Chapter 2: Intro to C++ Programming

C++ How to Program: An Objects-Natural Approach, 11/e

Presented by
Paul Deitel, CEO, Deitel & Associates, Inc.

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Chapter 2: Intro to C++ Programming

- Write simple C++ applications
- `cout`/`cin` for command-line output and input
- Fundamental types
- Declare variables
- Arithmetic, equality and relational operators
- Decision making: `if` statements
- Work with objects of pre-existing classes

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Chapter 2: Intro to C++ Programming

- **“Objects Natural” Approach**
  - Create and use objects of preexisting classes
  - Perform significant tasks with minimal code
  - Classes typically from
    - C++ standard library
    - Free third-party open-source
- Reminder
  - Before You Begin for setup instructions
  - Chapter 1 Test-Drive videos for compile/execute instructions

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Escape Sequences

Escape sequence	Description
`\n`	Newline. Positions the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Moves the screen cursor to the next tab stop.
`\r`	Carriage return. Positions the screen cursor to the beginning of the current line; does not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Includes a backslash character in a string.
`\'`	Single quote. Includes a single-quote character in a string.
`\"`	Double quote. Includes a double-quote character in a string.

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Fundamental Types

- Type `double` for specifying real numbers with decimal points, such as `3.4`, `0.0` and `–11.19`
- Type `char` for specifying a single lowercase letter, uppercase letter, digit or special character (e.g., `$` or `*`)

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Identifiers

- A variable name is any valid **identifier** that is not a keyword
- Consists of letters, digits and underscores (_)
- Must not begin with a digit
- C++ is **case sensitive**
  - `a1` and `A1` are different identifiers
- Avoid identifiers that begin with underscores and double underscores

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Arithmetic Operators

Operation	Arithmetic operator	C++ expression
Addition	`+`	`f + 7`
Subtraction	`-`	`p - c`
Multiplication	`*`	`b * m`
Division	`/`	`x / y`
Remainder	`%`	`r % s`

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Integer Division

- **Integer division** in which the numerator and the denominator are integers yields an integer quotient
- 7 / 4 evaluates to 1
- 17 / 5 evaluates to 3
- Any fractional part in the result is truncated—no rounding occurs

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Remainder Operator

- x **%** y yields the remainder after dividing x by y using integer division
- Can be used only with integer operands
- 7 % 4 yields 3
- 17 % 5 yields 2

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Parentheses for Grouping Subexpressions

- Parentheses group C++ expressions in the same manner as in algebra
- To multiply a times the quantity b + c, write
  - `a * (b + c)`

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Rules of Operator Precedence

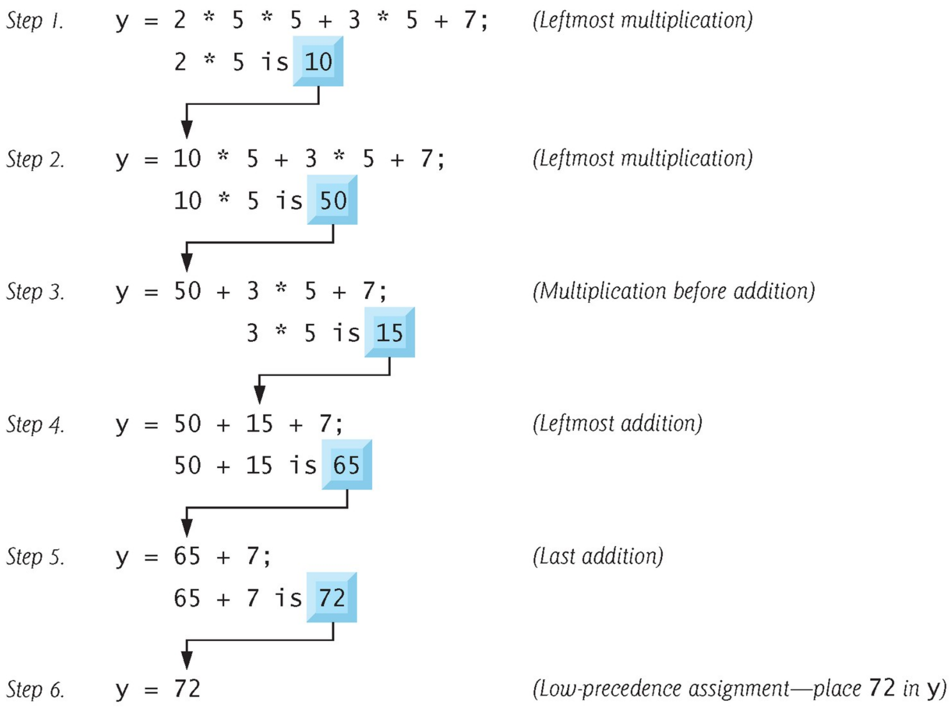
- C++ applies the operators in arithmetic expressions in a precise order determined by the **rules of operator precedence**—generally the same as in algebra
- Expressions in parentheses evaluate first.
- Multiplication, division and remainder operations evaluate next.
  - Applied from left-to-right.
- Addition and subtraction operations evaluate last.
  - Applied from left-to-right.
- **Caution:** If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear “on the same level,” the C++ Standard does not specify the order in which these parenthesized subexpressions will evaluate

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Operator Grouping

- **Grouping** refers to the order in which C++ applies certain operators
- In the expression a + b + c the + operators group from left-to-right as if we had written (a + b) + c
- Most C++ operators of the same precedence group left-to-right
- We’ll see that some operators group right-to-left

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com



Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`2 * 5 is 10`

Step 2. `y = 10 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`10 * 5 is 50`

Step 3. `y = 50 + 3 * 5 + 7;` *(Multiplication before addition)*

`3 * 5 is 15`

Step 4. `y = 50 + 15 + 7;` *(Leftmost addition)*

`50 + 15 is 65`

Step 5. `y = 65 + 7;` *(Last addition)*

`65 + 7 is 72`

Step 6. `y = 72` *(Low-precedence assignment—place* `72` *in* `y`*)*

Fig. 2.11 | Order in which a second-degree polynomial is evaluated.

Operators	Associativity	Type
()	*[See caution in Fig. 2.10]*	grouping parentheses
* / %	left to right	multiplicative
+ -	left to right	additive
<< >>	left to right	stream insertion/extraction
< <= > >=	left to right	relational
== !=	left to right	equality
=	right to left	assignment

Fig. 2.14 | Precedence and associativity of the operators discussed so far.

Equality and Relational Operators

Algebraic operator	C++ operator	Sample condition	Meaning
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Compound Assignment Operators

- `total = total + grade;`
- Assume: c = 3, d = 5, e = 4, f = 6, g = 12

OPERATOR	SAMPLE EXPRESSION	EXPLANATION	ASSIGNS
+=	c += 7	c = c + 7	10 to c
-=	d -= 4	d = d - 4	1 to d
*=	e *= 5	e = e * 5	20 to e
/=	f /= 3	f = f / 3	2 to f
%=	g %= 9	g = g % 9	3 to g

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com

Increment and Decrement Operators

- `passes = passes + 1;`

Operator	Operator name	Sample expression	Explanation
`++`	prefix increment	`++number`	Increment `number` by 1, then use the new value of `number` in the expression in which `number` resides.
`++`	postfix increment	`number++`	Use the current value of `number` in the expression in which `number` resides, then increment `number` by 1.
`--`	prefix decrement	`--number`	Decrement `number` by 1, then use the new value of `number` in the expression in which `number` resides.
`--`	postfix decrement	`number--`	Use the current value of `number` in the expression in which `number` resides, then decrement `number` by 1.

©Copyright 1992-2024 by Pearson Education, Inc. All Rights Reserved. https://deitel.com