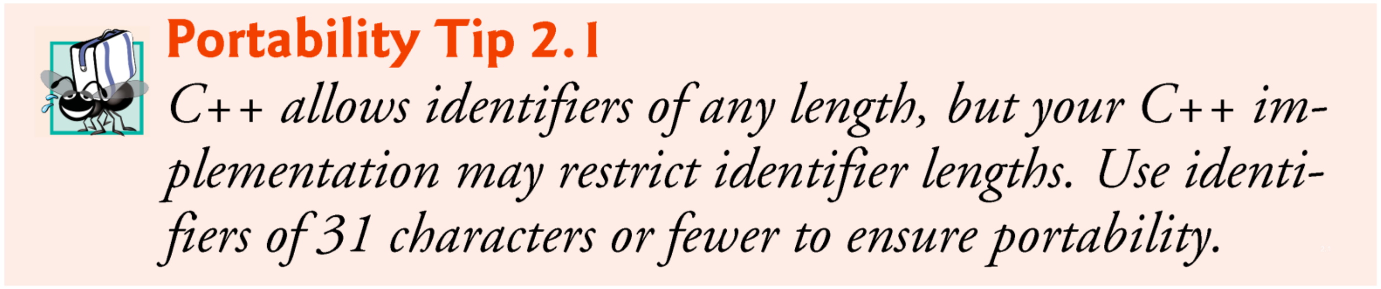
Lecture 5:

Introduction to C++ Programming (cont)

Ioan Raicu
Department of Electrical Engineering & Computer Science
Northwestern University

EECS 211
Fundamentals of Computer Programming II
April 5th, 2010

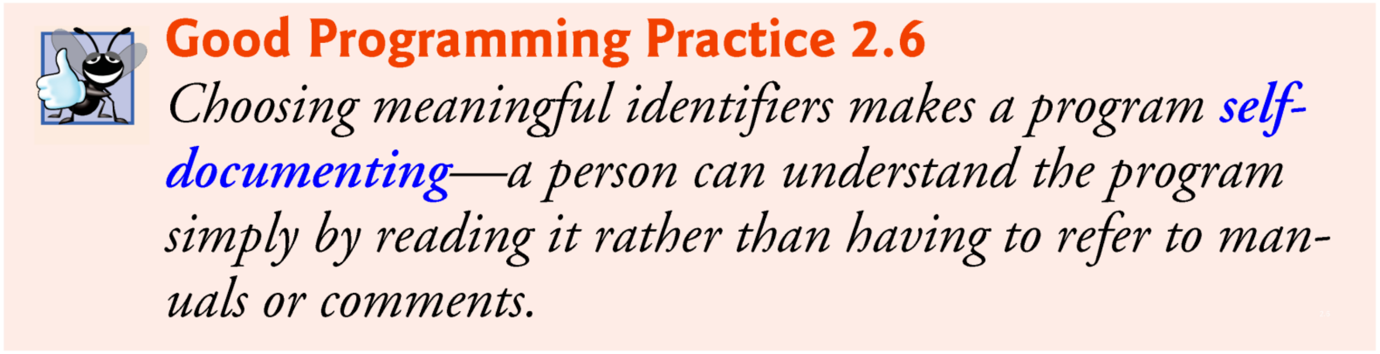
2.4 Another C++ Program: Adding Integers (cont.)



Portability Tip 2.1

C++ allows identifiers of any length, but your C++ implementation may restrict identifier lengths. Use identifiers of 31 characters or fewer to ensure portability.

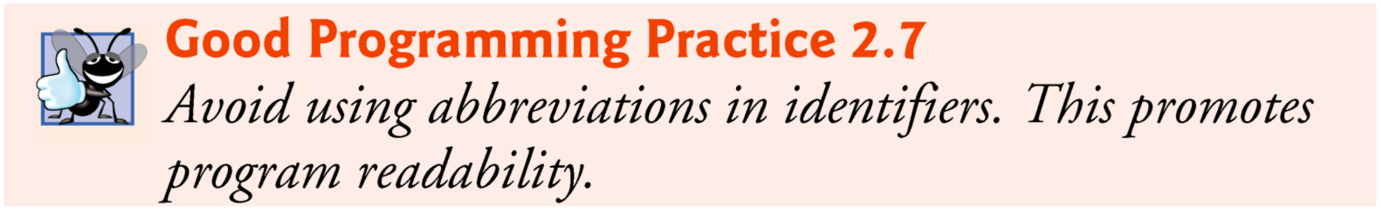
2.4 Another C++ Program: Adding Integers (cont.)



Good Programming Practice 2.6

Choosing meaningful identifiers makes a program ***self-documenting****—a person can understand the program simply by reading it rather than having to refer to manuals or comments.*

2.4 Another C++ Program: Adding Integers (cont.)



Good Programming Practice 2.7

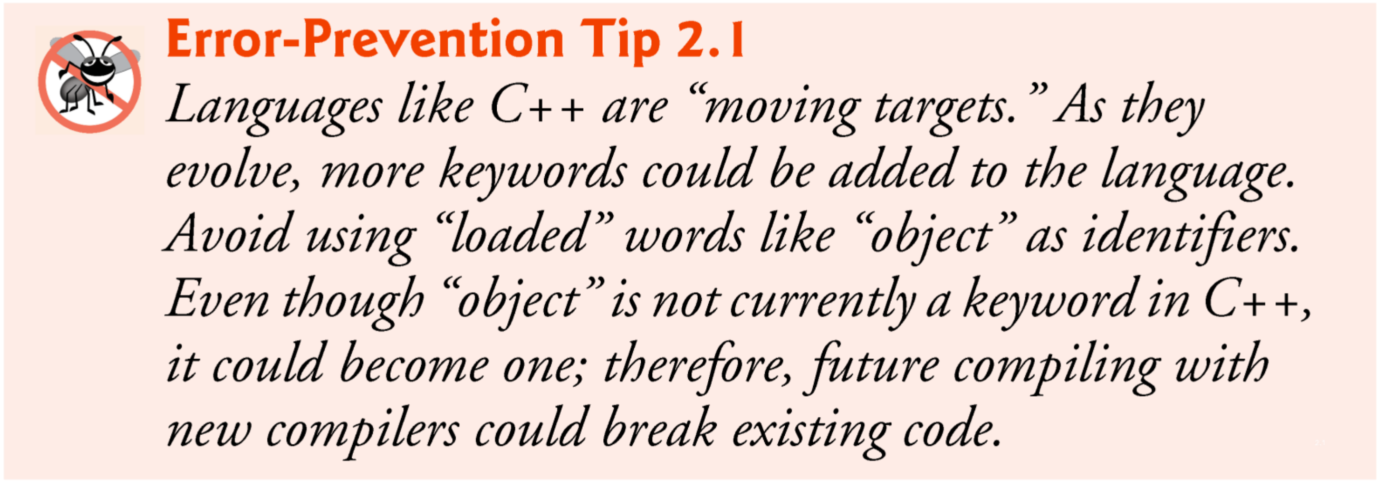
Avoid using abbreviations in identifiers. This promotes program readability.

2.4 Another C++ Program: Adding Integers (cont.)

Good Programming Practice 2.8

Avoid identifiers that begin with underscores and double underscores, because C++ compilers may use names like that for their own purposes internally. This will prevent names you choose from being confused with names the compilers choose.

2.4 Another C++ Program: Adding Integers (cont.)



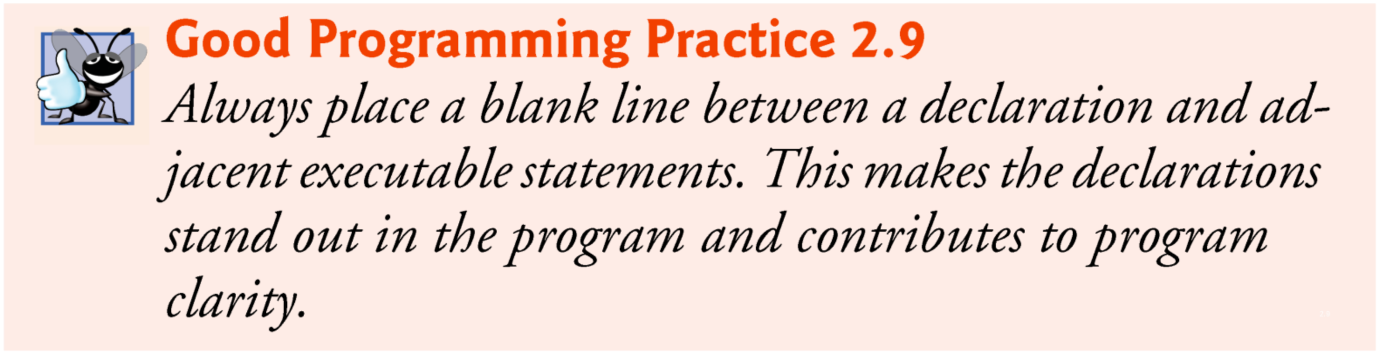
Error-Prevention Tip 2.1

Languages like C++ are "moving targets." As they evolve, more keywords could be added to the language. Avoid using "loaded" words like "object" as identifiers. Even though "object" is not currently a keyword in C++, it could become one; therefore, future compiling with new compilers could break existing code.

2.4 Another C++ Program: Adding Integers (cont.)

- Declarations of variables can be placed almost anywhere in a program, but they must appear before their corresponding variables are used in the program.

2.4 Another C++ Program: Adding Integers (cont.)



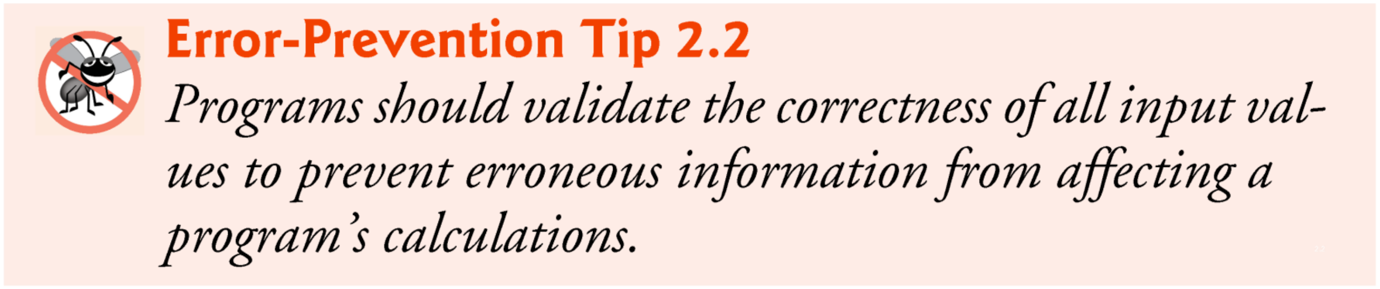
Good Programming Practice 2.9

Always place a blank line between a declaration and adjacent executable statements. This makes the declarations stand out in the program and contributes to program clarity.

2.4 Another C++ Program: Adding Integers (cont.)

- A prompt it directs the user to take a specific action.
- A `cin` statement uses the input stream object cin (of namespace `std`) and the stream extraction operator, >>, to obtain a value from the keyboard.
- Using the stream extraction operator with `std::cin` takes character input from the standard input stream, which is usually the keyboard.

2.4 Another C++ Program: Adding Integers (cont.)



Error-Prevention Tip 2.2

Programs should validate the correctness of all input values to prevent erroneous information from affecting a program's calculations.

2.4 Another C++ Program: Adding Integers (cont.)

- When the computer executes an input statement that places a value in an `int` variable, it waits for the user to enter a value for variable `number1`.
- The user responds by typing the number (as characters) then pressing the *Enter* key (sometimes called the Return key) to send the characters to the computer.
- The computer converts the character representation of the number to an integer and assigns (i.e., copies) this number (or value) to the variable `number1`.
- Any subsequent references to `number1` in this program will use this same value.
- In this program, an assignment statement adds the values of variables `number1` and `number2` and assigns the result to variable `sum` using the assignment operator `=`.
  - Most calculations are performed in assignment statements.

2.4 Another C++ Program: Adding Integers (cont.)

- The = operator and the + operator are called binary operators because each has two operands.

2.4 Another C++ Program: Adding Integers (cont.)

Good Programming Practice 2.10

Place spaces on either side of a binary operator. This makes the operator stand out and makes the program more readable.

2.4 Another C++ Program: Adding Integers (cont.)

- `std::endl` is a so-called stream manipulator.
- The name `endl` is an abbreviation for "end line" and belongs to namespace `std`.
- The `std::endl` stream manipulator outputs a newline, then "flushes the output buffer."
  - This simply means that, on some systems where outputs accumulate in the machine until there are enough to "make it worthwhile" to display them on the screen, `std::endl` forces any accumulated outputs to be displayed at that moment.
  - This can be important when the outputs are prompting the user for an action, such as entering data.

2.4 Another C++ Program: Adding Integers (cont.)

- Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.
- Calculations can also be performed in output statements.

2.5 Memory Concepts

- Variable names such as `number1`, `number2` and `sum` actually correspond to locations in the computer's memory.
- Every variable has a name, a type, a size and a value.
- When a value is placed in a memory location, the value overwrites the previous value in that location; thus, placing a new value into a memory location is said to be destructive.
- When a value is read out of a memory loca-tion, the process is nondestructive.

2.5 Memory Concepts

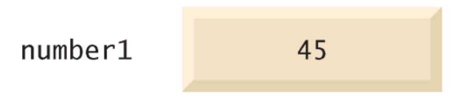


Fig. 2.6 | Memory location showing the name and value of variable `number1`.

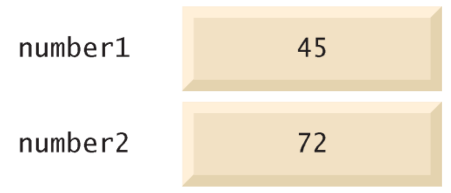


Fig. 2.7 | Memory locations after storing values for `number1` and `number2`.

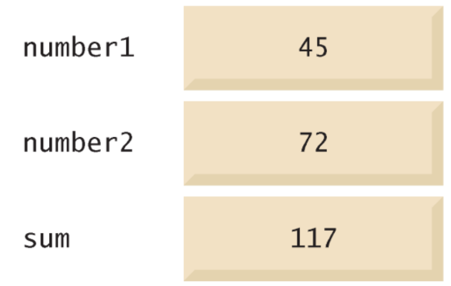


Fig. 2.8 | Memory locations after calculating and storing the `sum` of `number1` and `number2`.

2.6 Arithmetic

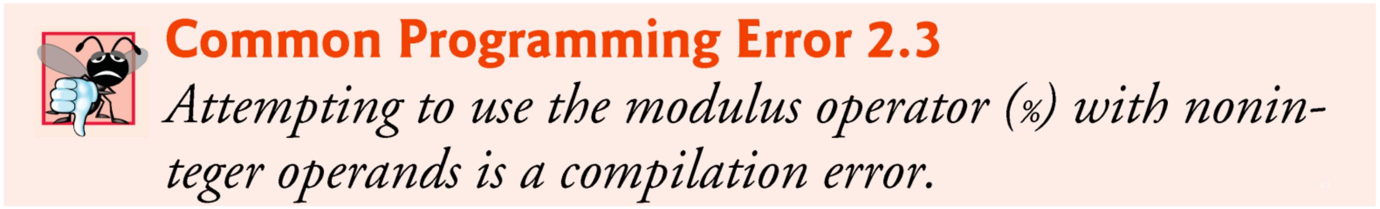
- Most programs perform arithmetic calculations.
- Figure 2.9 summarizes the C++ arithmetic operators.
- The asterisk (*) indicates multiplication.
- The percent sign (%) is the modulus operator that will be discussed shortly.
  - C++ provides the modulus operator, %, that yields the remainder after integer division.
  - The modulus operator can be used only with integer operands.
- The arithmetic operators in Fig. 2.9 are all binary operators.
- Integer division (i.e., where both the numerator and the denominator are integers) yields an integer quotient.
  - Any fractional part in integer division is discarded (i.e., truncated)—no rounding occurs.

2.5 Memory Concepts

C++ operation	C++ arithmetic operator	Algebraic expression	C++ expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`-`	$p - c$	`p - c`
Multiplication	`*`	bm or $b \cdot m$	`b * m`
Division	`/`	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Modulus	`%`	$r \bmod s$	`r % s`

Fig. 2.9 | Arithmetic operators.

2.5 Memory Concepts



Common Programming Error 2.3

Attempting to use the modulus operator (`%`) with noninteger operands is a compilation error.

2.6 Arithmetic (cont.)

- Arithmetic expressions in C++ must be entered into the computer in straight-line form.
- Expressions such as "`a` divided by `b`" must be written as `a / b`, so that all constants, variables and operators appear in a straight line.
- Parentheses are used in C++ expressions in the same manner as in algebraic expressions.
- For example, to multiply `a` times the quantity `b + c` we write `a a * ( b + c )`.

2.6 Arithmetic (cont.)

- C++ applies the operators in arithmetic expressions in a precise sequence determined by the following rules of operator precedence, which are generally the same as those followed in algebra.

2.6 Arithmetic (cont.)

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they're evaluated left to right.
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Fig. 2.10 | Precedence of arithmetic operators.

2.6 Arithmetic (cont.)

- There is no arithmetic operator for exponentiation in C++, so x^2 is represented as `x * x`.
- Figure 2.11 illustrates the order in which the operators in a second-degree polynomial are applied.
- As in algebra, it's acceptable to place unnecessary parentheses in an expression to make the expression clearer.
- These are called redundant parentheses.

Questions

?