Key point: *Operator precedence and associativity determine the order in which operators are evaluated.*

Operator precedence: The precedence of an operator determines the order in which it is executed. In an infix expression an operand will (often) have an operator on either side of it. Operand precedence determines which operator is evaluated first.

Example: 3 + 4 * 5
When the expression 3 + 4 * 5 is evaluated, is the answer 35 or is the answer 23. The answer depends on which operator, addition or multiplication, is evaluated first. If multiplication is performed first, the answer is 4 * 5 = 20 + 3 = 23. I addition is performed first, the answer is 3 + 4 = 7 * 5 = 35.

Table 3.8 Operator Precedence Chart

Precedence	*Operator*
High	**var++** and **var--** (Postfix)
	+, **-** (Unary plus and minus), **++var** and **--var** (prefix)
	(type) (Casting)
	! (Not)
	*****, **/**, **%** (Multiplication, division, and remainder)
	+, **-** (Binary addition and subtraction)
	<, **<=**, **>**, **>=** (Relational)
	==, **!=** (Equality)
	^ (Exclusive OR)
	&& (Logical AND)
	|| (Logical OR)
Low	**=**, **+=**, **-=**, ***=**, **/=** (Assignment operators)

Associativity: Associativity determines the order of evaluation when operators have the same precedence.

Consider: a – b + c – d
Addition and subtraction have the same precedence and, like other binary operators, they are ***left*** associative. Hence,

a – b + c - d = (((a – b) + c) – d)

Assignment operators are right associative making
a = b += c = d equivalent to a = (b += (c =d))