Data types and operators in C++

Objectives

- In this chapter, you will learn about:
  - Data Types
  - Arithmetic Operators
  - Variables and Declarations
  - Common Programming Errors
  - Bits, Bytes, and Binary Number Representations

Data Types

- The objective of all programs is to process data
- Data is classified into specific types
  - Numerical
  - Alphabetical
  - Audio
  - Video
- C++ allows only certain operations to be performed on certain types of data
  - Prevents inappropriate programming operations

Data Types (cont'd.)

- Data type
  - Set of values and operations that can be applied to these values
- Example of a data type: Integer
  - The values: set of all Integer (whole) numbers
  - The operations: familiar mathematical and comparison operators

Data Types (cont'd.)

- Class data type
  - Programmer-created data type
  - Set of acceptable values and operations defined by a programmer using C++ code
- Built-in data type
  - Provided as an integral part of C++
  - Also known as a primitive type
  - Requires no external code
  - Consists of basic numerical types
  - Majority of operations are symbols (e.g. +,-,*,…)

Data Types (cont'd.)

- Literal
  - Value explicitly identifies itself
- The numbers 2, 3.6, and –8.2 are literals
  - Values are literally displayed
- The text “Hello World!” is a literal
    - Text itself is displayed
- Literals also known as literal values and constants

Integer Data Types

- C++ provides nine built-in integer data types
- Three most important
  - `int`
  - `char`
  - `bool`
- Reason for remaining types is historical
  - Originally provided for special situations
  - Difference among types based on storage requirements

Integer Data Types (cont'd.)

- `int` data type
  - Set of values supported are whole numbers
  - Whole numbers mathematically known as integers
- Explicit signs allowed
- Commas, decimal points, and special signs not allowed
- Examples of `int`:
  - Valid: 0 5 -10 +25 1000 253 -26351 +36
  - Invalid: $255.62 2,523 3. 6,243,982 1,492.89

Integer Data Types (cont'd.)

- Different compilers have different internal limits on the largest and smallest values that can be stored in each data type
  - Most common allocation for `int` is four bytes

Integer Data Types (cont'd.)

- `char` data type
  - Used to store single characters
  - Letters of the alphabet (upper- and lowercase)
  - Digits 0 through 9
  - Special symbols such as + $. , - !
- Single character value: letter, digit, or special character enclosed in single quotes
  - Examples 'A' '$' 'b' '7' 'y' '!' 'M' 'q'

Integer Data Types (cont'd.)

- Character values stored in ASCII or Unicode codes
- ASCII: American Standard Code for Information Exchange
  - Provides English language-based character set plus codes for printer and display control
  - Each character code contained in one byte
  - 256 distinct codes

Integer Data Types (cont'd.)

- Unicode: provides other language character sets
  - Each character contained in two bytes
  - Can represent 65,536 characters
- First 256 Unicode codes have same numerical value as the 256 ASCII codes

Integer Data Types (cont'd.)

- The escape character
  - Backslash (\ )
  - Special meaning in C++
  - Placed in front of a group of characters, it tells the compiler to escape from normal interpretation of these characters
- Escape sequence: combination of a backslash and specific characters
  - Example: newline escape sequence, `\n`

Integer Data Types (cont'd.)

- `bool` data type
  - Represents boolean (logical) data
  - Restricted to `true` or `false` values
- Often used when a program must examine a specific condition
  - If condition is `true`, the program takes one action; if `false`, it takes another action
- Boolean data type uses an integer storage code

Determining Storage Size

- C++ makes it possible to see how values are stored
- `sizeof()`
  - Provides the number of bytes required to store a value for any data type
  - Built-in operator that does not use an arithmetic symbol

Determining Storage Size (cont'd.)

- Signed data type: stores negative, positive, and zero values
- Unsigned data type: stores positive and zero values
  - Provides a range of positive values double that of unsigned counterparts
- Some applications only use unsigned data types
  - Example: date applications in form *yearmonthday*

Floating-Point Types

- The number zero or any positive or negative number that contains a decimal point
  - Also called real number
  - Examples: +10.625 5. -6.2 3521.92 0.0
  - 5. and 0.0 are floating-point, but same values without a decimal (5, 0) would be integers
- C++ supports three floating-point types
  - `float`, `double`, `long double`
  - Different storage requirements for each

Floating-Point Types (cont'd.)

- Most compilers use twice the amount of storage for `double`s as for `float`s
  - Allows a `double` to have approximately twice the precision of a `float`
- A `float` value is sometimes referred to as a **single-precision number**
- A `double` value is sometimes referred to as a **double-precision number**

Exponential Notation

- Floating-point numbers can be written in exponential notation
  - Similar to scientific notation
  - Used to express very large and very small values in compact form

Decimal Notation	Exponential Notation	Scientific Notation
162.5	1.625e2	1.625×10^{2}
63421.	6.3421e4	6.3421×10^{4}
.00731	7.31e-3	7.31×10^{-3}
.000625	6.25e-4	6.25×10^{-4}

Arithmetic Operations

Operation	Operator
Addition	+
Subtraction	–
Multiplication	*
Division	/
Modulus division[3]	%

- Binary operators: require two operands
  - Can be a literal or an identifier
- Binary arithmetic expression:
  - `literalValue operator literalValue`

Arithmetic Operators

- `cout` allows for display of result of a numerical expression
  - Display is on standard output device
- Example:
  - `cout << "The sum of 6 and 15 is " << (6 + 15)`
  - Statement sends string and value to `cout`
  - String: "The total of 6 and 15 is "
  - Value: value of the expression 6 + 15
  - Display produced: The total of 6 and 15 is 21

Arithmetic Operators (cont'd.)

- Manipulator
  - Item used to change how an output stream of characters is displayed
  - `endl` manipulator causes a newline character ('`\n`') to be inserted in the display first
    - Then forces all current insertions to be displayed immediately

Expression Types

- Mixed-mode expression
  - Arithmetic expression containing integer and noninteger operands
- Rule for evaluating arithmetic expressions
  - Both operands are integers: result is integer
  - One operand is floating-point: result is floating-point
- Overloaded operator: a symbol that represents more than one operation
  - Execution depends on types of operands

Integer Division

- Division of two integers yields an integer
  - Integers cannot contain a fractional part; results may seem strange
  - Example: integer 15 divided by integer 2 yields the integer result 7
- Modulus operator (%): captures the remainder
  - Also called the remainder operator
  - Example: 9 % 4 is 1 (remainder of 9/4 is 1)

Negation

- A unary operation that negates (reverses the sign of) the operand
- Uses same sign as binary subtraction (-)

Operator Precedence and Associativity

- Rules for expressions with multiple operators
  - Two binary operators cannot be placed side by side
  - Parentheses may be used to form groupings
  - Expressions within parentheses are evaluated first
  - Sets of parentheses may be enclosed by other parentheses
  - Parentheses cannot be used to indicate multiplication (multiplication operator (*) must be used)

Variables and Declarations

- Symbolic names used in place of memory addresses
  - Symbolic names are called variables
  - These variables refer to memory locations
  - The value stored in the variable can be changed
  - Simplifies programming effort
- Assignment statement: assigns a value to a variable
  - Format: `variable name = value assigned;`
  - Example: `num1 = 45;`

Declaration Statements

- Names a variable and specifies its data type
  - General form: `dataType variableName;`
  - Example: `int sum;` declares `sum` as variable which stores an integer value
- Declaration statements can be placed anywhere in function
  - Typically grouped together and placed immediately after the function's opening brace
- Variable must be declared before it can be used

Multiple Declarations

- Variables with the same data type can be grouped together and declared in one statement
  - Format: `dataType variableList;`
  - Example: `double grade1, grade2, total, average;`
- Initialization: using a declaration statement to store a value in a variable
  - Good programming practice is to declare each initialized variable on a line by itself
  - Example: `double grade2 = 93.5;`

Memory Allocation

- Each data type has its own storage requirements
  - Computer must know variable's data type to allocate storage
  - Definition statements: declaration statements used for the purpose of allocating storage

Common Programming Errors

- Forgetting to declare all variables used in a program
- Attempting to store one data type in a variable declared for a different type
- Using a variable in an expression before the variable is assigned a value
- Dividing integer values incorrectly

Common Programming Errors (cont'd.)

- Mixing data types in the same expression without clearly understanding the effect produced
  - It is best not to mix data types in an expression unless a specific result is desired
- Forgetting to separate individual data streams passed to `cout` with an insertion (“put to”) symbol

Summary

- Four basic types of data recognized by C++
  - Integer, floating-point, character, boolean
- `cout` object can be used to display all data types
- Every variable in a C++ program must be declared as the type of variable it can store

Summary (cont'd.)

- A simple C++ program containing declaration statements has the format:

```
#include <iostream>
using namespace std;
int main()
{
  declaration statements;

  other statements;

  return 0;
}
```

Summary (cont'd.)

- Declaration statements: inform the compiler of function's valid variable names
- Definition statements: declaration statements that also cause computer to set aside memory locations for a variable
- `sizeof()` operator: determines the amount of storage reserved for a variable

Chapter Supplement: Bits, Bytes, and Binary Number Representations

- This section explains how numbers are stored in a computer's memory and different means of representing them

Bits and Bytes

- **Bit**
  - A switch that can be open or closed
- **Byte**
  - Group of eight bits
- **Character code**
  - Collection of patterns used to represent letters, single digits, and other characters
- **Number codes**
  - Patterns used to store numbers
- **Words and addresses**