Basic C++ Operators, Variable Declaration, Assignment and Conditional Statement

Variable Declaration

- Variable must be declared before use
  - Syntax: data-type identifier ;

- Use a comma-separated list to declare multiple variables (note: they all must have the same data type)
  - Syntax:

```
data-type  identifier1,  identifier2,   identifer3 ;
```

Variable Declaration and Initialization

- Declare and initialize
  - Syntax:

    ```
    data-type   variable-identifier  =   initial-value ;
    ```

  - Declared variable contains undetermined value by default. It is a good programming practice to initialize the variable when it is declared.

- Example of different ways to initialize
  - `int    num = 123 ;`
  - `int    unitPrice(124) ;`
  - `int    count{125} ;`

sizeof operator: C++ data type sizes

```
#include <iostream>
using namespace std;

int main()
{
  cout << "sizeof(bool): " << sizeof(bool) << endl;

  cout << "sizeof(char): " << sizeof(char) << endl;
  cout << "sizeof(char16_t): " << sizeof(char16_t) << endl;

  cout << "sizeof(int): " << sizeof(int) << endl;
  cout << "sizeof(long int): " << sizeof(long int) << endl;
  cout << "sizeof(long long int): " << sizeof(long long int) << endl;

  cout << "sizeof(float): " << sizeof(float) << endl;
  cout << "sizeof(double): " << sizeof(double) << endl;
  cout << "sizeof(long double): " << sizeof(long double) << endl;

  return 0;
}
```

Compound Assignment

- +=, -=, *=, /=, %=

```
int count = 10 ;
count += 1;
count *= 2 ;
count %= 5 ;
cout << count ;
```

Arithmetic Operators

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

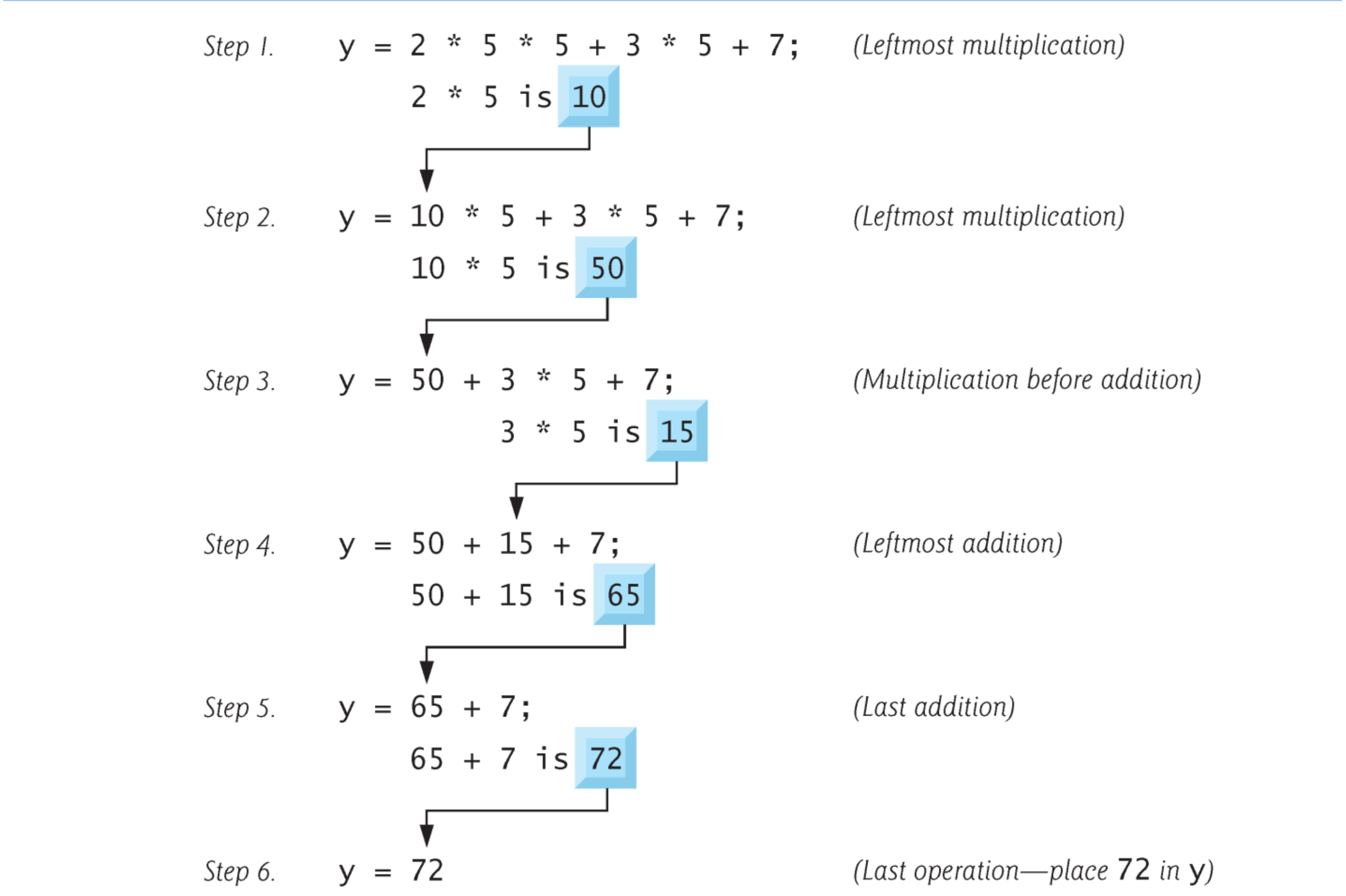
Relational Operators

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Compound Assignment

expression	equivalent to...
y += x;	y = y + x;
x -= 5;	x = x - 5;
x /= y;	x = x / y;
price *= units + 1;	price = price * (units+1);

Source: http://www.cplusplus.com/doc/tutorial/operators/



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

`2 * 5 is` $\boxed{10}$

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

`10 * 5 is` $\boxed{50}$

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

`3 * 5 is` $\boxed{15}$

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

`50 + 15 is` $\boxed{65}$

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

`65 + 7 is` $\boxed{72}$

Step 6. `y = 72` *(Last operation—place* `72` *in* `y`*)*

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

Arithmetic Expression

- Straight-line form
  - a + b
- Parenthesis for group sub-expressions
- Rules of Operator Precedence
  - Same as in algebra: within parenthesis first, multiplication and division are next. Addition and subtraction are applied next.
  - Associativity: left-to-right for multiplication, division, addition and subtraction

C++ Data Formatting

- <iomanip> header file for IO manipulators

```
#include <iomanip>
…
double pi = 3.14159 ;
cout << fixed ;
cout << "PI: " << setprecision(2) << pi << endl;
```

Algebraic relational or equality operator	C++ relational or equality operator	Sample C++ condition	Meaning of C++ condition
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

Fig. 2.12 | Relational and equality operators.

Logical Operators: && and ||

a	b	a && b	a || b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Source: http://www.cplusplus.com/doc/tutorial/operators/

Compound Statement

- Also referred as a “block” or “group”
- Multiple statements enclosed in a pair of curly braces

```
{
   int num = 1;
   cout << num;
}
```

Decision Making/Selection Statement

- “if-else” statement
  - The condition expression is evaluated. If it is true (non-zero), the statement will be executed. If false, it will be ignored (e.g. skipped) and the “else” part will be executed if present.

- Note: putting the “;” after the condition will terminate the “if” statement and makes the body of the “if” statement empty.

```
if (unitPrice > 100)
    ;   // ignore

if (count <= 10)
   cout << "Not on sale" << endl;
else
   cout << "50% discount" << endl;
```

Nested “if” statement

- “if-else if-else” statement

```
if (count <= 10)
   cout << "Not on sale" << endl;
else if (count <= 100)
   cout << "20% discount" << endl;
 else
    cout << "50% discount" << endl;
```

Dangling-**else** statement

- “if-if-else” statement

```
if (count <= 10)
   if (stockQuantity < 100)
         cout << "Not on sale" << endl ;
else
   cout << "10% discount";
```

Increment and Decrement operators

- ++
- --

```
if (hasExtraCredit == true)
   points++;
else if (hasNoOutput == true)
   points--;
```

Operato r	Called	Sample expression	Explanation
`++`	preincrement	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	postincrement	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	predecrement	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	postdecrement	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.

Fig. 4.18 | Increment and decrement operators.

Conditional Ternary Operator

- ?
  - Syntax: condition ? result-for-true : result-for-false ;

```
int quantity, fullPrice=100, discountPrice = 80;
cin >> quantity;
(quantity > 100)? discountPrice : fullPrice ;
```