Week 1

Operators and Data Types, I/O

Gaddis: Chapters 1, 2, 3

CS 5301 Spring 2014

Jill Seaman

1

3

Structure of a C++ Program

• Hello world:

//This program outputs a message to the screen
#include <iostream>
using namespace std;

int main() {
 cout << "Hello world!" << endl;</pre>

• In general:

//This is a comment
#include <includefile> ...
using namespace std;

int main() {
 statements ...

Variables, Data Types

- Variable: portion of memory that stores a value
- Identifier: name of a program element
- Fundamental data types

short int long	float double long double	bool char				
 Variable Declaration statement 						
datatype identifier;			float hours;			
Variable Initialization statement:						
<pre>datatype identifier = constant;</pre>		ant;	<pre>int count = 0;</pre>			

Integer types

• Integers are whole numbers such as 12, 7, and -99

Data Type	Range
short	-23,768 to 32,767
int	-2,147,483,648 to 2,147,483,647
long	-2,147,483,648 to 2,147,483,647

- char type stores characters such as 'A', '@', and '9'
 - The ascii code value (an integer) of the character is stored in memory.

2

Floating-point types

- Floating point types store real numbers such as 12.45 and -3.8
- They are stored using scientific notation.

Data Type	Range
float	±3.4E-38 to ±3.4E38
double	$\pm 1.7E-308$ to $\pm 1.7E308$
long double	±1.7E-308 to ±1.7E308

- bool type stores values that are true or false
 - false is 0, true is 1.

Assignment statement, expressions

• To change the value of a variable:

variable = expression;

count = 10;

5

7

- * The lefthand side must be a variable
- * The righthand side is an expression of the right type
- What is an expression?
 - an expression has a type and evaluates to a value
 - literal
 - named constant
 - variable
 - arithmetic expression
 - etc.

Arithmetic Operations

• arithmetic operators:

Named Constants:

+ addition - subtraction * multiplication / division % modulo (remainder)

x +	10
x + 7 *	2
8 -	5 * 10
(3 *	: 10) / 2

'Α'

'2'

• Integer division:

14 ÷ 3 = 4 r. 2 (because 4*3+2 = 14) 14/3 => 4 in C++ 14*3 => 2 in C++ 14.0/3.0 => 4.6666667 in C++

Constants

• Literals (specific value of a given type)

variable whose value cannot be changed

12.45 -3.8	true false
-3.8 6.25e-5	Ialse

const datatype identifier = constant;

const double TAX RATE = 0.0675;

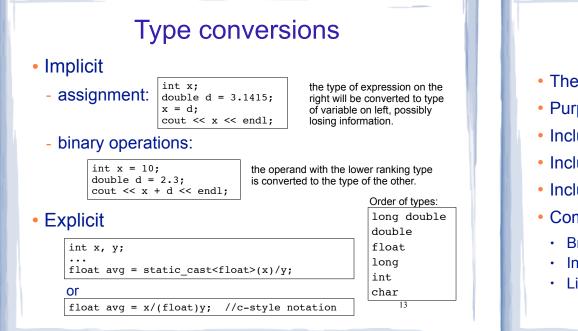
1

75

-2

Operator precedence Basic Input/Output • Output (cout and <<) In an expression with multiple operators, which one happens first? cout << expression;</pre> cout << expr1 << expr2;</pre> • Use this order for different operators: cout << "hello";</pre> + - (unary) cout << "Count is: " << count << endl: * / % + - (binary) Input (cin and >>) <><=>= == != We will study relational and cin >> variable: logical operators next week. && right hand side must be a variable! cin >> var1 >> var2; Use this order for multiple occurrences of the cout << "Enter the height and width: "; same operator cin >> height >> width; cout << "The height is " << height << endl;</pre> – (unary negation) associates right to left • *, /, %, +, - associate left to right 9 10

Formatting output The string class Goal: control how output displays for numeric data string literals: represent sequences of chars: these require #include<iomanip> cout << "Hello";</pre> setw(x): print next value in a field at least x spaces To define string variables: wide (right justified, padded with spaces). cout << setw(6) << 1234 << setw(6) << 5 << endl; 1234 5 string firstName, lastName; cout << setw(6) << 5 << setw(6) << 1234 << endl; 5 1234 fixed: always use decimal notation (not scientific) Operations include: string name = "George"; cout << name.size() << " ";</pre> setprecision(x): when used with fixed, print floating for assignment cout << name[2] << endl;</pre> point values using x digits after the decimal - .size() function for length cout << fixed << setprecision(2);</pre> 3.14 - [n] to access one character in the nth position. cout << 3.14159 << endl; 20.00 float x = 20;cout << x << endl; 11 12



Programming Style

- The visual organization of the source code
- Purpose: improve the readability of the source code
- Includes the use of spaces, tabs, and blank lines
- Includes naming of variables, constants.
- Includes where to use comments.
- · Common elements to improve readability:
 - Braces { } aligned vertically
 - Indentation of statements within a set of braces
 - Lines shorter than 80 characters.

14