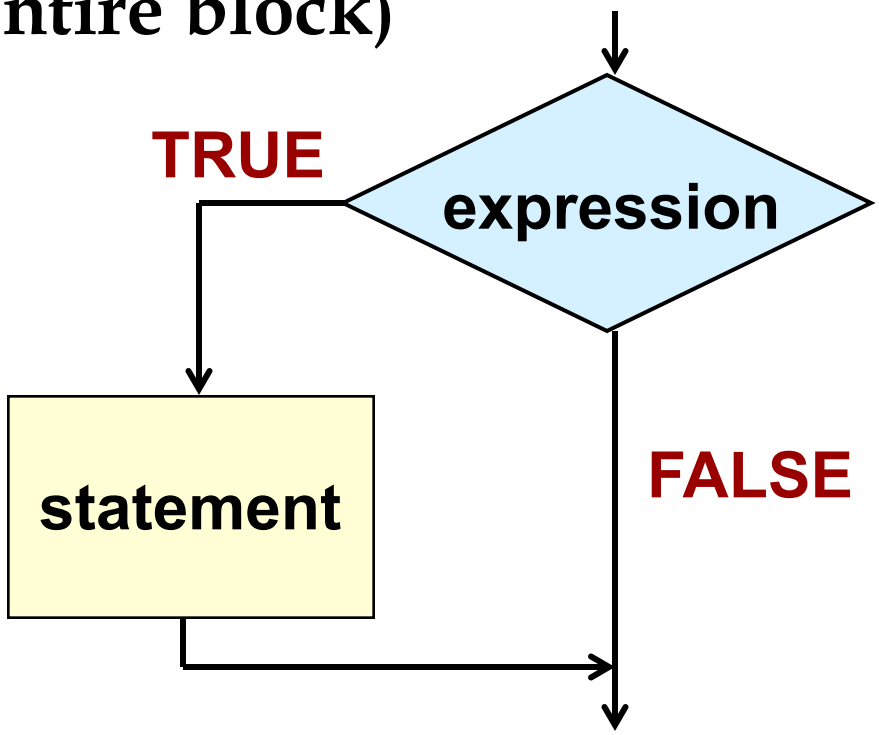
Braces cannot be omitted!

```
if (purchase  >  100.00)
{
    discountRate  =  .25;
    shipCost   =  10.00;
}
else
{
    discountRate  =  .15;
    shipCost   =  5.00;
}

totalBill = purchase * (1.0 - discountRate) +
shipCost;
```

If-Then Statement

Determine whether or not to execute a statement (which can be a single statement or an entire block)



If-Else Syntax

```
if (Expression)
        Statement
```

NOTE: Statement can be a single statement, a null statement, or a block

Example

```
// Stop processing if bad data
int  number;

cout  <<  "Enter a non-zero number ";
cin    >>   number;

if (number ==  0)
{
    cout  <<  "Bad input. Program terminated
  ";
    return 1;
}
// Otherwise continue processing
```

These are equivalent. *Why?*

```
if  (number == 0)
{
        .
        .
        .
}
```

```
if  (! number )
{
        .
        .
        .
}
```

Each expression is only true when number has value 0

Examples

If taxCode is ‘T’, increase price by adding taxRate times price to it

If code has value 1, read values for income and taxRate from myInfile, and calculate and display taxDue as their product

If A is strictly between 0 and 5, set B equal to 1/A, otherwise set B equal to A

Some Answers

```
if  (taxCode == 'T')
    price = price + taxRate * price;

if (code == 1)
{
    myInfile >> income >> taxRate;
    taxDue = income * taxRate;
    cout << taxDue;
}
```

Remaining Answer

```
if ((A > 0) && (A < 5))
    B = 1/A;
else
    B = A;
```

Example

What is output? Why?

```
int  age;

age = 20;

if  (age = 16)
{
  cout << "Did you get driver's
  license?";
}
```

Example

What is output? Why?

```
int  age;

age =  30;

if  (age <  18)
    cout << "Do you drive?";
    cout << "Too young to vote";
```

Example

What is output? Why?

```
int  code;

code =  0;

if  (! code)
    cout << "Yesterday";
else
    cout << "Tomorrow";
```

Example

What is output? Why?

```
int  number;

number =  0;

if  (number =  0)
    cout << "Zero value";
else
    cout << "Non-zero value";
```

Nested If Statements

```
if  (Expression1 )
            Statement1

else  if  (Expression2 )
            Statement2
                  .
                  .
                  .
else if  (ExpressionN )
            StatementN
else
            Statement N+1
```

Exactly 1 of these statements will be executed

Nested If Statements

Each Expression is evaluated in sequence, until some Expression is found that is true

Only the specific Statement following that particular true Expression is executed

Nested If Statements

- If no Expression is true, the Statement following the final else is executed
- Actually, the final else and final Statement are optional, and if omitted and no Expression is true, then no Statement is executed

An example . . .

Multi-way Branching

```
if  (creditsEarned >= 90 )
    cout  <<  "SENIOR STATUS ";

else if  (creditsEarned >= 60 )
    cout  <<  "JUNIOR STATUS ";

else if  (creditsEarned >= 30 )
    cout  <<  "SOPHOMORE STATUS ";

else
    cout  <<  "FRESHMAN STATUS ";
```

Example

Display one word to describe the int value of number as "Positive", "Negative", or "Zero"

Your city classifies a pollution index

- **less than 35 as "Pleasant",**
- **35 through 60 as "Unpleasant",**
- **above 60 as "Health Hazard"**

Display the correct description of the pollution index value

One Answer

```
if (number >  0)

    cout << “Positive”;

else if (number < 0)

    cout << “Negative”;

else

    cout << “Zero”;
```

Other Answer

```
if (index < 35)

    cout << “Pleasant”;

else if (index <= 60)

    cout << “Unpleasant”;

else

    cout << “Health Hazard”;
```

Example

Write a void function DisplayMessage that you can call from main to describe the pollution index value it receives as an argument

Your city describes a pollution index

- **less than 35 as "Pleasant",**
- **35 through 60 as "Unpleasant",**
- **above 60 as "Health Hazard."**

```
void DisplayMessage(int index)
{
    if (index < 35)

        cout << “Pleasant”;

    else if (index <= 60)

        cout << “Unpleasant”;

    else

        cout << “Health Hazard”;
}
```

A Driver Program

```
#include <iostream>

using namespace std;

void DisplayMessage (int);     // Declare
  function

int main (void)
{
    int pollutionIndex; // Declare variable
```

A Driver Program, cont...

```
cout << "Enter air pollution index";
    cin >> pollutionIndex;
    DisplayMessage(pollutionIndex);  // Call
    return 0;
}
```

Example

Every Monday thru Friday you go to class

When it is raining you take an umbrella

But on the weekend, what you do depends on the weather

If it is raining you read in bed

Otherwise, you have fun outdoors

Solution

```
// Program tells how to spend your day
#include < iostream >
using namespace std;
void main (void)
{
    int       day;
    char   raining;
    cout << "Enter day (use 1 for Sunday)";
    cin   >>  day;
    cout << "Is it raining? (Y/N)";
    cin   >>  raining;
    if  ((day == 1) || (day == 7))
```

Solution, cont...

```
{ // Sat or Sun
        if  (raining == 'Y')
          cout << "Read in bed";
        else
          cout << "Have fun outdoors";
    }
    else
    {
        cout << "Go to class ";
        if  (raining == 'Y')
            cout << "Take an umbrella";
    }
}
```

In the absence of braces,

- an `else` is always paired with the closest preceding `if` that doesn't already have an `else` paired with it

Example

```
float average;

average = 100.0;

if  (average >= 60.0)
    if (average < 70.0)
       cout << "Marginal PASS";
else
    cout << "FAIL";
```

100.0

average

FAIL is printed; WHY? **The compiler ignores indentation and pairs the else with the second if**

Use Braces to Correct Problem

```
float average;

average = 100.0;

if  (average >= 60.0)
{
    if (average < 70.0)
        cout << "Marginal PASS";
}
else
    cout << "FAIL";
```

100.0

average

Each I/O stream has a state (condition)

- An **input stream** enters fail state when you
  - try to read invalid input data
  - try to open a file which does not exist
  - try to read beyond the end of the file

- An **output stream** enters fail state when you
  - try to create a file with an invalid name
  - try to create a file on a write-protected disk
  - try to create a file on a full disk

Determining the Stream State

- The stream identifier can be used as if it were a Boolean variable that has value false when the last I/O operation on that stream failed and has value true when it did not fail
- After you use a file stream, you should check on its state

Checking the State

```
ofstream  myOutfile;

myOutfile.open ("myOut.dat");

if  (! myOutfile)
{
  cout  <<  "File opening  error.  "
       <<  "Program terminated."  <<  endl;
  return 1;
}
// Otherwise send output to myOutfile
```

Testing Selection Control Structures

- To test a program with branches, use enough data sets to ensure that every branch is executed at least once
- This strategy is called **minimum complete coverage**

Testing Often Combines Two Approaches

WHITE BOX TESTING	BLACK BOX TESTING
Code Coverage	**Data Coverage**
Allows us to see the program code while designing the tests, so that data values at the boundaries, and possibly middle values, can be tested.	Tries to test as many allowable data values as possible without regard to program code.

Testing

- Design and implement a test plan
- A test plan is a document that specifies the test cases to try, the reason for each, and the expected output
- Implement the test plan by verifying that the program outputs the predicted results

PHASE	RESULT	TESTING TECHNIQUE
Problem solving	Algorithm	Algorithm walk-through
Implementation	Coded program	Code walk-through, Trace
Compilation	Object program	Compiler messages
Execution	Output	Implement test plan

Body Mass Index Problem

Problem

Implement a measure called the Body Mass Index (BMI)

BMI computes a ratio of your weight and height, which has become a popular tool to determine an appropriate weight.

The formula for non-metric values is

$$\mathbf{BMI = weight * 703 / height^2}$$

What is the BMI?

BMI correlates with body fat, which can be used to determine if a weight is unhealthy for a certain height.

Do a search of the Internet for "body mass index" and you will find more than a million hits.

What is the BMI?, continued

In these references, the formula remains the same but the interpretation varies somewhat, depending on age and sex.

Here is a the most commonly used generic interpretation.

BMI	*Interpretation*
< 20	Underweight
20-25	Normal
26-30	Overweight
over 30	Obese

Algorithm

Get Data **Level 1**

Prompt for weight
Read weight
Prompt for height
Read height

Test Data

IF weight < 0 OR height < 0
  Set dataAreOK to false
ELSE
  Set dataAreOK to true

Calculate BMI

Set bodyMassIndex to weight * 703 / height ^{2}

Algorithm Continued

Print

```
Print "Your BMI is ", bodyMassIndex,   '.'
Print "Interpretation and instructions."
IF bodyMassIndex <20
   Print "Underweight: Have a milk shake."
ELSE IF bodyMassIndex < 26
   Print "Normal: Have a glass of milk."
ELSE IF bodyMassIndex < 30
   Print "Overweight: Have a glass of iced tea."
ELSE
   Print "Obese: See your doctor."
```

C++ Program

```
//
   ****************************************
   **
// BMI Program
// This program calculates the body mass index
   (BMI)
// given a weight in pounds and a height in inches
   and
// prints a health message based on the BMI.
//
   ****************************************
   ***
#include <iostream>
using namespace std;
```

C++ BMI Program, continued

```
int main()
{
   const int BMI_CONSTANT = 703;// Non-metric constant
   float weight;                // Weight in pounds
   float height;                // Height in inches
   float bodyMassIndex;         // Appropriate BMI
   bool dataAreOK;              // True if data OK
```

C++ BMI Program, cont...

```
// Calculate body mass index
  bodyMassIndex = weight * BMI_CONSTANT
          / (height * height);
  // Print message indicating status
  cout << "Your BMI is "
       << bodyMassIndex << ". " << endl;
  cout << "Interpretation and instructions."
       << endl;
```

C++ BMI Program, cont...

```
  if (bodyMassIndex < 20)
        cout << "Underweight: ...." <<
  endl;
    else if (bodyMassIndex <= 25)
        cout << "Normal: ...."  << endl;
    else if (bodyMassIndex <=  30)
        cout << "Overweight:...."  <<
  endl;
    else
        cout << "Obese: ...." << endl;
    return 0;
}
```

Testing the BMI Program

There is no testing in this program, but there should be!!

- ***Should you use white box or black box testing?***
- ***What test should be included?***
- ***Where should the tests(s) be inserted?***