An Introduction to Programming with C++ Eighth Edition

Chapter 3: Variables and Constants

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Chapter Objectives

- Distinguish among a variable, a named constant, and a literal constant
- Explain how data is stored in memory
- Select an appropriate name, data type, and initial value for a memory location
- Declare a memory location in C++

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Beginning Step 4 in the Problem-Solving Process

- After Step 3, programmer has an algorithm and has desk-checked it
- The fourth step in the process is coding the algorithm into a program
- The step begins by assigning a descriptive name, data type, and (optionally) initial value to each unique input, processing, and output item in the IPO chart
- These are used to store the item in the computer's internal memory

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Internal Memory

- Computer's internal memory is composed of memory locations, each with a unique numeric address
- Similar to collection of storage bins
- Each address can store one item at a time
- Address can contain numbers, text, or program instructions
- To use a memory location, programmer must reserve the address, called *declaring*

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Internal Memory (cont'd.)

- Declaring a memory location is done with an instruction that assigns a name, data type, and (optional) initial value
- The name allows the programmer to refer to the memory location elsewhere in the program using a descriptive word, rather than the numeric address
- The data type indicates what type of information the address will store (e.g., number or text)

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Internal Memory (cont'd.)

- Two types of memory locations can be declared: variables and named constants
- Variables are memory locations whose values can change during runtime (when the program is running)
- Most memory locations are variables
- Named constants are memory locations whose values cannot change during program execution

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Internal Memory (cont'd.)

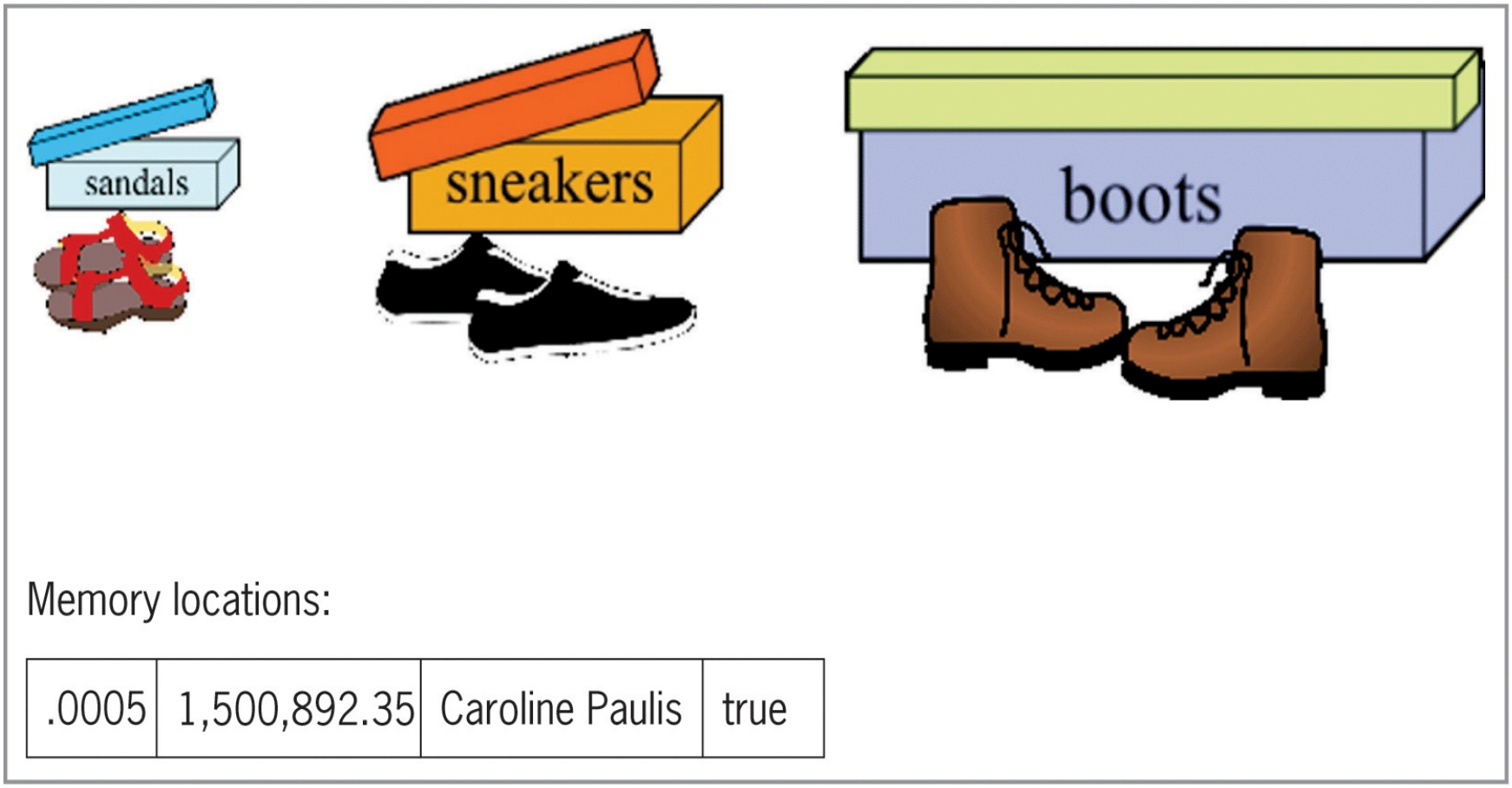


Figure 3-1 Illustration of show boxes and memory locations

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Name for a Memory Location

- Name (identifier) assigned to a memory location should be descriptive
- Should help the programmer/other programmers remember/understand the memory location's purpose
- Should be as short as possible while still being descriptive (especially if referenced often)
- Short names are easier to read and result in more concise code

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Name for a Memory Location (cont'd.)

- Rules for memory location names in C++
  - Name must begin with a letter and contain only letters, numbers, and the underscore character
  - No punctuation marks, spaces, or other special characters (such as $ or %) are allowed
  - Cannot be a keyword (word that has special meaning in C++)
  - Names are case sensitive
    - Example: `discount` is different from `DISCOUNT` and from `Discount`

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Name for a Memory Location (cont'd.)

- Most programmers use uppercase letters for named constants and lowercase for variables
  - Example: `PI` (constant), `radius` (variable)
- If constants contain more than one word, separate words with underscores
  - Example: `TAX_RATE`
- If variables contain more than one word, capitalize the first letter of each word after the first (called camel case)
  - Example: `adjustedGrossIncome`

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Name for a Memory Location (cont'd.)

HOW TO Name a Memory Location in C++

1. The name must begin with a letter.
2. The name can contain only letters, numbers, and the underscore character. No punctuation marks, spaces, or other special characters are allowed in the name.
3. The name cannot be a keyword. Appendix A contains a list of keywords in C++.
4. Names in C++ are case sensitive.

Valid names

```
grossPay, interest, TAX_RATE, PI
```

Invalid names	Reason
`2018Sales`	the name must begin with a letter
`end Balance`	the name cannot contain a space
`first.name`	the name cannot contain punctuation
`int`	the name cannot be a keyword
`RATE%`	the name cannot contain a special character

Figure 3-2 How to name a memory location in C++

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Revisiting the Addison O'Reilly Problem

Problem specification

Addison O'Reilly wants a program that calculates and displays the cost of a 4K Ultra HD TV, which is finally on sale at one of the stores in her area. The program should calculate the cost by multiplying the sale price by the state sales tax rate and then adding the result to the sale price.

Input	Processing	Output
sale price sales tax rate	Processing items: sales tax	cost

Algorithm:

1. enter the sale price and sales tax rate
2. calculate the sales tax by multiplying the sale price by the sales tax rate
3. calculate the cost by adding the sales tax to the sale price
4. display the cost

sale price	sales tax rate	sales tax	cost
~~2300~~	~~.0~~	~~11~~	~~241~~
200	.03	1	3

Figure 3-3 Problem specification, IPO chart, and desk-check table from Chapter 2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Revisiting the Addison O'Reilly Problem (cont'd.)

- IPO chart contains four input, processing, and output items
- Four memory locations are needed to store the values of the items
- Memory locations will be variables since their values will change during runtime

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Revisiting the Addison O'Reilly Problem (cont'd.)

IPO chart item	Variable name
sale price	salePrice
sales tax rate	taxRate
sales tax	salesTax
cost	cost

Figure 3-4 Names of the variables for the Addision O'Reilly problem

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Data Type for a Memory Location

- Memory locations come in different types and sizes
- Type and size you choose depends on the item you want to store
- A memory location will only accept an item that matches its data type
- Data type of a memory location is determined by the programmer when declaring the location

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Data Type for a Memory Location (cont'd.)

- Fundamental data types are basic data types built into C++
  - Also called primitive or built-in data types
  - Include `short`, `int`, `float`, `double`, `bool`, and `char`
- `bool` data type stores Boolean values (true and false)
- `short` and `int` types store integers (numbers without a decimal place)
  - Differences are range of values and memory used (`int` has the greater of both)

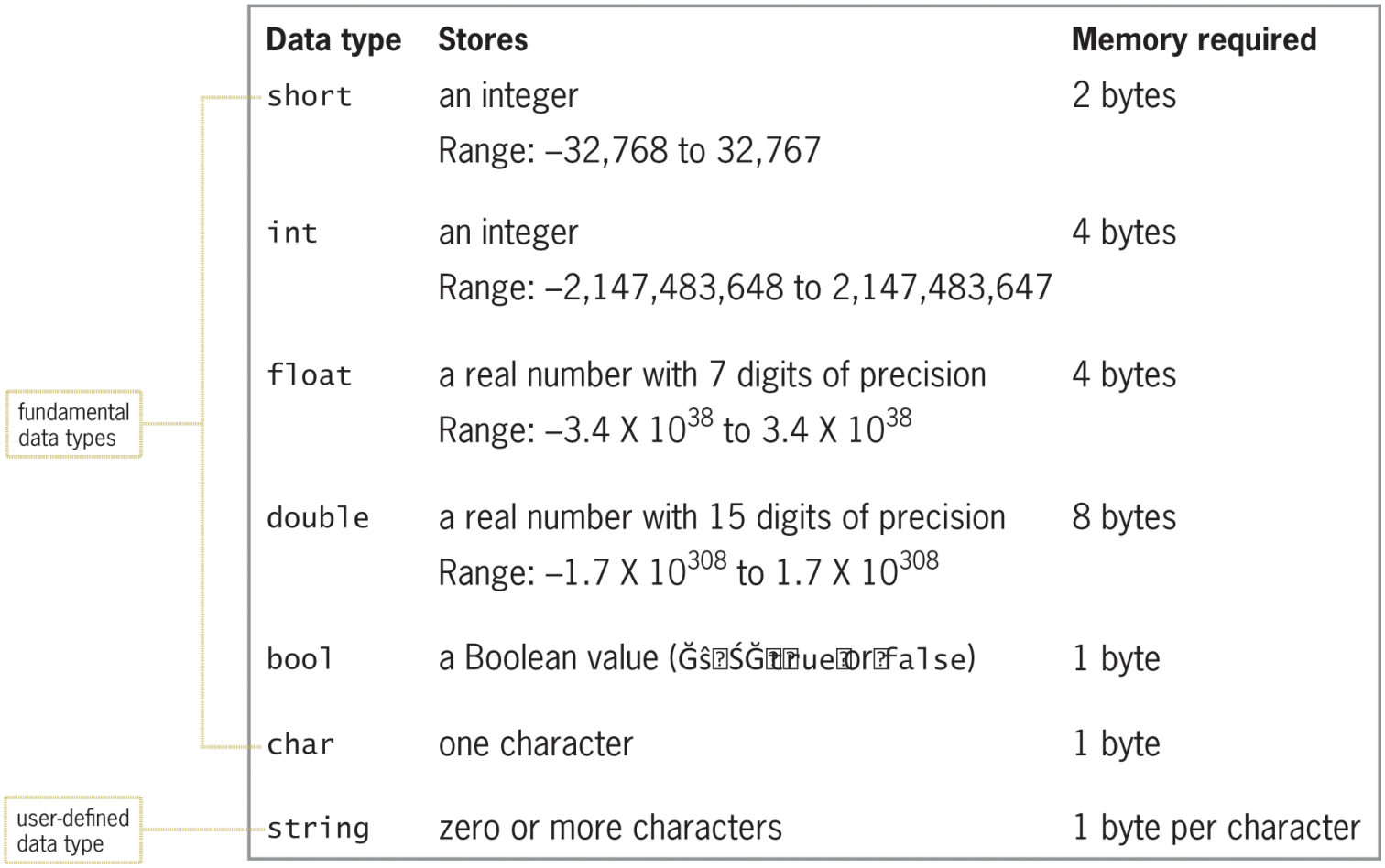
© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Data Type for a Memory Location (cont'd.)

- `float` and `double` types store real numbers (numbers with a decimal place)
  - Differences are range of values, precision, and memory used (`double` has the greater of each)
- `char` type stores characters (letter, symbol, or number that will not be used in a calculation)
  - Only one character stored at a time
- `string` data type is a user-defined data type (defined with a class, or group of instructions)
  - Can store zero or more characters

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Data Type for a Memory Location (cont'd.)



Data type	Stores	Memory required
`short`	an integer Range: –32,768 to 32,767	2 bytes
`int`	an integer Range: –2,147,483,648 to 2,147,483,647	4 bytes
`float`	a real number with 7 digits of precision Range: -3.4×10^{38} to 3.4×10^{38}	4 bytes
`double`	a real number with 15 digits of precision Range: -1.7×10^{308} to 1.7×10^{308}	8 bytes
`bool`	a Boolean value (Ğŝ□ŚĞ□□ue□r□false)	1 byte
`char`	one character	1 byte
`string`	zero or more characters	1 byte per character

Figure 3-5 Most commonly used data types in C++

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting a Data Type for a Memory Location (cont'd.)

IPO chart item	Variable name	Data type
sale price	salePrice	double
sales tax rate	taxRate	double
sales tax	salesTax	double
cost	cost	double

Figure 3-6 Data type assigned to each variable for the Addison O'Reilly problem

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

How Data Is Stored in Internal Memory

- Numbers represented in internal memory using binary (base 2) number system (two digits, 0 and 1)
- We are used to the decimal (base 10) number system (ten digits, 0 through 9)
- Character data is stored using ASCII codes
  - Eight-bit codes (bit = binary digit, 0 or 1)
  - Upper- and lowercase versions of letters have distinct codes
- Computer distinguishes between numbers and ASCII codes based on data type

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

How Data Is Stored in Internal Memory (cont'd.)

HOW TO Use the Decimal (Base 10) Number System

Decimal number	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
110						1	1	0
3475					3	4	7	5
21509				2	1	5	0	9

Figure 3-7 How to use the decimal (*base 10*) number system

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

How Data Is Stored in Internal Memory (cont'd.)

HOW TO Use the Binary (Base 2) Number System

Binary number	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Decimal equivalent
110						1	1	0	6
11010				1	1	0	1	0	26
1001					1	0	0	1	9

Figure 3-8 How to use the binary (*base 2*) number system

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

How Data Is Stored in Internal Memory (cont'd.)

Character	ASCII	Binary	Character	ASCII	Binary	Character	ASCII	Binary
0	48	00110000	K	75	01001011	g	103	01100111
1	49	00110001	L	76	01001100	h	104	01101000
2	50	00110010	M	77	01001101	i	105	01101001
3	51	00110011	N	78	01001110	j	106	01101010
4	52	00110100	O	79	01001111	k	107	01101011
5	53	00110101	P	80	01010000	l	108	01101100
6	54	00110110	Q	81	01010001	m	109	01101101
7	55	00110111	R	82	01010010	n	110	01101110
8	56	00111000	S	83	01010011	o	111	01101111
9	57	00111001	T	84	01010100	p	112	01110000
:	58	00111010	U	85	01010101	q	113	01110001
;	59	00111011	V	86	01010110	r	114	01110010
A	65	01000001	W	87	01010111	s	115	01110011
B	66	01000010	X	88	01011000	t	116	01110100
C	67	01000011	Y	89	01011001	u	117	01110101
D	68	01000100	Z	90	01011010	v	118	01110110
E	69	01000101	a	97	01100001	w	119	01110111
F	70	01000110	b	98	01100010	x	120	01111000
G	71	01000111	c	99	01100011	y	121	01111001
H	72	01001000	d	100	01100100	z	122	01111010
I	73	01001001	e	101	01100101			
J	74	01001010	f	102	01100110			

Figure 3-9 Partial ASCII chart

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting an Initial Value for a Memory Location

- Setting an initial value for a variable or named constant is called initializing
- Required for constants; recommended for variables
- Memory locations are usually initialized with a literal constant (item of data that can appear in a program instruction and be stored in memory)
- Data type of literal constant should match data type of memory location it is assigned to

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting an Initial Value for a Memory Location (cont'd.)

- Numeric literal constants initialize `short`, `int`, `float`, and `double` data types
  - Can contain digits `0` through `9`, `+`, `-`, `.`, and `e` or `E` (for scientific notation)
- Character literal constants initialize `char` data types
  - Consist of one character in single quotation marks
- String literal constants initialize `string` data types
  - Zero or more characters enclosed in double quotation marks
  - Empty string (`""`) is a valid string literal constant

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting an Initial Value for a Memory Location (cont'd.)

- Before assigning initial value to a memory location, computer checks that value's data type matches location's data type
- If they don't match, computer performs implicit type conversion to match them
  - If initial value is converted to type that holds larger numbers, value is promoted
  - If initial value is converted to type that only holds smaller numbers, value is demoted
- Promoting will not usually have adverse effects, but demoting can (information is lost)

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting an Initial Value for a Memory Location (cont'd.)

- Important to initialize memory locations with values of the same data type
- Named constants should be initialized with the value they will hold for the duration of the program
- Variables whose initial values are not known should still be initialized
  - `short` and `int` types usually initialized to `0`
  - `float` and `double` types usually initialized to `0.0`
  - `string` types usually initialized to empty string (`""`)
  - `bool` types initialized to either `true` or `false`

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Selecting an Initial Value for a Memory Location (cont'd.)

IPO chart item	Variable name	Data type	Initial value
sale price	salePrice	double	0.0
sales tax rate	taxRate	double	0.0
sales tax	salesTax	double	0.0
cost	cost	double	0.0

Figure 3-11 Initial values for the variables in the Addison O'Reilly problem

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location

- Variables and named constants are declared using a statement (C++ instruction)
- A statement that declares a variable causes the computer to set aside a memory location with the given name, data type, and initial value
- Statements must follow correct syntax (rules of a programming language)
- In C++, all statements must end with a semicolon

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

- When declaring variables, a data type and name must be provided
- Syntax for declaring a variable in C++
  - `dataType variableName [= initialValue];`
- After variable is declared, you use its name to refer to it later in the program
- Initial value is optional but recommended
- If variable is not initialized, it contains the previous value of that memory location, which may be the wrong type (called a garbage value)

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

- Syntax for declaring a named constant in C++
  - `const` *dataType constantName = value;*
- The `const` keyword indicates that the memory location is a named constant (value cannot be changed during runtime)
- Initial value required for constants, unlike variables
- As with variables, after declaring a constant, you can use its name to refer to it later in the program

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

- Several advantages to using named constants when appropriate
  - Make program more self-documenting (meaningful words in place of numbers)
  - Value cannot be inadvertently changed during runtime
  - Typing a name is less error-prone than a long number
  - Mistyping a constant's name will trigger a compiler error; mistyping a number will not
  - If the constant needs to be changed when modifying the program, it only needs to be changed in one place

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

HOW TO Declare a Variable in C++

Syntax

dataType variableName [= *initialValue*];

Examples

```
int quantity = 0;
double salesTax = 0.0;
bool insured = true;
char grade = ' ';
string city = "";
```

Figure 3-12 How to declare a variable in C++

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

IPO chart item	Variable name	Data type	Initial value	C++ statement
sale price	salePrice	double	??	double salePrice = 0.0;
sales tax rate	taxRate	double	??	double taxRate = 0.0;
sales tax	salesTax	double	??	double salesTax = 0.0;
cost	cost	double	??	double cost = 0.0;

Figure 3-13 C++ declaration statements for the variables in the Addision O'Reilly problem

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Declaring a Memory Location (cont'd.)

HOW TO Declare a Named Constant in C++

Syntax

const *dataType constantName* **=** *value*;

Examples

```
const double PI = 3.141593;
const int MIN_AGE = 65;
const bool PAID = true;
const char YES = 'Y';
const string BANK = "Harrison Trust and Savings";
```

Figure 3-14 How to declare a named constant in C++

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Summary

- Fourth step in problem-solving process is coding the algorithm
- Memory location is declared for each input, processing, and output item in IPO chart
- Numeric data is stored in computer's internal memory using binary number system
- Memory locations store one item at a time
- Memory location's data type determines how a value is stored and interpreted when retrieved

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Summary (cont'd.)

- Two types of memory locations: variables and named constants
- Memory locations are declared using a statement that assigns a name, data type, and initial value
- Initial value required for named constants but optional for variables (though recommended)
- Most memory locations initialized with literal constants, except `bool` (initialized with keywords `true` or `false`)

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Summary (cont'd.)

- Data type of literal constant assigned to memory location should be same as memory location's type
- If types don't match, implicit type conversion is used to either promote or demote so they match
- Promoting doesn't usually cause problems, but demoting can
- Syntax for declaring variables
  - `dataType variableName [= initialValue];`
- Syntax for declaring named constants
  - `const dataType constantName = value;`

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-1: Stop and Analyze

Study the IPO chart shown in Figure 3-15 on page 64 and answer the questions below.

- How many memory locations will the problem require?
- How many of the memory locations will be variables, and how many will be named constants? Why did you choose one over the other?
- How would you write the appropriate declaration statements? Use the *int* data type for the quantity sold, and the double data type for the remaining items.

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-2: Plan and Create

Problem specification

Boughton Inc. wants a program that calculates and displays the amount of a salesperson's commission. The commission is calculated by multiplying the salesperson's sales amount by 10%.

Figure 3-16 Problem specification for Lab 3-2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-2: Plan and Create (cont'd.)

Input	Processing	Output
commission rate (10%) sales amount	Processing items: none Algorithm:	commission

Figure 3-17 Partially completed IPO chart showing input and output items

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-2: Plan and Create (cont'd.)

Input	Processing	Output
commission rate (10%) sales amount	Processing items: none Algorithm: 1. enter the sales amount 2. calculate the commission by multiplying the sales amount by the commission rate 3. display the commission	commission

Figure 3-18 Completed IPO chart for Lab 3-2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-2: Plan and Create (cont'd.)

commission rate	sales amount	commission
~~.1~~	~~1328.50~~	~~132.85~~
.1	2 . 0	2 .

Figure 3-20 Completed desk-check table for Lab 3-2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-2: Plan and Create (cont'd.)

IPO chart information	C++ instructions
Input	
commission rate (10%)	`const double COMM_RATE = 0.1;`
sales amount	`double sales = 0.0;`
Processing	
none	
Output	
commission	`double commission = 0.0;`

Figure 3-21 IPO chart information and C++ instructions for Lab 3-2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-3: Modify

Modify the IPO chart in Figure 3-18 so that it allows the user to enter the commission rate. Then make the appropriate modifications to Figure 3-21.

Input	Processing	Output
commission rate (10%) sales amount	Processing items: none Algorithm: 1. enter the sales amount 2. calculate the commission by multiplying the sales amount by the commission rate 3. display the commission	commission

Figure 3-18 Completed IPO chart for Lab 3-2

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-4: What's Missing?

Professor Merrita wants a program that calculates and displays the volume of a cylinder, given the cylinder's radius (r) and height (h). Given the items listed below in Figure 3-22, create a complete IPO chart similar to Figure 3-21 for the problem.

Items	C++ statements
height	double height = 0.0;
pi (3.14)	double radius = 0.0;
radius	double volume = 0.0;
volume	

Figure 3-22 Items and statements for Lab 3-4

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-5: Desk-Check

After creating an appropriate algorithm for Lab 3-4, desk-check it twice. Use 9 and 6 as the height and radius fro the first desk-check, then use 17 and 15.

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lab 3-6: Debug

- Correct the C++ instructions shown in Figure 3-23
  - The memory locations will store real numbers

IPO chart information	C++ instructions
Input	
first number	`first = 0.0;`
second number	`second = 0.0;`
third number	`third = 0.0;`
Processing	
sum	`sum = 0.0`
Output	
average	`average = 0.0;`

Figure 3-23 IPO chart information and C++ instructions for Lab 3-6

© 2016 Cengage Learning®. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.