COMPREHENSIVE EDITION

PROGRAMMING AND PROBLEM SOLVING WITH C++

SIXTH EDITION

Nell Dale and Chip Weems

Chapter 5
Conditions, Logical Expressions, and Selection Control Structures

Background image © Toncsi/ShutterStock, Inc.

www.jblearning.com

Chapter 5 Topics

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

Review

1. What are the basic data types?
   a. How do you declare a variable?
   b. How do you initialize a variable?
   c. How do you assign a variable?
2. What are examples of stl functions?
3. What are examples of stl objects?
   a. What functions can you call on cin?
   b. What functions can you call on cout?
   c. What functions can you call on string?

Review

4. What is a command prompt?
5. What command do I need to compile make an executable called "theBestProgram" from a file named "DoubleYourMoney.cpp"?
6. What command do I use to rename a file from "theWrongName.cp" to "theRightName.cpp"?

Chapter 5 Topics

- If-Then Statements
- Nested If Statements for Multi-way Branching

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

< <= > >= == !=

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

Given

```
string   myState;
string   yourState;

myState = “Texas”;
yourState = “Maryland”;
```

Expression	Value
myState == yourState	false
myState > yourState	true
myState == “Texas”	true
myState < “texas”	true

Flow of Control

- Statement execution is linear unless specified otherwise
- Some programming statements allow us to:
  - decide whether or not to execute a particular statement
  - execute a statement over and over, repetitively
- These decisions are based on *boolean expressions* (or *conditions*) that evaluate to true or false
- The order of statement execution is called the *flow of control*

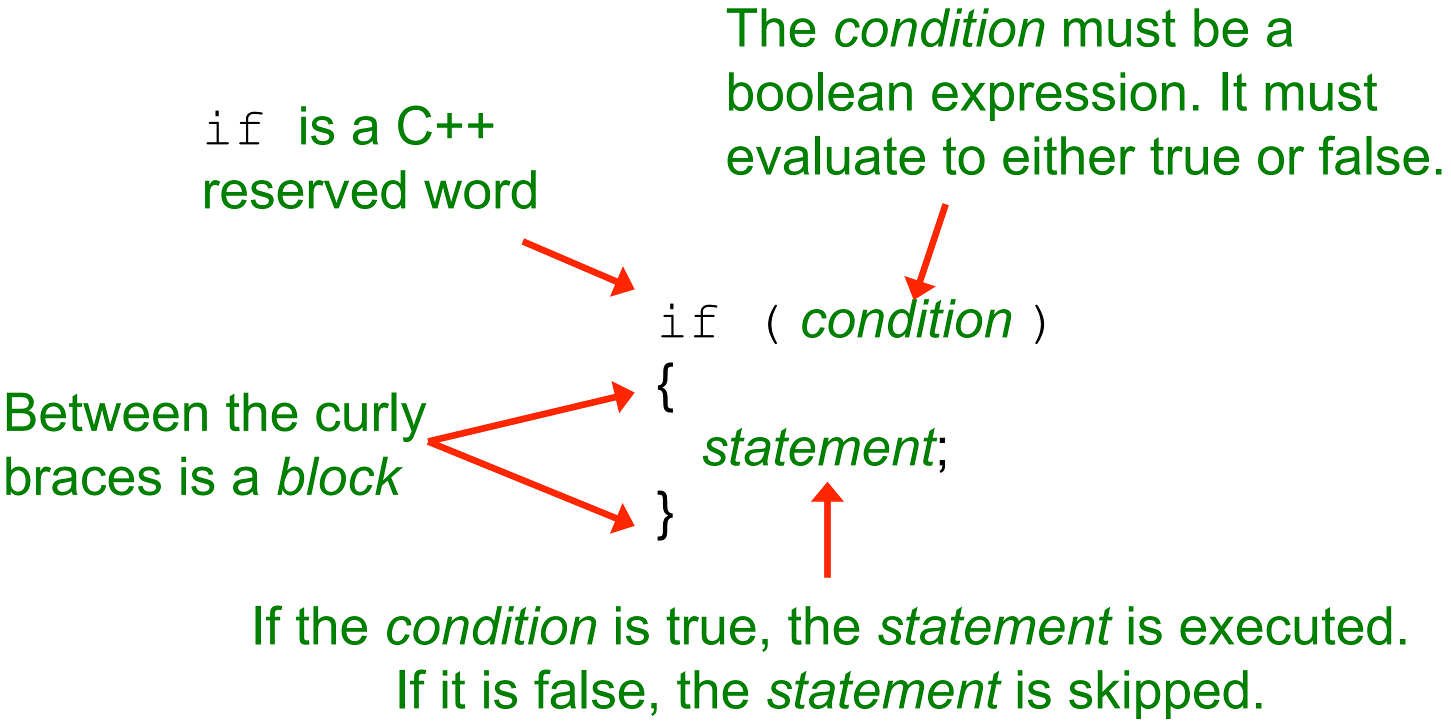
- Wake up early.
- Check for snow.
- Shovel sidewalks.
- Go back to sleep.
- Check for closures.
- Go to school.
- Wait 2 hours.
- Go to school.
- No school!
- Grade projects.
- Prepare for class.
- Etc.

Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next
- Therefore they are sometimes called *selection statements*
- Conditional statements give us the power to make basic decisions
- The C++ conditional statements are the
  - *if statement*
  - *if-else statement*
  - *if-else-if statement*
  - *switch statement*

The if Statement

- The syntax of a basic if statement is:



`if` is a C++ reserved word

The *condition* must be a boolean expression. It must evaluate to either true or false.

```
if ( condition )
{
   statement;
}
```

Between the curly braces is a *block*

If the *condition* is true, the *statement* is executed.
If it is false, the *statement* is skipped.

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

Conditions

- Examples of if statements:

- Equality:

```
if (x == y)
{
   // found a match
}
```

Inequality:

```
if (x != y)
{
   // found difference
}
```

Conditions

- Examples of if statements:

- Threshold:

```
if (x >= y)
{
   // x has reached
   // threshold y
}
```

```
if (x <= y)
{
   // x has not
   // reached
   // threshold y + 1
}
```

```
if (x > y – 1)
{
   // x has reached
   // threshold y
}
```

```
if (x < y + 1)
{
   // x has not
   // reached
   // threshold y + 1
}
```

Expressions

Control structure use logical expressions which may include

6 Relational Operators

`<    <=    >    >=    ==    !=`

The if Statement

- Consider the following if statement:

```
if (sum > MAX)
    delta = sum – MAX;
cout << "The sum is " <<  sum << endl;
```

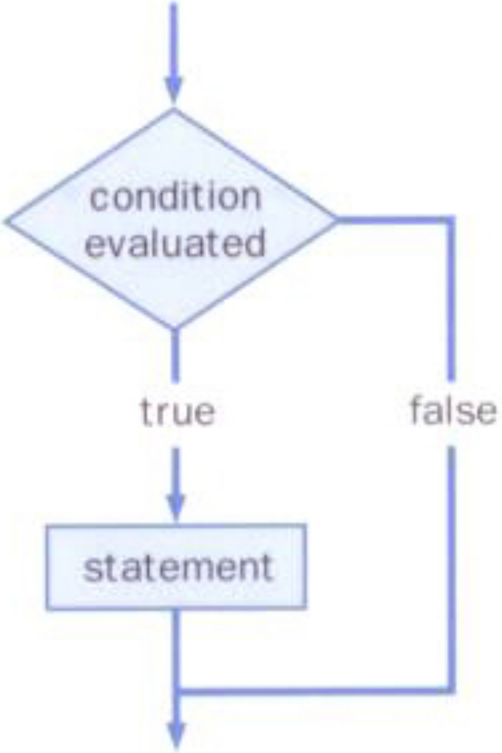
- First the condition is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not

- If the condition is true, the assignment statement is executed -- if it isn’t, it is skipped.

- Either way, the call to println is executed next

The if Statement

- The logic of an if statement:



Indentation

- The statement controlled by the `if` statement is indented to indicate that relationship
- The use of a consistent indentation style makes a program easier to read and understand
- Although it makes no difference to the compiler, proper indentation is crucial

"Always code as if the person who ends up maintaining your code will be a violent psychopath who knows where you live."

-- Martin Golding

The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

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

- If the *condition* is true, *statement1* is executed; if the condition is false, *statement2* is executed

- One or the other will be executed, but not both

Block Statements

- Several statements can be grouped together into a *block statement* delimited by braces

- A block statement can be used wherever a statement is called for in the Java syntax rules

```
if (total > MAX)
{
   cerr << "Error!!" << endl;
   errorCount++;
}
```

Indentation Revisited

- Remember that indentation is for the human reader, and is ignored by the computer

```
if (total > MAX)
   cerr << "Error!!" << endl;
   errorCount++;
```

- Despite what is implied by the indentation, the increment will occur whether the condition is true or not

The if-else Statement

- In an `if-else` statement, the `if` portion, or the `else` portion, or both, could be block statements

```
if (total > MAX)
{
   cerr << "Error!!" << endl;
   errorCount++;
}
else
{
   cout << "Total: "  <<  total << endl;
   current = total*2;
}
```

3.3 Multiple Alternatives

- What if you have more than two branches?
- Count the branches for the following earthquake effect example:
  - 8 (or greater)
  - 7 to 7.99
  - 6 to 6.99
  - 4.5 to 5.99
  - Less than 4.5

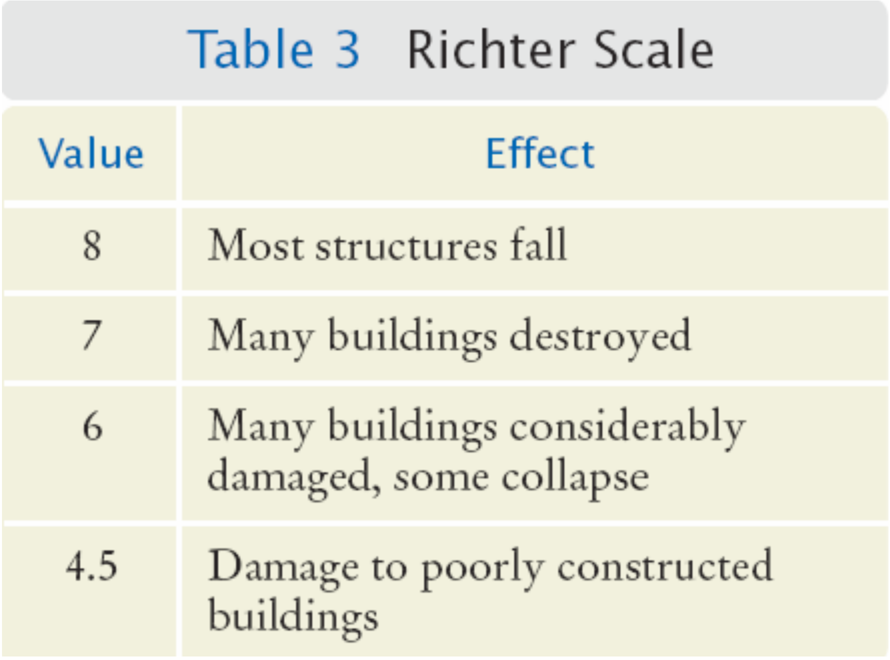
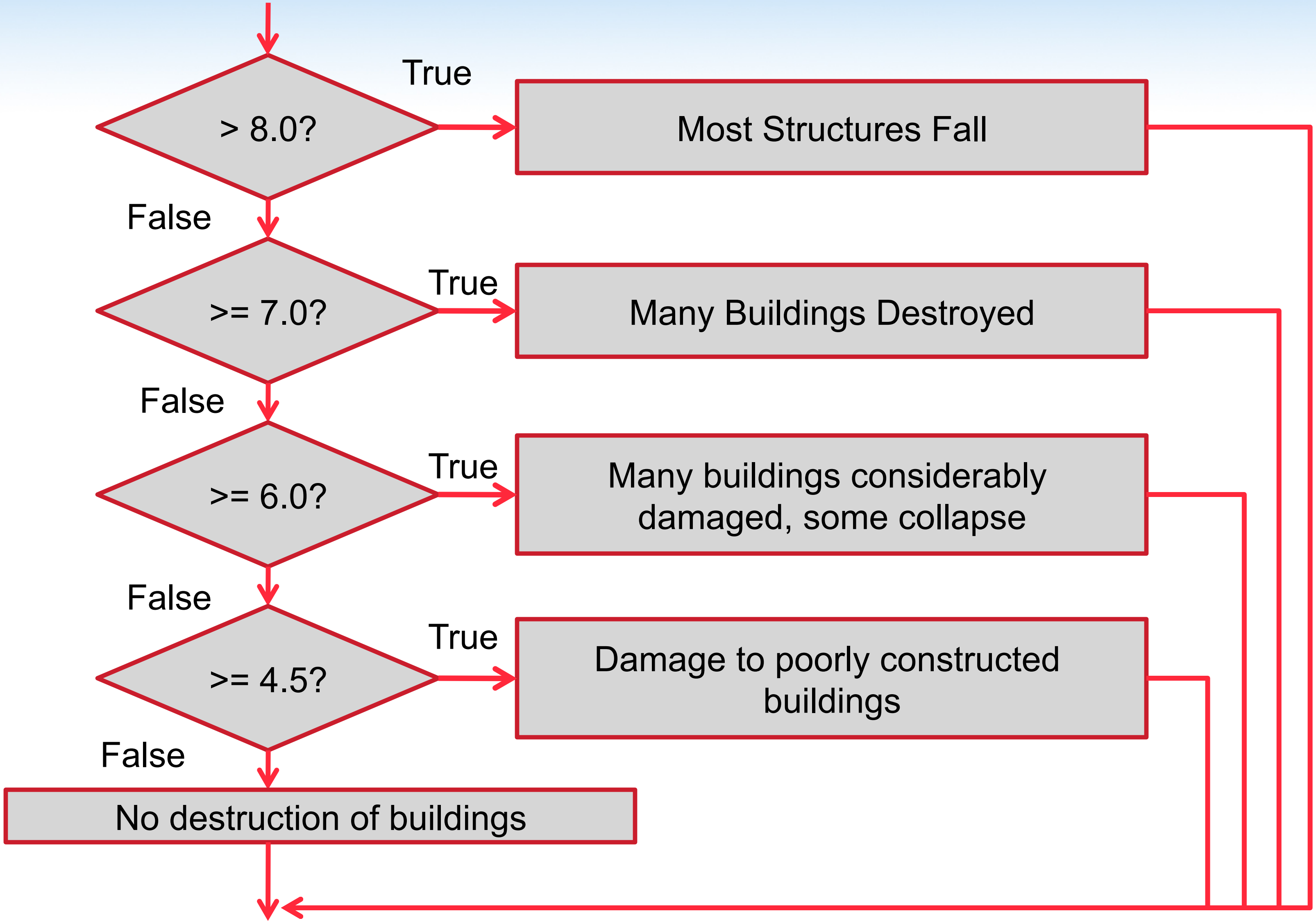


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

When using multiple `if` statements, test general conditions after more specific conditions.

Flowchart of Multiway branching


> 8.0?
True
Most Structures Fall
False
>= 7.0?
True
Many Buildings Destroyed
False
>= 6.0?
True
Many buildings considerably damaged, some collapse
False
>= 4.5?
True
Damage to poorly constructed buildings
False
No destruction of buildings

if, else if multiway branching

```
if (richter >= 8.0)   // Handle the ‘special case’ first
{
  cout << "Most structures fall"<< endl;
}
else if (richter >= 7.0)
{
  cout << "Many buildings destroyed"<< endl;
}
else if (richter >= 6.0)
{
  cout << "Many buildings damaged, some collapse"<< endl;
}
else if (richter >= 4.5)
{
  cout << "Damage to poorly constructed buildings"<< endl;
}
else    // so that the ‘general case’ can be handled last
{
  cout << "No destruction of buildings"<< endl;
}
```

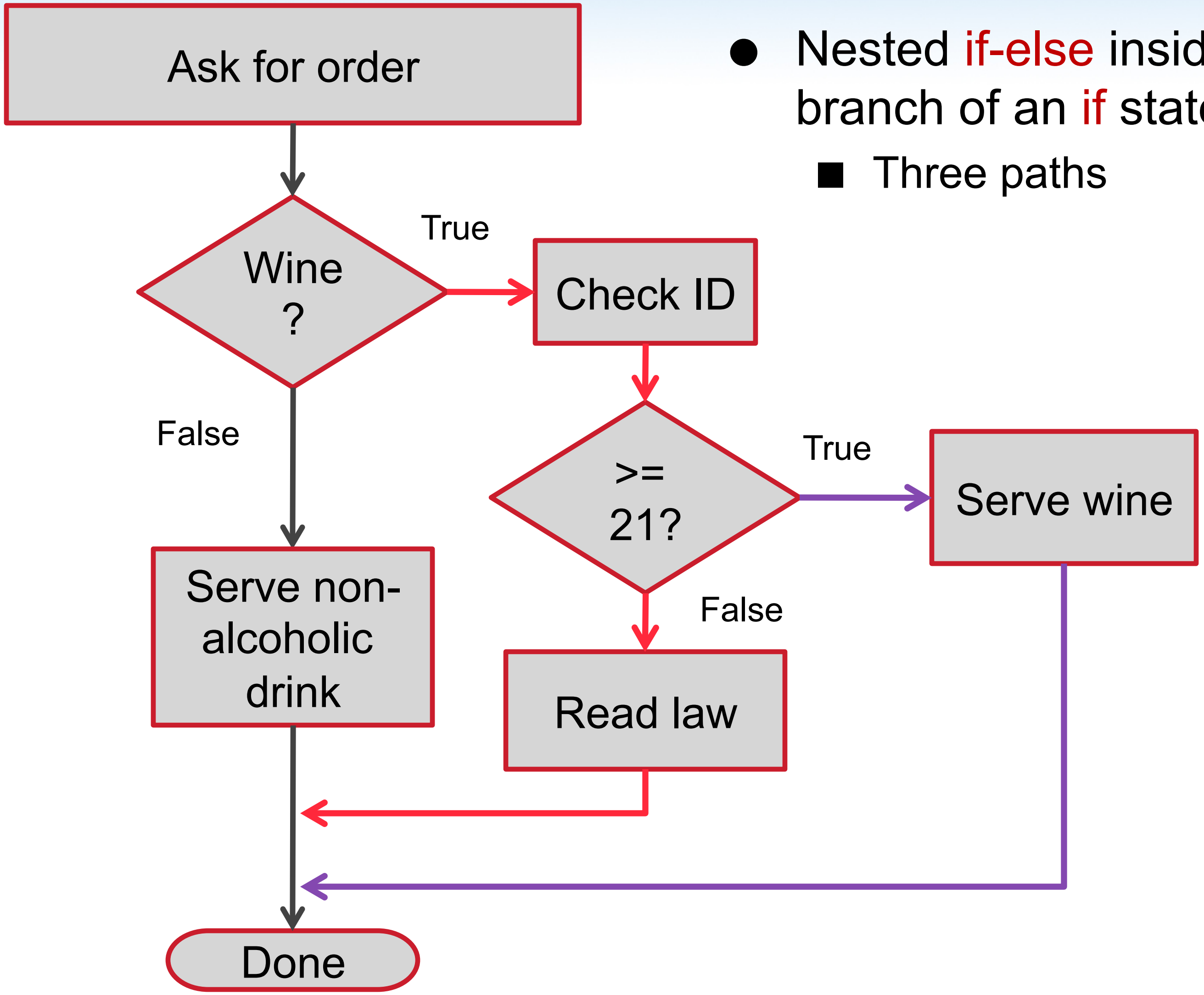
What is wrong with this code?

```
if (richter >= 8.0)
{
  cout << "Most structures fall"<< endl;
}
if (richter >= 7.0)
{
  cout << "Many buildings destroyed"<< endl;
}
if (richter >= 6.0)
{
  cout << "Many buildings damaged, some collapse"<< endl;
}
if (richter >= 4.5)
{
  cout << "Damage to poorly constructed buildings"<< endl;
}
```

3.4 Nested Branches

- You can *nest* an if inside either branch of an if statement.
- Simple example: Ordering drinks
  - Ask the customer for their drink order
  - `if` customer orders wine
    - Ask customer for ID
    - `if` customer’s age is 21 or over
      - Serve wine
    - Else
      - Politely explain the law to the customer
  - Else
    - Serve customers a non-alcoholic drink

Flowchart of a Nested if



- Nested if-else inside true branch of an if statement.
  - Three paths

Tax Example: Nested ifs

- Four outcomes (branches)
  - Single
    - <= 32000
    - > 32000
  - Married
    - <= 64000
    - > 64000

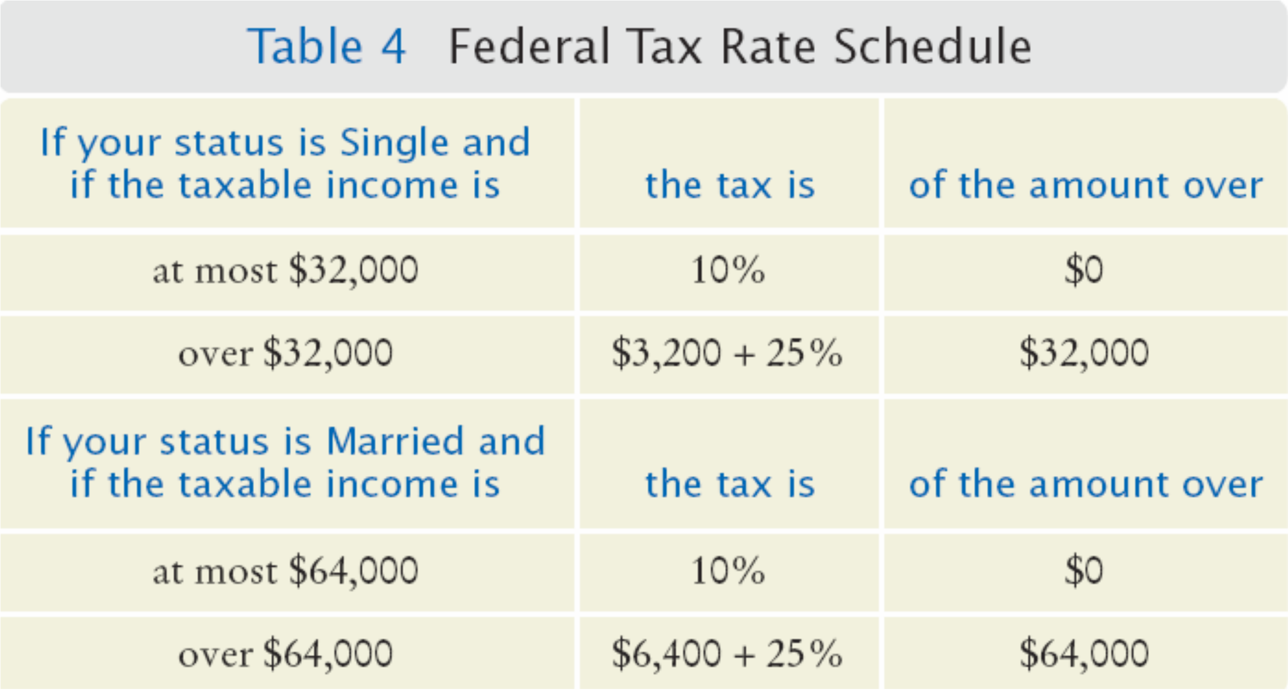
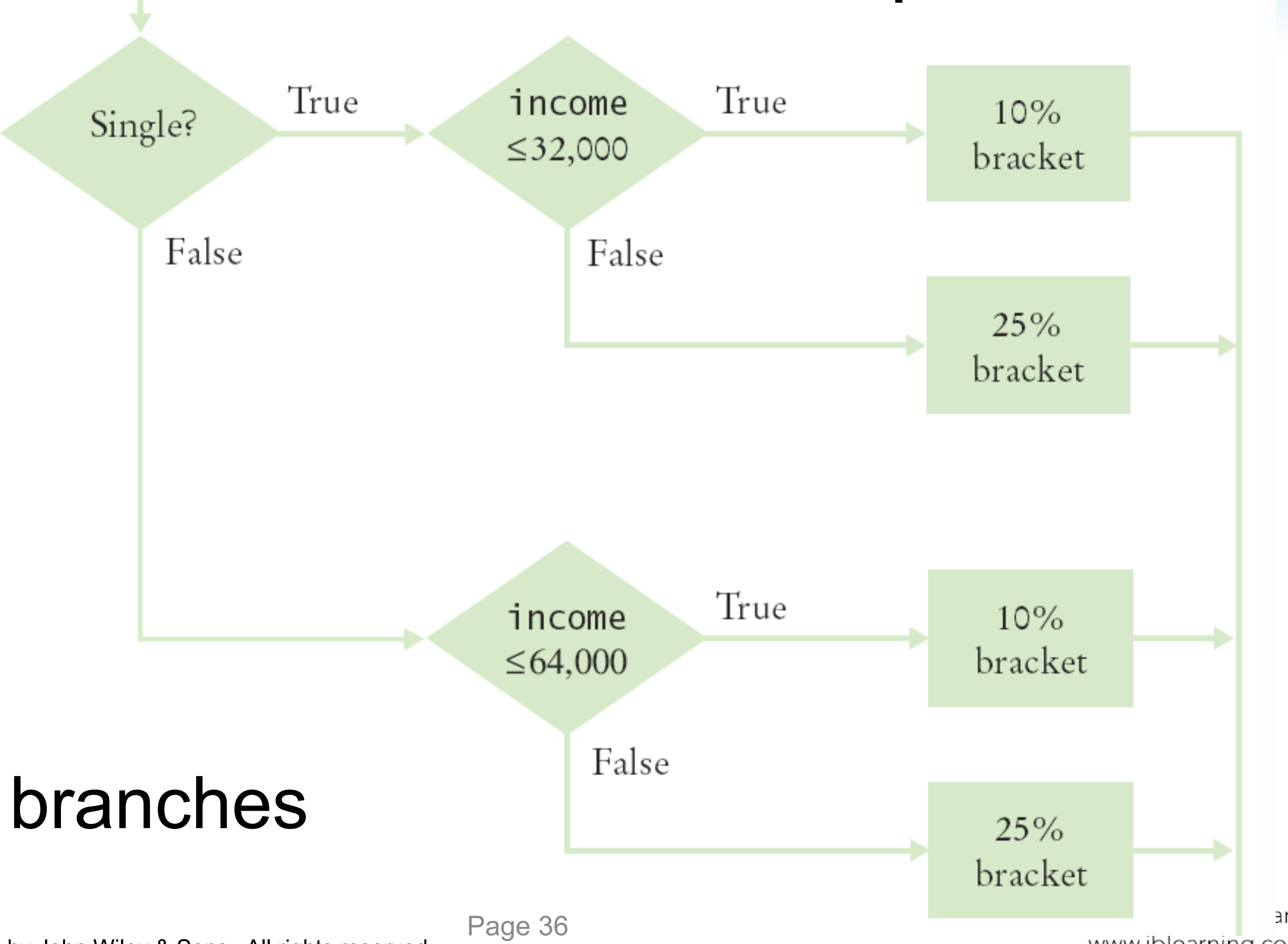


Table 4 Federal Tax Rate Schedule

If your status is Single and if the taxable income is	the tax is	of the amount over
at most $32,000	10%	$0
over $32,000	$3,200 + 25%	$32,000
If your status is Married and if the taxable income is	**the tax is**	**of the amount over**
at most $64,000	10%	$0
over $64,000	$6,400 + 25%	$64,000

Flowchart for Tax Example

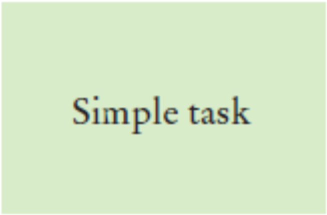


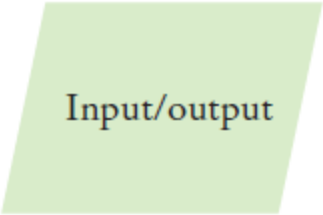
- Four branches

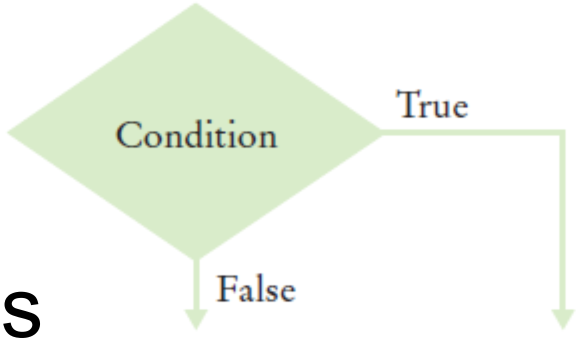
any
www.jblearning.com

Problem Solving: Flowcharts

- ❑ You have seen a few basic flowcharts
- ❑ A flowchart shows the structure of decisions and tasks to solve a problem
- ❑ Basic flowchart elements:







- ❑ Connect them with arrows
  - ▪ But never point an arrow inside another branch!

Each branch of a decision can contain tasks and further decisions.

Exercise

- Write an algorithm and a flowchart for a season calculator.
- The calculator should take a month and day as input
- If the month is valid, then the output should be the season, otherwise the output should be an error message.
- Limit your data types, classes and objects to
  - basic data types
  - string
  - cin
  - cout
- Work in pairs or groups of three.

Comparing Data

- When comparing data using boolean expressions, it's important to understand the nuances of certain data types
- Let's examine some key situations:
  - comparing floating point values for equality
  - comparing characters
  - comparing strings (alphabetical order)

Comparing Float Values

- You should rarely use the equality operator (`==`) when comparing two floating point values (`float` or `double`)
- Two floating point values are equal only if their underlying binary representations match exactly
- Computations often result in slight differences that may be irrelevant
- In many situations, you might consider two floating point numbers to be "close enough" even if they aren't exactly equal

Comparing Float Values

- To determine the equality of two floats, you may want to use the following technique:

```
if (fabs(f1 - f2) < TOLERANCE)
   cout << "Essentially equal";
```

- If the difference between the two floating point values is less than the tolerance, they are considered to be equal
- The tolerance could be set to any appropriate level, such as 0.000001

Comparing Strings

- We can use the relational operators to compare strings because the relational operators are **overloaded**
- The `string` class contains a function called `compare` to determine if one string comes before another
- A call to `name1.compare(name2)`
  - returns zero if `name1` and `name2` are equal (contain the same characters)
  - returns a negative value if `name1` is less than `name2`
  - returns a positive value if `name1` is greater than `name2`

Comparing Strings

```
if (name1.compare(name2) < 0)
   cout << name1 << " comes first.";
else
   if (name1.compare(name2) == 0)
      cout << "Same name";
   else
      cout << name2  << " comes first."<< endl;
```

- Because comparing characters and strings is based on a character set, it is called a *lexicographic ordering*

Lexicographic Ordering

- Lexicographic ordering is not strictly alphabetical when uppercase and lowercase characters are mixed
- For example, the string `"Great"` comes before the string `"fantastic"` because all of the uppercase letters come before all of the lowercase letters in Unicode
- Also, short strings come before longer strings with the same prefix (lexicographically)
- Therefore `"book"` comes before `"bookcase"`

== operator for string

- == is an operator function that takes two arguments as follows
- inline bool operator==(const string& lhs, const string& rhs){
  /* do actual comparison */
  return (lhs.compare(rhs) == 0);
  }

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”;
```

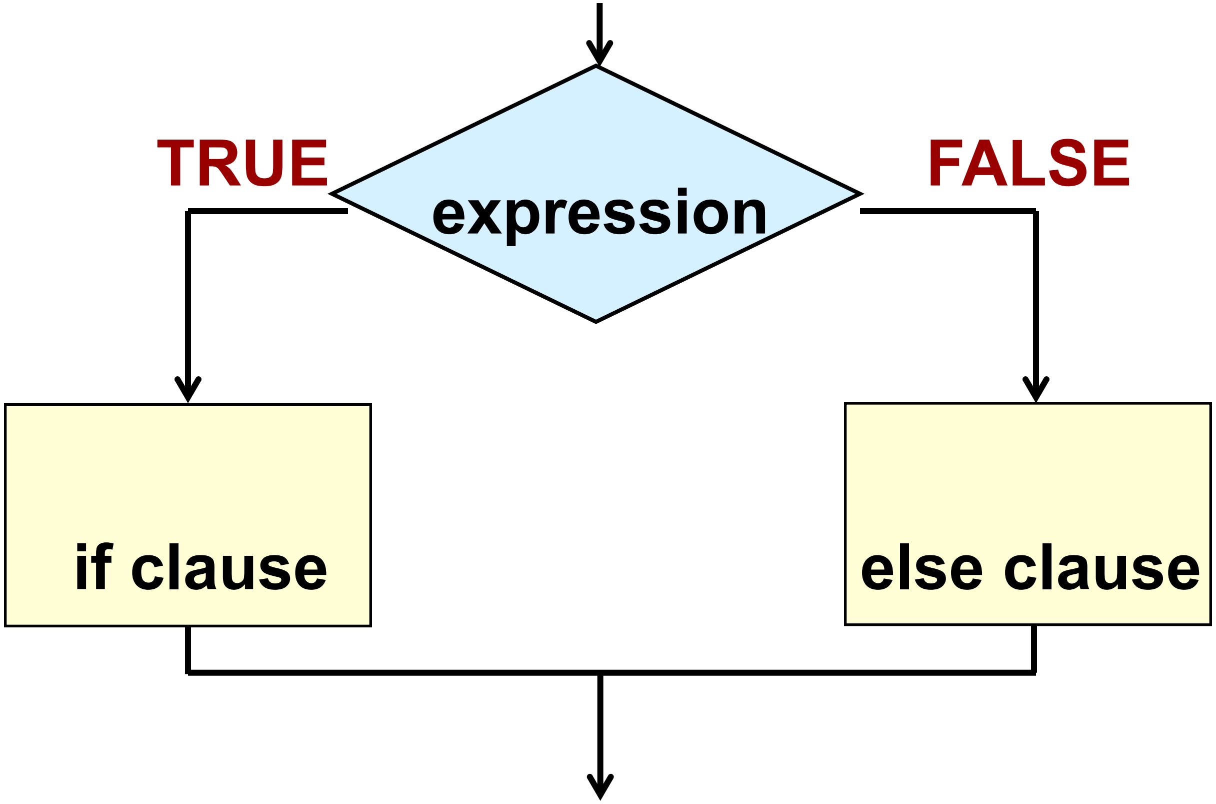
If-Then-Else Syntax

```
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)

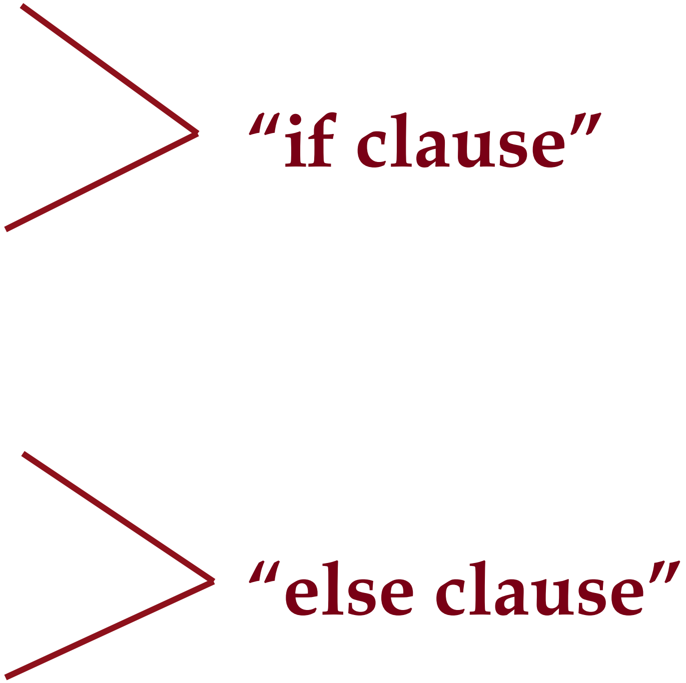


Blocks Recommended

```
if (Expression)
{

}
else
{

}
```



"if clause"

"else clause"

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

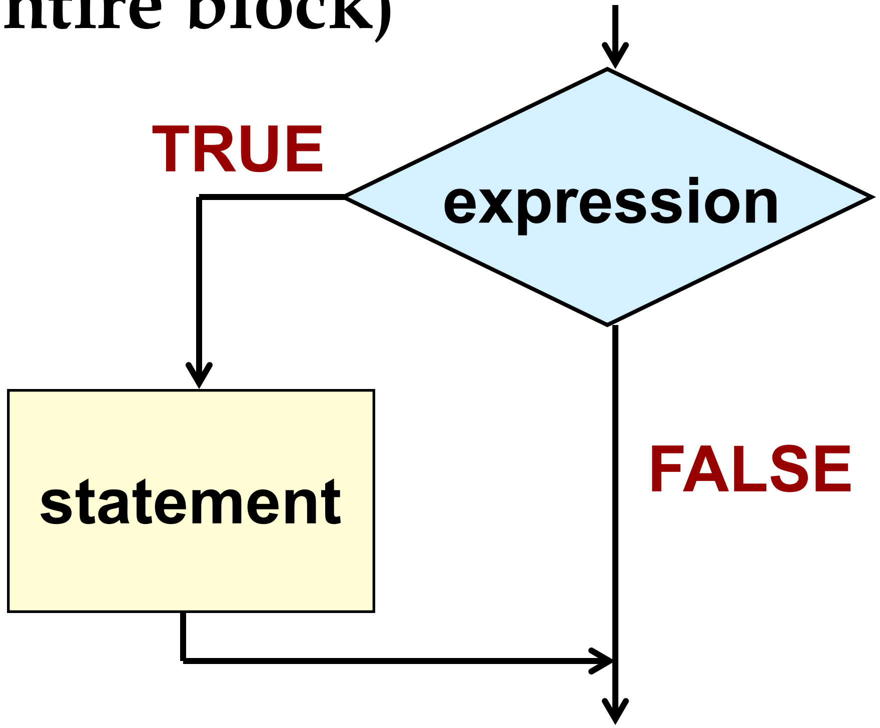
Braces cannot be omitted!

```
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”;
```

Exercise

- Consider a volunteer schedule program
  - Each day there are 2 times to volunteer (Morn., Eve.)
  - At least 7 people are needed for the task to be completed
  - Volunteers records have
    - a set of preferences for each day of the week
      - yes, no, maybe
    - past attendance
    - some future commitments of availability and unavailability
  - The volunteer captain needs a prioritized list of volunteers to contact for days where fewer than 7 people have committed.
  - write an algorithm and flow chart to create this list. Assume you have methods to get records in whatever format you need them (as long as you specify the format)