15CSE102
Computer Programming

$$A + B = C$$

Arithmetic Operators

Operator	Meaning
*	Multiply
/	Divide
+	Add
-	Subtract
%	Modulus (return the remainder after division)

Credits: Practical C Programming

Algebraic vs C Expressions

Algebra: $m = \frac{a + b + c + d + e}{5}$

C:
```
m = ( a + b + c + d + e ) / 5;
```

Algebra: $y = mx + b$

C:
```
y = m * x + b;
```

Algebra: $z = pr\%q + w/x - y$

C:
```
z = p * r % q + w / x - y;
```

Credits: How to Program C

Evaluating Expressions

Are these two expressions the same?

```
m = a + b + c + d + e / 5;
```

```
m = (a + b + c + d + e) / 5;
```

Evaluating Expressions

Are these two expressions the same?

```
m = a + b + c + d + e / 5;
```

```
m = (a + b + c + d + e) / 5;
```

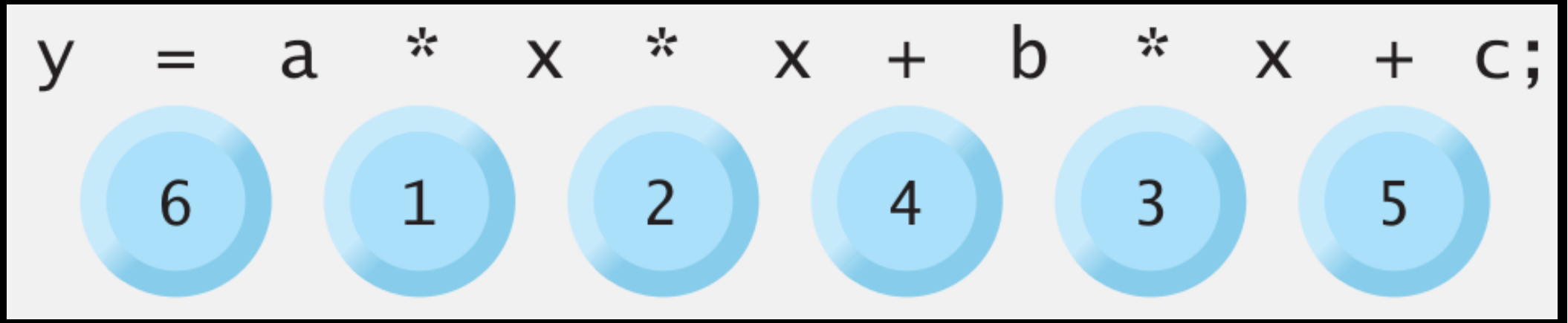
Depends on the order of evaluation!!

Evaluating Expressions

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the *innermost* pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they're evaluated left to right.
*	Multiplication	Evaluated second. If there are several, they're evaluated left to right.
/	Division	
%	Remainder	
+	Addition	Evaluated third. If there are several, they're evaluated left to right.
-	Subtraction	
=	Assignment	Evaluated last.

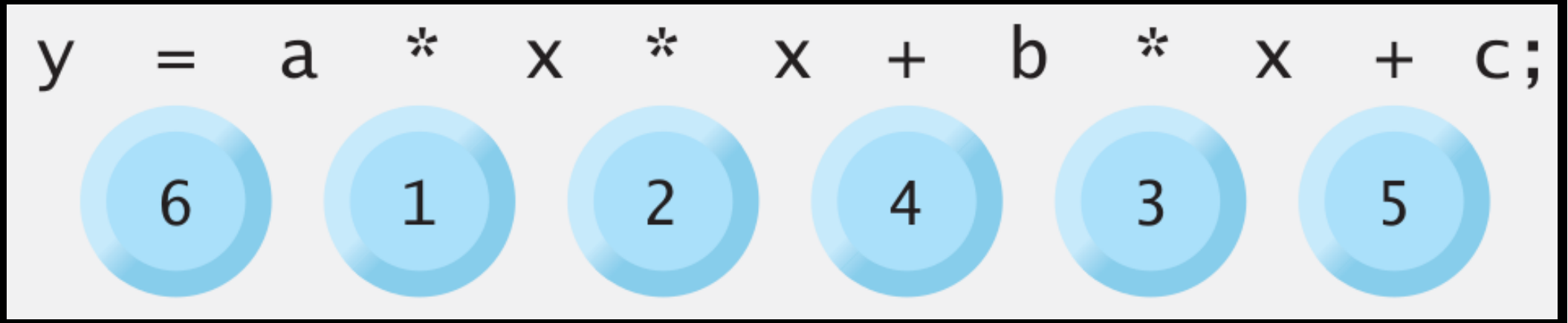
Credits: How to Program C

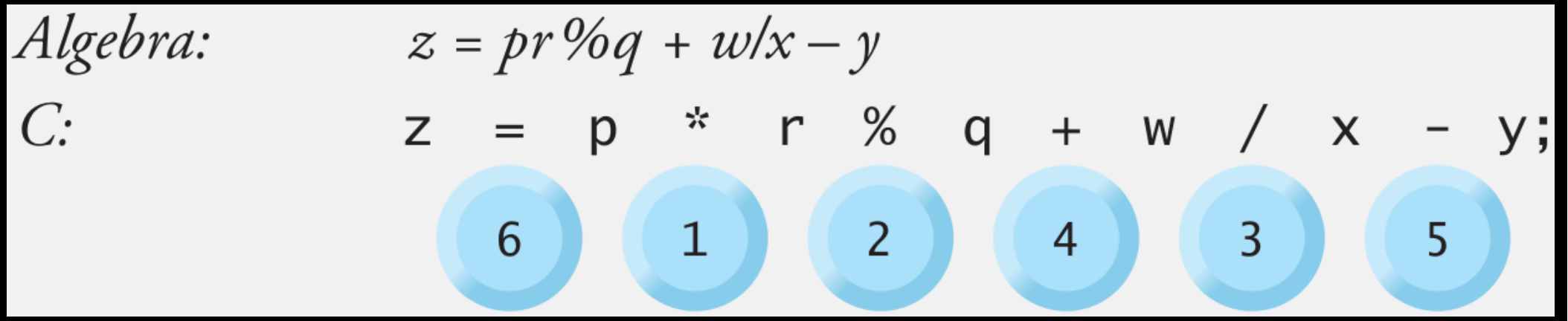
Evaluating Expressions



Credits: How to Program C

Evaluating Expressions





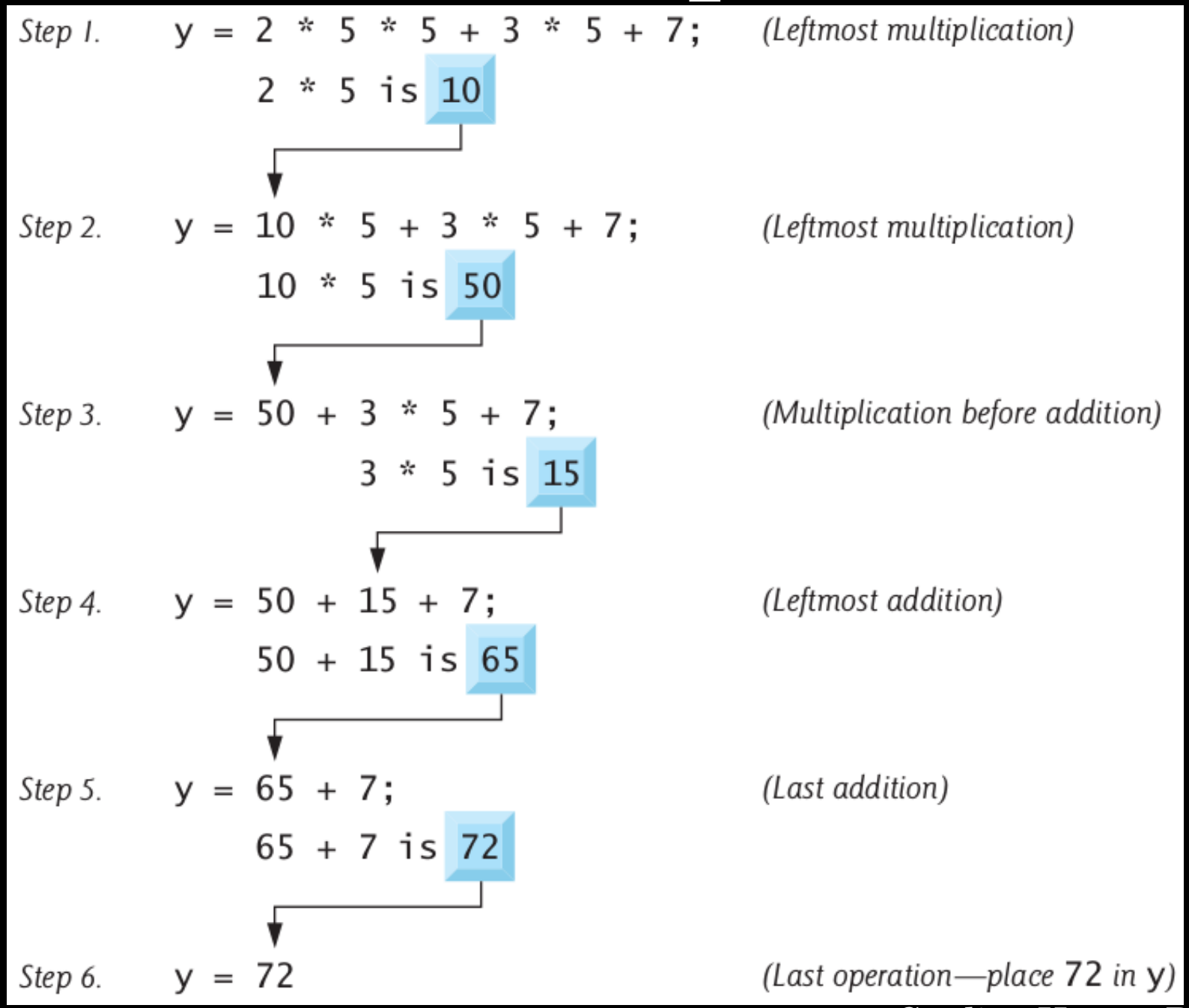
Credits: How to Program C

Evaluate Expression

Give the value of y

```
y = 2 * 5 * 5 + 3 * 5 + 7;
```

Evaluate Expression



Credits: How to Program C

Evaluate Expression

If you think order of evaluation is hard to remember

```
y = 2 * 5 * 5 + 3 * 5 + 7;
```

Evaluate Expression

If you think order of evaluation is hard to remember

```
y = 2 * 5 * 5 + 3 * 5 + 7;
```

parantheses comes very handy!!!!

```
y =(2 * 5 * 5) + (3 * 5) + 7;
```

Be the Compiler

```
#include <stdio.h>
int main()
{
  float c;
  c = 5/9;
  printf("c = %f", c);
  return 0;
}
```

What do you think is the output of this code?

Credits: www.tech-recipes.com

Be the Compiler

```
#include <stdio.h>
int main()
{
  float c;
  c = 5/9;
  printf("c = %f", c);
  return 0;
}
```

The following will be the output

c = 0.000000

Credits: www.tech-recipes.com

Float vs Integer

Expression	Result	Result type
1 + 2	3	Integer
1.0 + 2.0	3.0	Floating Point
19 / 10	1	Integer
19.0 / 10.0	1.9	Floating Point

Credits: Practical C Programming

Float vs Integer Divide

```
#include <stdio.h>
int main()
{
  float c;
  c = 5.0/9;
  printf("c = %f", c);
  return 0;
}
```

The following will be the output

c = 0.555556

Credits: www.tech-recipes.com

Remember

1. If an arithmetic operator has integer operands then integer operation is Performed (resulting in integer type!)

2. If an arithmetic operator has one floating point operator and one integer operator, the integer will be converted to float before the operation is done

Credits: www.tech-recipes.com

Relational Operators

Algebraic equality or relational operator	C equality or relational operator	Example of C condition
Equality operators		
=	==	x == y
≠	!=	x != y
Relational operators		
>	>	x > y
<	<	x < y
≥	>=	x >= y
≤	<=	x <= y

Credits: How to Program C

Precedence & Associativity

Operators	Associativity
()	left to right
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
=	right to left

Credits: How to Program C

Logical Operators

Logical Operators		
Operator	**Description**	**Example**
&&	AND	x=6 y=3 x<10 && y>1 Return True
||	OR	x=6 y=3 x==5 || y==5 Return False
!	NOT	x=6 y=3 !(x==y) Return True

Credits: studytipsandtricks.blogspot.in

Credits: www.programiz.com

Bitwise AND &

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bit Operation of 12 and 25
  00001100
& 00011001
  ----------
  00001000  = 8 (In decimal)
```

Credits: www.programiz.com

Bitwise OR |

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25
   00001100
|  00011001
   ----------
   00011101  = 29 (In decimal)
```

Credits: www.programiz.com

Bitwise XOR ^

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25
   00001100
|  00011001
   ----------
   00010101  = 21 (In decimal)
```

Credits: www.programiz.com

Bitwise Complement ~

```
35 = 00100011 (In Binary)


Bitwise complement Operation of 35
~ 00100011
  ________
  11011100  = 220 (In decimal)
```

Credits: www.programiz.com

Bitwise Complement ~

```
35 = 00100011 (In Binary)

Bitwise complement Operation of 35
~ 00100011
  ___________
  11011100  = 220 (In decimal)
```

But the bitwise complement of 35 is −36 how?

Bitwise Complement ~

```
35 = 00100011 (In Binary)

Bitwise complement Operation of 35
~ 00100011
  ___________
  11011100  = 220 (In decimal)
```

Negative numbers are stored as two's complement of positive counterpart.
220 is two's complement of −36!!

Assignment Operators

Assignment operator	Sample expression	Explanation
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`		
`+=`	`c += 7`	`c = c + 7`
`-=`	`d -= 4`	`d = d - 4`
`*=`	`e *= 5`	`e = e * 5`
`/=`	`f /= 3`	`f = f / 3`
`%=`	`g %= 9`	`g = g % 9`

Credits: How to Program C

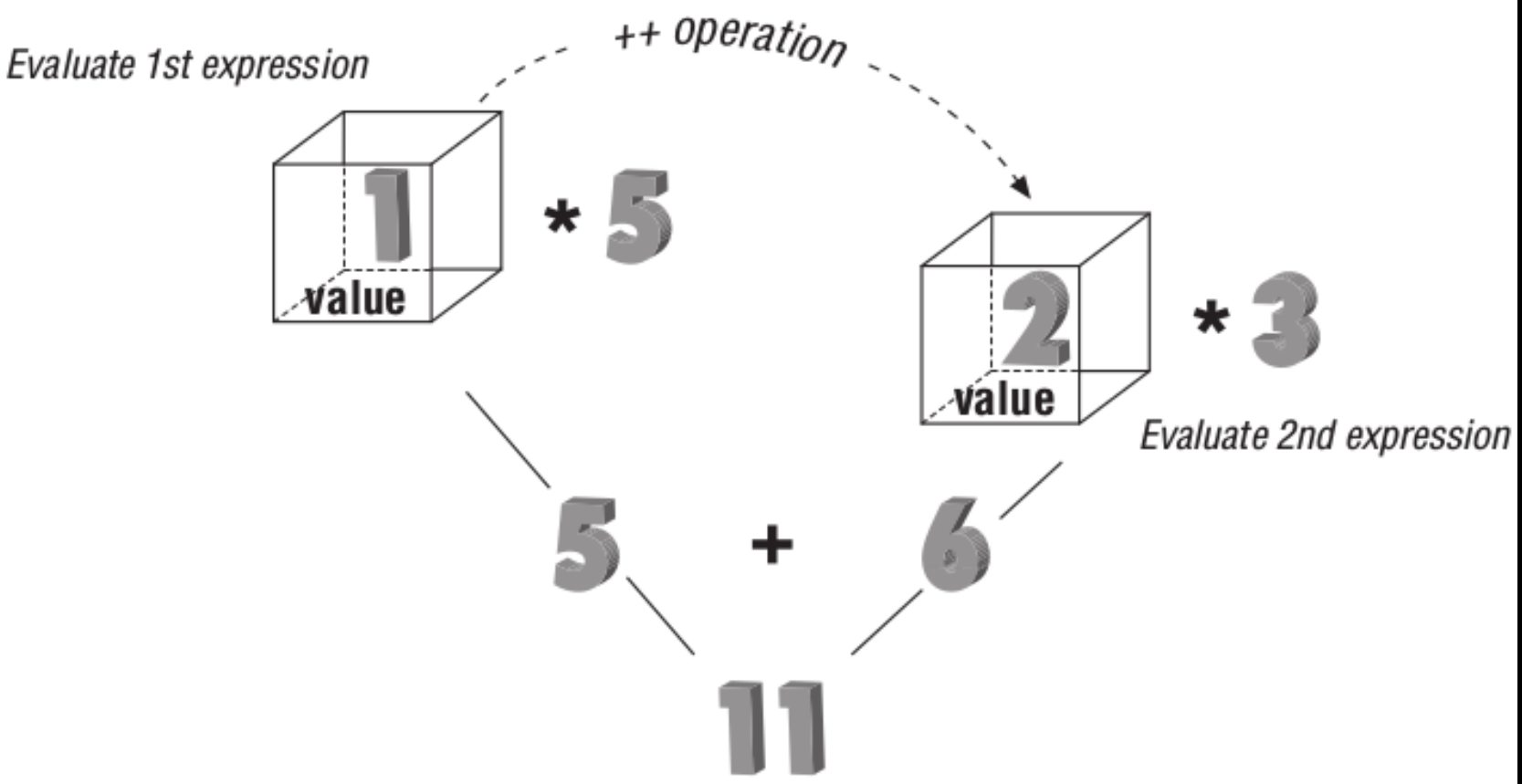
Unary Operators

Operator	Sample expression	Explanation
++	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Credits: How to Program C

Side-Effect Problems

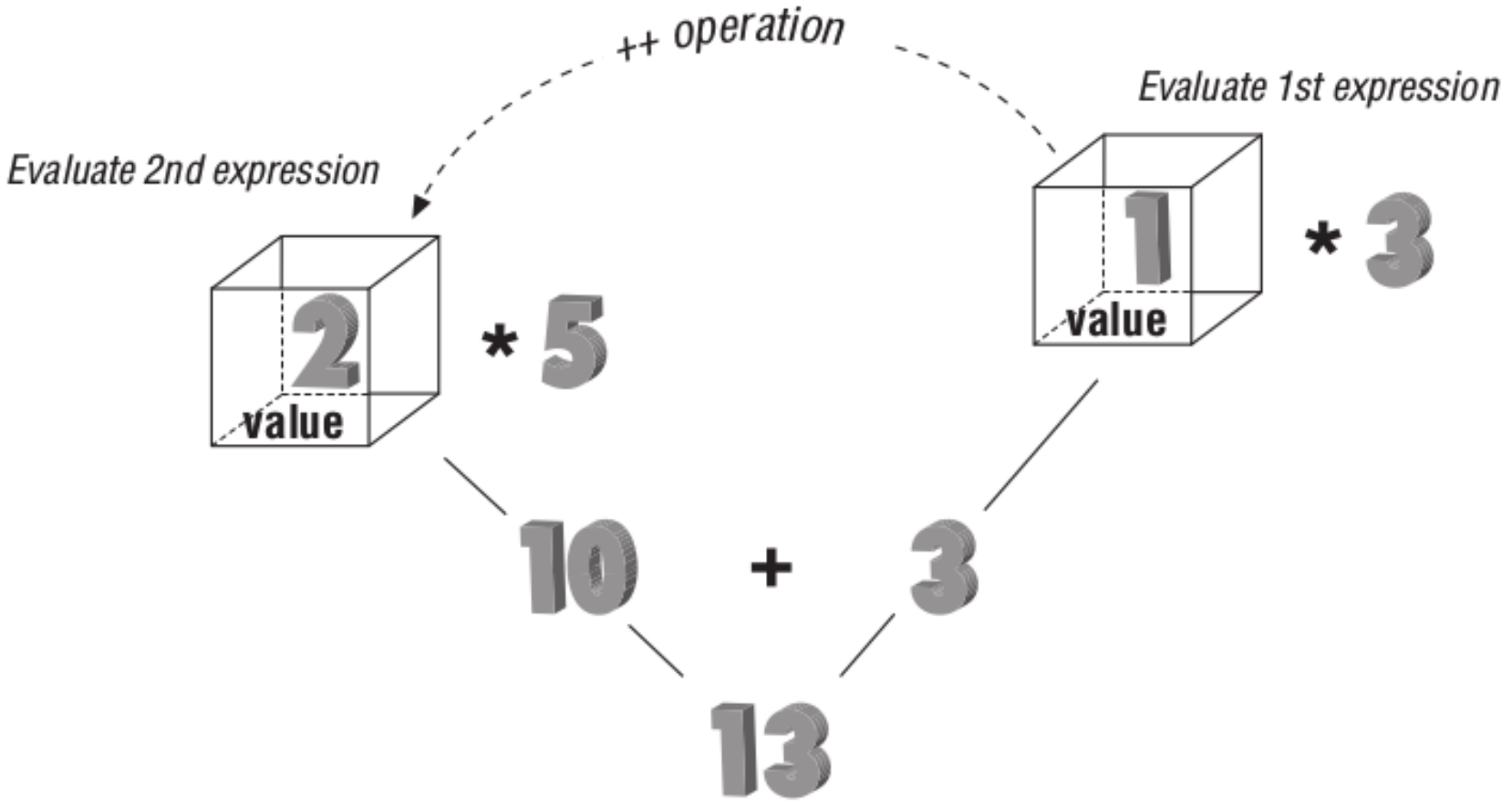
```
result  =  (value++ * 5)  +  (value++ * 3);
```



Credits: Practical C Programming

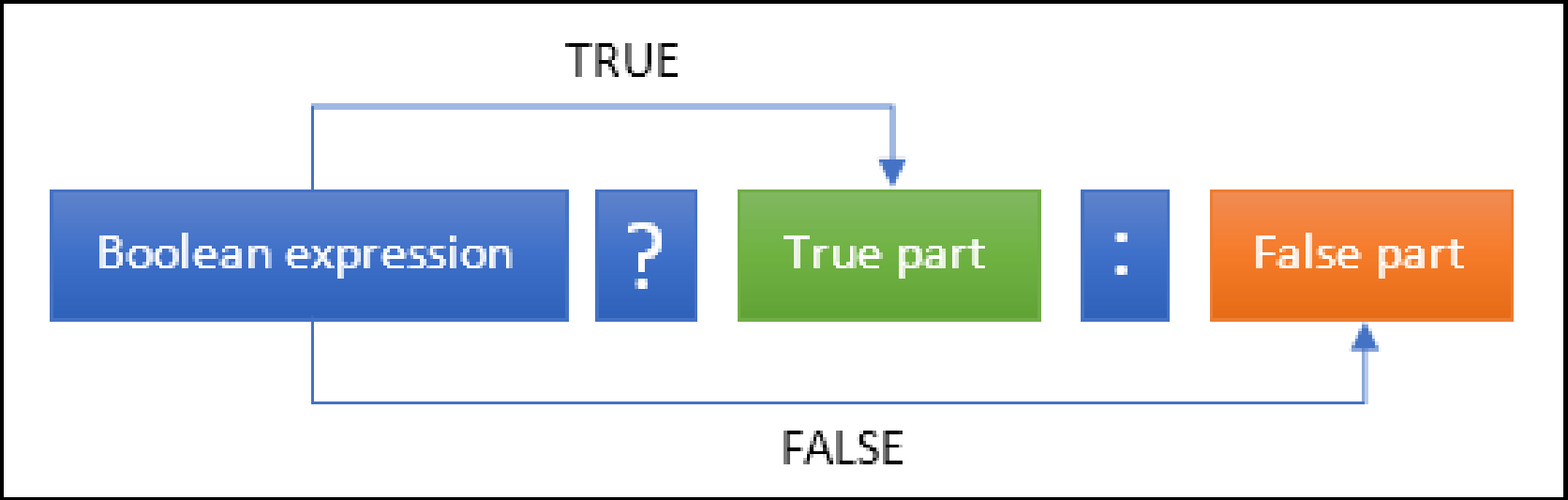
Side-Effect Problems

```
result  =  (value++ * 5)  +  (value++ * 3);
```



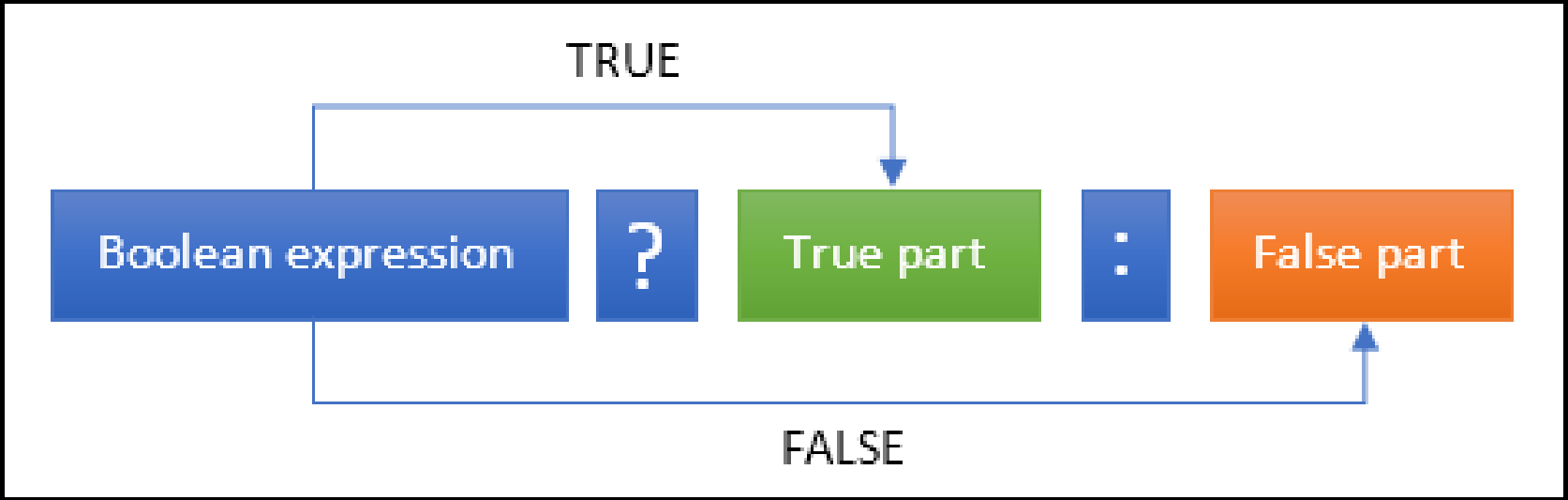
Credits: Practical C Programming

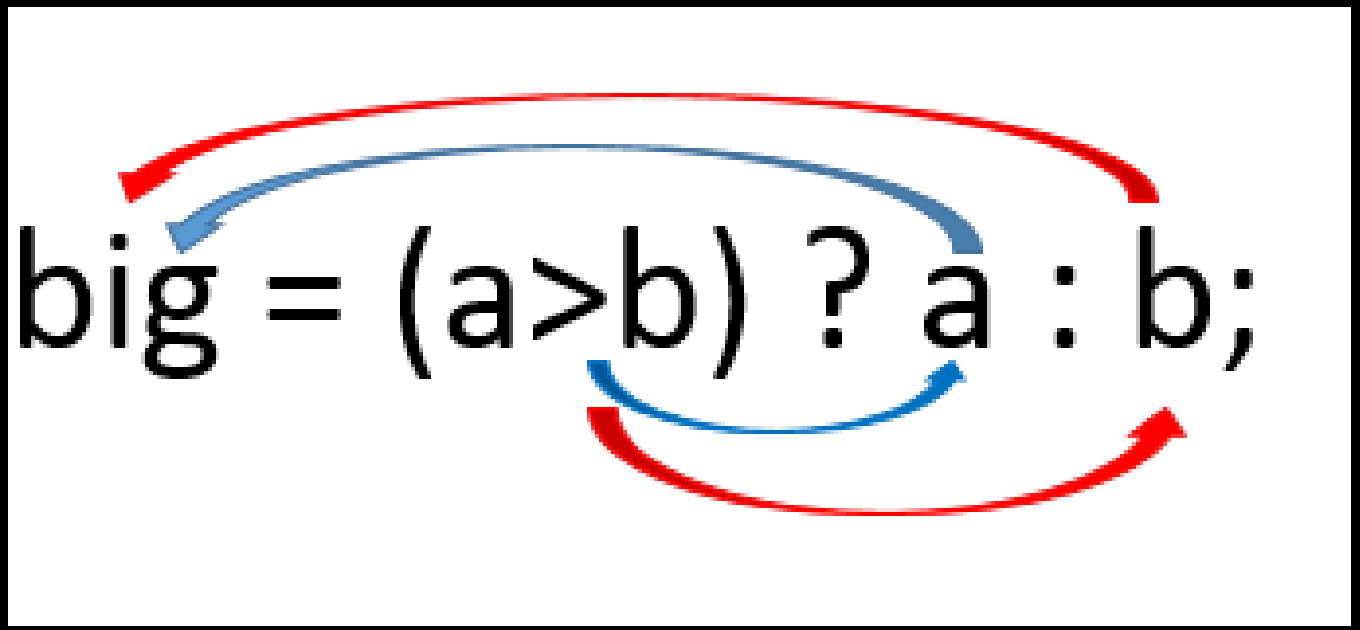
Conditional Operator



Credits: codeforwin.in &

Conditional Operator



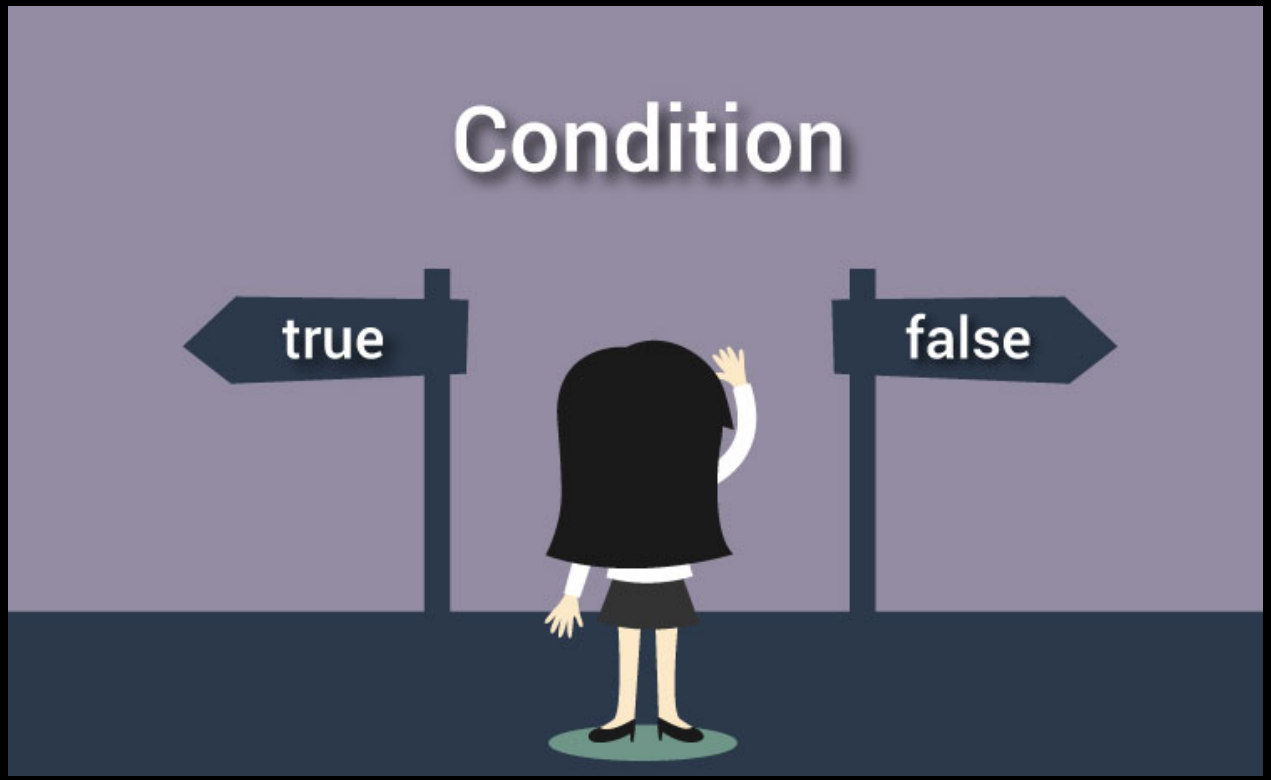


Credits: codeforwin.in & infoletcodes.blogspot.in

Big Picture

Operators	Associativity	Type
`++` *(postfix)* `--` *(postfix)*	right to left	postfix
`+` `-` (*type*) `++` *(prefix)* `--` *(prefix)*	right to left	unary
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment

CSE102
Computer Programming
(Next Topic)



Credits: programiz.com