Standard Version of Starting Out with C++, 4th Edition

Chapter 2

Introduction to C++

Topics

2.1 Parts of a C++ Program

2.2 The `cout` Object

2.3 The `#include` Directive

2.4 Variables and Constants

2.5 Identifiers

2.6 Integer Data Types

2.7 The `char` Data Type

2.8 Floating-Point Data Types

Topics (continued)

2.9 The `bool` Data Type

2.10 Determining the Size of a Data Type

2.11 Variable Assignments and Initialization

2.12 Scope

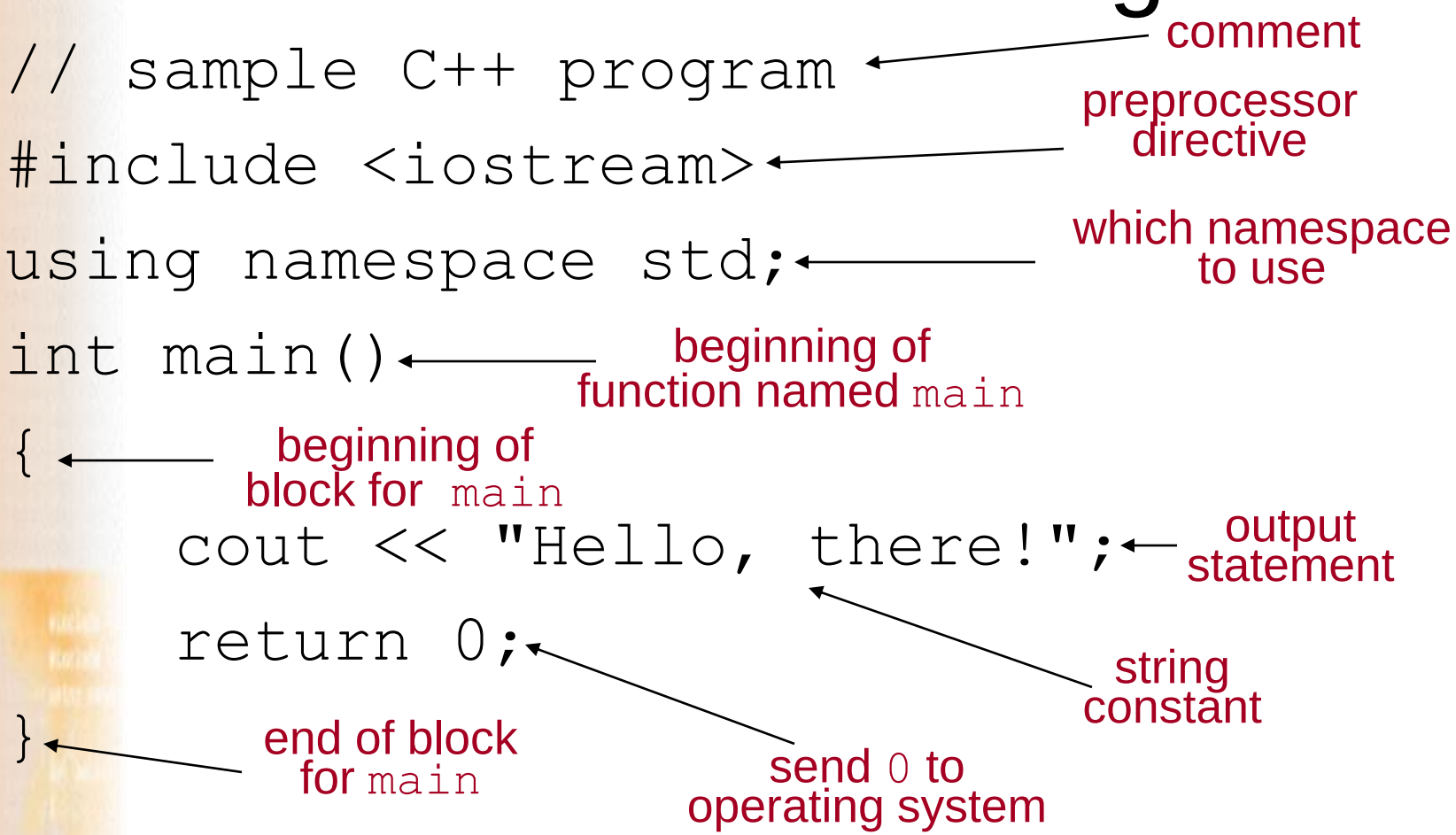
2.13 Arithmetic Operators

2.14 Comments

2.15 Programming Style

2.16 Standard and Prestandard C++

2.1 Parts of a C++ Program



Special Characters

Character	Name	Meaning
//	Double slash	Beginning of a comment
#	Pound sign	Beginning of preprocessor directive
< >	Open/close brackets	Enclose filename in #include
( )	Open/close parentheses	Used when naming a function
{ }	Open/close brace	Encloses a group of statements
" "	Open/close quotation marks	Encloses string of characters
;	Semicolon	End of a programming statement

2.2 The `cout` Object

- Displays information on computer screen
- Uses `<<` to send information to cout:

```
cout << "Hello, there!";
```

- Can be used to send > 1 item to cout:

```
cout << "Hello, " << "there!";
```

Or:

```
cout << "Hello, ";
cout << "there!";
```

The `cout` Object

- To get multiple lines of output on screen:
  - Use `endl`

```
cout << "Hello, there!" << endl;
```

  - Use `\n` in output string

```
cout << "Hello, there!\n";
```

2.3 The `#include` Directive

- Inserts the contents of another file into the program
- Preprocessor directive, not part of C++ language
- `#include` lines not seen by compiler
- Do not use ; at end of `#include` line

2.4 Variables and Constants

- Variable: storage location in memory
  Not the same meaning as in Math!
  - Has a name and a type of data it can hold
  - Must be defined before it can be used:

```
int item;
```

- Constant: item whose value does not change during program execution

```
"hello, there"
```
(string constant)

```
12
```
(integer constant)

2.5 Identifiers

- Programmer-chosen names to represent parts of the program: variables, functions, etc.
- Name should represent the use of the variable
- Cannot use C++ key words as identifiers
- Must begin with alpha character or _, followed by alpha, numeric, or _
- Upper- and lower-case characters are distinct

Valid and Invalid Identifiers

IDENTIFIER	VALID?	REASON IF INVALID
`totalSales`	Yes	
`total_Sales`	Yes	
`total.Sales`	No	Cannot contain .
`4thQtrSales`	No	Cannot begin with digit
`totalSale$`	No	Cannot contain $

2.6 Integer Data Types

- Designed to hold whole numbers
- Can be `signed` or `unsigned`:

  `-12  -6  +3`

- Available in different sizes (number of bytes): `short`, `int`, and `long`
- Size of `short` ≤ size of `int` ≤ size of `long`

Defining Variables

- Variables of the same type can be defined
  - On separate lines:

```
int length;
int width;
unsigned int area;
```

  - On the same line:

```
int length, width;
unsigned int area;
```

- Variables of different types must be in different definitions

Integral Constants

- Integer constants stored in as `int`s by default
- To store an integer constant in a long memory location, put '`L`' at the end of the number:
  `1234L`
- Constants that begin with '`0`' (zero) are base 8:
  `075`
- Constants that begin with '`0x`' are base 16:
  `0x75A`

2.7 The `char` Data Type

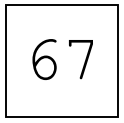
- Used to hold characters or very small integer values
- Usually 1 byte of memory
- Numeric value of character from character set is stored in memory:

CODE:

```
char letter;
letter = 'C';
```

MEMORY:

letter



Character Strings

- Can store a series of characters in consecutive memory locations:

  `"Hello"`

- Stored with the null terminator, `\0`, at the end:

H	e	l	l	o	\0

- Comprised of the characters between the `" "`

2.8 Floating-Point Data Types

- Designed to hold real numbers

  `12.45` `-3.8`
- Stored in a form similar to scientific notation
- All numbers are signed
- Available in different sizes (number of bytes): `float`, `double`, and `long double`
- Size of `float` ≤ size of `double`
  ≤ size of `long double`

Floating-point Constants

- Can be represented in
  - Fixed point (decimal) notation:

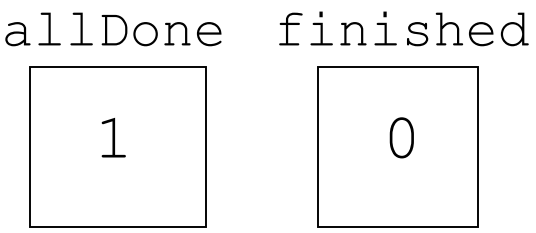
    `31.4159` `0.0000625`
  - E notation:

    `3.14159E1` `6.25e-5`
- Are `double` by default
- Can be forced to be float (`3.14159f`) or long double (`0.0000625L`)

2.9 The `bool` Data Type

- Represents values that are `true` or `false`
- `bool` variables are stored as small integers
- `false` is represented by 0, `true` by 1:

```
bool allDone = true;
bool finished = false;
```



2.10 Determining the Size of a Data Type

The `sizeof` operator gives the size of any data type or variable:

```
float amount;
cout << "A float is stored in "
     << sizeof(float) << "bytes\n";
cout << "Variable amount is stored in "
     << sizeof(amount)
     << "bytes\n";
```

2.11 Variable Assignments and Initialization

Assignment:

- Uses the = operator
- Has a single variable on the left side and a value (constant, variable, or expression) on the right side
- Copies the value on the right into the variable on the left:

```
item = 12;
```

Variable Assignments and Initialization

- Initialize a variable: assign it a value when it is defined:

```
int length = 12;
```

- Can initialize some or all variables:

```
int length = 12, width = 5, area;
```

2.12 Scope

- The scope of a variable: where the program can access the variable
- A variable cannot be used before it is defined

2.13 Arithmetic Operators

- Used for performing numeric calculations
- C++ has unary, binary, and trinary operators:
  - unary (1 operand) `-5`
  - binary (2 operands) `13 - 7`
  - ternary (3 operands) `exp1 ? exp2 : exp3`

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	VALUE OF `ans`
+	addition	`ans = 7 + 3;`	10
-	subtraction	`ans = 7 - 3;`	4
*	multiplication	`ans = 7 * 3;`	21
/	division	`ans = 7 / 3;`	2
%	modulus	`ans = 7 % 3;`	1

/ Operator

- / (division) operator performs integer division if both operands are integers

```
cout << 13 / 5;     // displays 2
cout << 91 / 7;     // displays 13
```

- If either operand is floating point, the result is floating point

```
cout << 13 / 5.0;   // displays 2.6
cout << 91.0 / 7;   // displays 13.0
```

% Operator

- % (modulus) operator computes the remainder resulting from integer division

```
cout << 13 % 5;   // displays 3
```

- % requires integers for both operands

```
cout << 13 % 5.0; // error
```

2.14 Comments

- Used to document parts of the program
- Intended for persons reading the source code of the program:
  - Indicate the purpose of the program
  - Describe the use of variables
  - Explain complex sections of code
- Are ignored by the compiler

C++ Style Comments

Begin with `//` through to the end of line:

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area

// calculate rectangle area
area = length * width;
```

C-Style Comments

- Begin with `/*`, end with `*/`
- Can span multiple lines:

```
/* this is a multi-line
   C-style comment
*/
```

- Can be used like C++ style comments:

```
int area;   /* calculated area */
```

2.15 Programming Style

- The visual organization of the source code
- Includes the use of spaces, tabs, and blank lines
- Does not affect the syntax of the program
- Affects the readability of the source code

Programming Style

Common elements to improve readability:

- Braces { } aligned vertically
- Indentation of statements within a set of braces
- Blank lines between declaration and other statements
- Long statements wrapped over multiple lines with aligned operators

2.16 Standard and Prestandard C++

Older-style C++ programs:

- Use `.h` at end of header files:

```
#include <iostream.h>
```

- Do not use `using namespace` convention
- May not compile with a standard C++ compiler

Standard Version of Starting Out with C++, 4th Edition

Chapter 2

Introduction to C++