Introduction to Programming in C++
Seventh Edition

Chapter 3:
Variables and Constants

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++

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

Objectives

- Go to class website

- Open GitHub
- Log into computer

- Objective: Students should be able to understand different data types

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)

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

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*

Internal Memory (cont'd.)

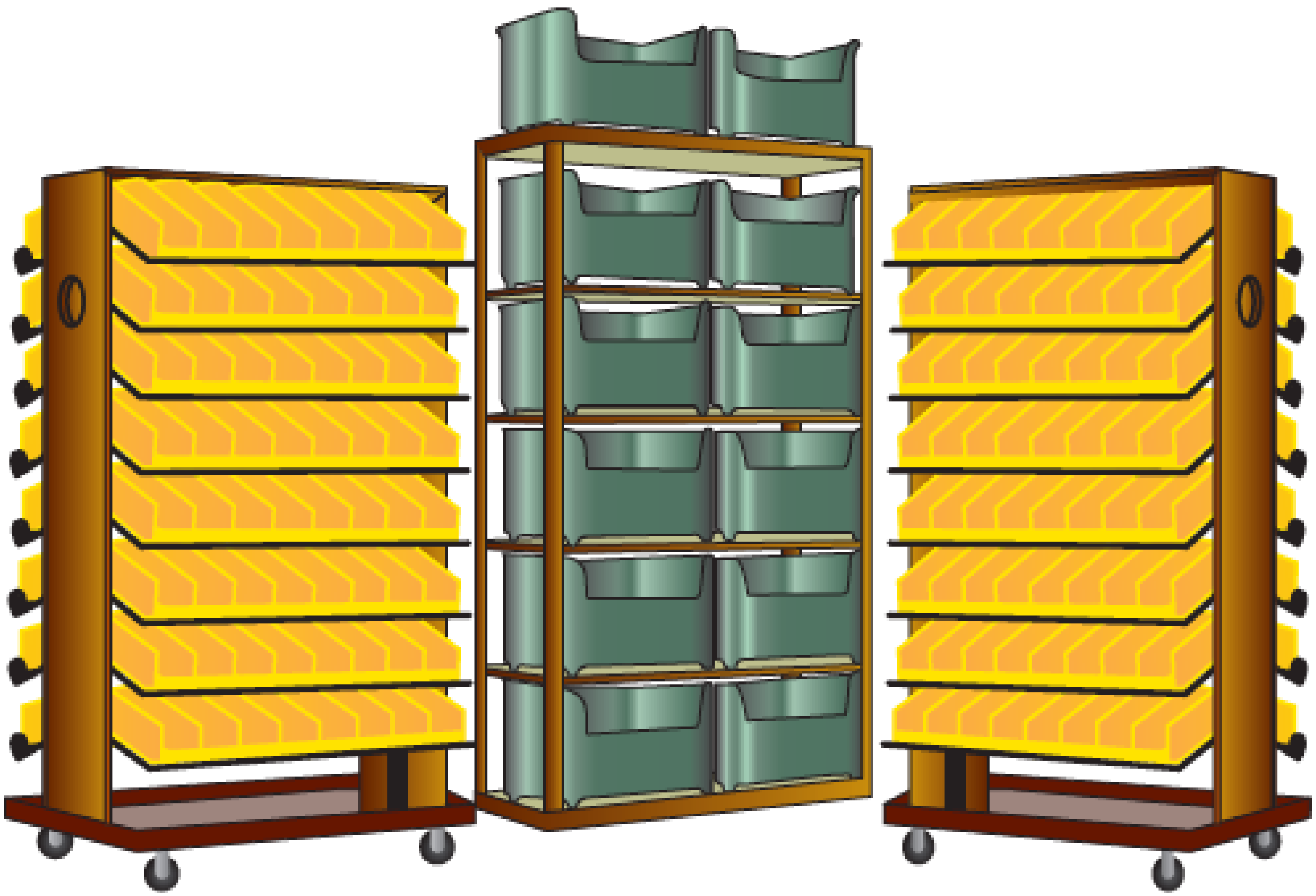

Figure 3-1 Illustration of storage bins

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

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`

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`

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 B contains a listing of keywords in C++.
4. Names in C++ are case sensitive.

Valid names
`grossPay`, `interest`, TAX_RATE, PI

Invalid names	Reason
`2011Sales`	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++

Revisiting the Treyson Mobley Problem

Problem specification
Treyson Mobley wants a program that calculates and displays the amount he should tip a waiter at a restaurant. The program should subtract any liquor charge from the total bill and then calculate the tip (using a percentage) on the remainder.

Input	Processing	Output
total bill liquor charge tip percentage	Processing items: total bill without liquor charge Algorithm: 1. enter the total bill, liquor charge, and tip percentage 2. calculate the total bill without liquor charge by subtracting the liquor charge from the total bill 3. calculate the tip by multiplying the total bill without liquor charge by the tip percentage 4. display the tip	tip

total bill	liquor charge	tip percentage	total bill without liquor charge	tip
~~45~~	~~10~~	~~.2~~	~~35~~	~~7~~
30	0	.15	30	4.50

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

Revisiting the Treyson Mobley Problem (cont'd.)

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

Revisiting the Treyson Mobley Problem (cont'd.)

IPO chart item	Variable name
total bill	`totalBill`
liquor charge	`liquor`
tip percentage	`tipPercent`
total bill without liquor charge	`totalNoLiquor`
tip	`tip`

Figure 3-4 Names of the variables for the Treyson Mobley problem

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

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)

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

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

	Data type	Stores	Memory required
fundamental data types	`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 (either `true` or `false`)	1 byte
	`char`	one character	1 byte
user-defined data type	`string`	zero or more characters	1 byte per character

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

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

IPO chart item	Variable name	Data type
total bill	totalBill	double
liquor charge	liquor	double
tip percentage	tipPercent	double
total bill without liquor charge	totalNoLiquor	double
tip	tip	double

Figure 3-6 Data type assigned to each variable for the Treyson Mobley problem

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

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

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

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

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

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

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)

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`

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

IPO chart item	Variable name	Data type	Initial value
total bill	totalBill	double	0.0
liquor charge	liquor	double	0.0
tip percentage	tipPercent	double	0.0
total bill without liquor charge	totalNoLiquor	double	0.0
tip	tip	double	0.0

Figure 3-10 Initial values for the variables in the Treyson Mobley problem

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

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)

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

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

Declaring a Memory Location (cont'd.)

HOW TO Declare a Variable in C++

Syntax

dataType variableName [= *initialValue*];

Examples

```
int age = 0;
double price = 0.0;
bool paid = false;
char grade = ' ';
string company = "";
```

Figure 3-11 How to declare a variable in C++

Declaring a Memory Location (cont'd.)

IPO chart item	Variable name	Data type	Initial value	C++ statement
total bill	totalBill	double	0.0	double totalBill = 0.0;
liquor charge	liquor	double	0.0	double liquor = 0.0;
tip percentage	tipPercent	double	0.0	double tipPercent = 0.0;
total bill without liquor charge	totalNoLiquor	double	0.0	double totalNoLiquor = 0.0;
tip	tip	double	0.0	double tip = 0.0;

Figure 3-12 C++ declaration statements for the variables in the Treyson Mobley problem

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 INSURED = true;
const char YES = 'Y';
const string BANK = "Harrison Trust and Savings";
```

Figure 3-13 How to declare a named constant in C++

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

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`)

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*;

Lab 3-1: Stop and Analyze

Problem specification

Aiden Nelinski is paid every Friday. He is scheduled to receive anywhere from a 2% to 4.5% raise next week. He wants a program that calculates and displays the amount of his new weekly pay.

Input	Processing	Output
current weekly pay raise percentage	Processing items: none Algorithm: 1. enter the current weekly pay and raise percentage 2. calculate the new weekly pay by multiplying the current weekly pay by the raise percentage and then adding the result to the current weekly pay 3. display the new weekly pay	new weekly pay

current weekly pay	raise percentage	new weekly pay
~~300~~	~~.02~~	~~306~~
500	.025	512.50

Figure 3-14 Problem specification, IPO chart, and desk-check table for Lab 3-1

Lab 3-2: Plan and Create

Professor Chang wants a program that calculates and displays the area of a circle, given the circle's radius. The formula for calculating the area of a circle is πr^2, where π and r represent pi and the radius, respectively. The professor wants to use the value of pi rounded to two decimal places, which is 3.14.

Figure 3-15 Problem specification for Lab 3-2

Lab 3-3: Modify

- Modify the IPO chart in Figure 3-17 so that it includes the radius squared as a processing item

Input	Processing	Output
radius pi (3.14)	Processing items: none Algorithm: 1. enter the radius 2. calculate the area by multiplying the radius by itself and then multiplying the result by pi 3. display the area	area

Figure 3-17 Completed IPO chart for Lab 3-2

Lab 3-4: Desk-Check

- Using the IPO chart modified in Lab 3-3, modify the manual calculations and desk-check tables shown in Figures 3-18 and 3-19

First desk-check		Second desk-check	
4	(radius)	5.5	(radius)
* 4	(radius)	* 5.5	(radius)
* 3.14	(pi)	* 3.14	(pi)
50.24	(area)	94.985	(area)

Figure 3-18 Manual area calculations for the two desk-checks

radius	pi	area
~~4~~	~~3.14~~	~~50.24~~
5.5	3.14	94.985

Figure 3-19 Completed desk-check table for Lab 3-2

Lab 3-5: Debug

- Correct the C++ instructions shown in Figure 3-21
  - The memory locations will store real numbers
- Find and correct the errors in the algorithm

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 3-21 IPO chart information and C++ instructions for Lab 3-5