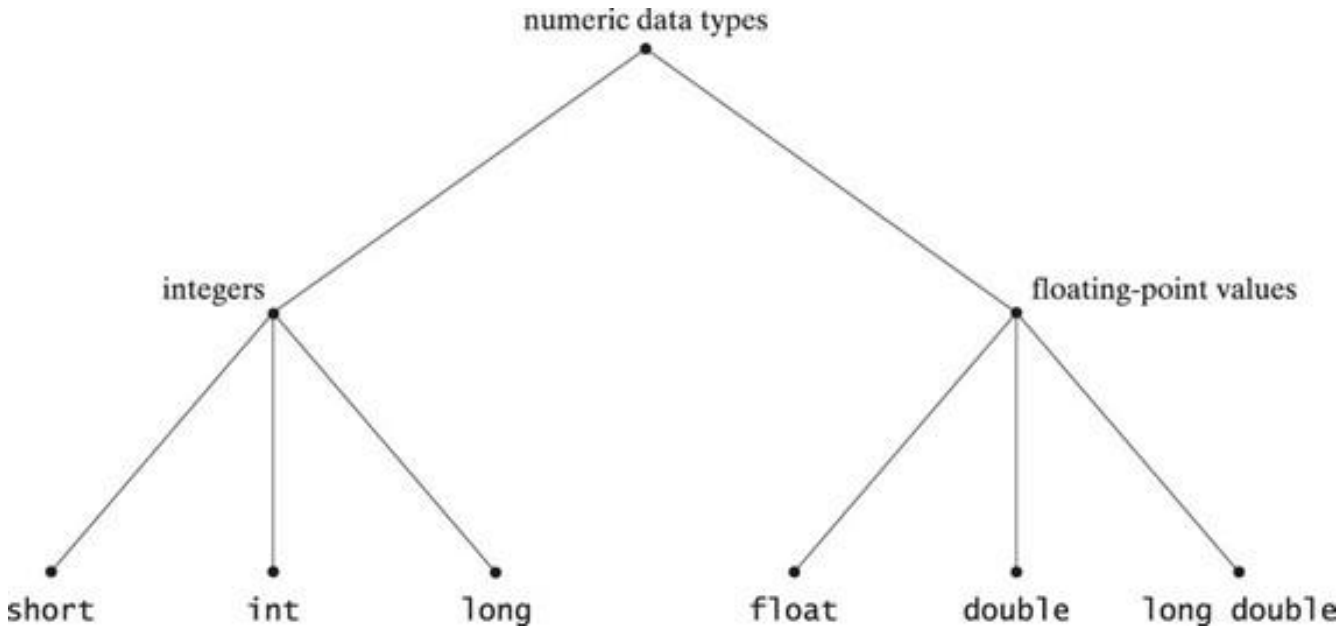
Chapter 4: Basic C Operators

- In this chapter, you will learn about:
  - Arithmetic operators
    - Unary operators
    - Binary operators
  - Assignment operators
  - Equalities and relational operators
  - Logical operators
  - Conditional operator

Arithmetic Operators

- There are 2 types of arithmetic operators in C:
  - unary operators
    - operators that require only one operand.
  - binary operators.
    - operators that require two operands.

Numeric Data Types



Data Type Limits

Integers	
`short`	Maximum = 32,767
`int`	Maximum = 2,147,483,647
`long`	Maximum = 2,147,483,647
Floating Point	
`float`	6 digits of precision Maximum exponent 38 Maximum value 3.402823e+38
`double`	15 digits of precision Maximum exponent 308 Maximum value 1.797693e+308
`long double`	15 digits of precision Maximum exponent 308 Maximum value 1.797693e+308

*Microsoft Visual C++ 6.0 compiler.

Unary Operator

C operation	Operator	Example	Explanation
Positive	+	a = +3	
Negative	-	b = -4	
Increment	++	i++	Equivalent to i = i + 1
Decrement	- -	i - -	Equivalent to i = i - 1

PRE / POST Increment

- It is also possible to use ++i and --i instead of i++ and i--
- However, the two forms have a slightly yet important difference.
- Consider this example:

```
int a = 9;
printf("%d\n", a++);
printf("%d", a);
```

- The output would be:

  9

  10

PRE / POST Increment cont…

- But if we have:

```
int a = 9;
printf("%d\n", ++a);
printf("%d", a);
```

- The output would be:

  10

  10

- a++ would return the current value of *a* and then increment the value of *a*
- ++a on the other hand increment the value of *a* before returning the value

The following table illustrates the difference between the prefix and postfix modes of the increment and decrement operator.

Assuming we have the following variables declaration.

```
int R = 10, count=10;
```

++ Or -- Statement	Equivalent Statements	R	Count
R = count++;	R = count; count = count + 1	10	11
R = ++count;	count = count + 1; R = count;	11	11
R = count --;	R = count; count = count – 1;	10	9
R = --count;	Count = count – 1; R = count;	9	9

Binary Operators

C operation	Operator	Example
Addition	+	b = a + 3
Subtraction	-	b = a - 4
Multiplication	*	b = a * 3
Division	/	b = a / c
Modulus	%	b = a % c

- The division of variables of type integer will always produce a variable of type integer as the result.
- You could **only use** modulus (%) operation on integer variables.

- The division of variables of type integer will always produce a variable of type integer as the result.

- Example

```
int a = 7, b;
b = a/2;
printf("%d\n", b);
```

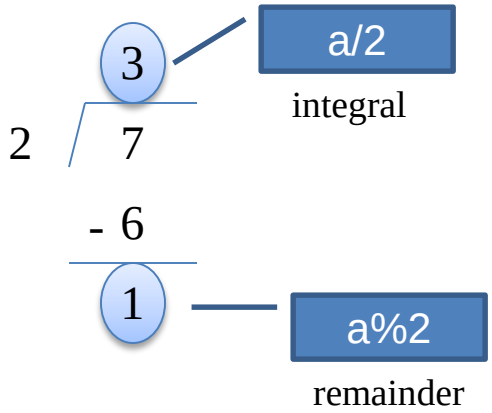
Since b is declared as an integer, the result of a/2 is 3, not 3.5

```
3
Press any key to continue
```

- You could only use modulus (%) operation on integer variables/integer division.
- Example

```
int a = 7, b, c;
b = a%2;
c = a/2;
printf("b = %d\n", b);
printf("c = %d\n", c);
```

Modulus will result in the remainder of a/2.




```
b = 1
c = 3
Press any key to continue
```

Numeric Conversion

- If a value is assigned to a variable that has a different data type, a *conversion* will occur during execution.

e.g.
```
int a;
....
a = 12.8;
```

- Conversion from low order to high order is safe, but not the other way round.

High	long double
	double
	float
	long integer
	integer
Low	short integer

Overflow and Underflow

- Occurs when result of an arithmetic operation exceeds (either too large or too small) allowable range.

- Overflow example:

```
x = 2.5e30;
y = 1.0e30
z = x*y
```

- Underflow example:

```
x = 2.5e-30;
y = 1.0e30
z = x/y
```

Assignment Operators

- Assignment operators are used to combine the '=' operator with one of the binary arithmetic operators
- In the following example, **all operations starting from c = 9**

Operator	Example	Equivalent Statement	Results
+=	c += 7	c = c + 7	c = 16
-=	c -= 8	c = c – 8	c = 1
*=	c *= 10	c = c * 10	c = 90
/=	c /= 5	c = c / 5	c = 1
%=	c %= 5	c = c % 5	c = 4

Precedence Rules

- Precedence rules come into play when there is a mixed of arithmetic operators in one statement. For example: x = 3 * a - ++b%3;
- The rules specify which of the operators will be evaluated first.

Precedence Level	**Operator**	**Associativity**
1 (highest)	()	left to right
2	unary	right to left
3	* / %	left to right
4	+ -	left to right
5 (lowest)	= += -= *= /= %=	right to left

Precedence Rules cont…

- For example: x = 3 * a - ++b % 3;

  how would this statement be evaluated? What is the value for X, given the following values: a = 2, b = 4?

x = 3 * a - ++b % 3;
x = 3 * a - 5 % 3;
x = 3 * a - 5 % 3;
x = 6 - 5 % 3;
x = 6 - 2
x = 4

- If we intend to have the statement

  x = 3 * a - ++b % 3;

  evaluated differently from the way specified by the precedence rules, we need to specify it using parentheses ( )
- Consider having the following statement:

  x = 3 * ((a - ++b)%3);
- In this case, the expression inside a parentheses will be evaluated first.
- The inner parentheses will be evaluated earlier compared to the outer parentheses.

- x = 3 * ((a - ++b)%3);

  how would this statement be evaluated? What is the value for X, given the following values: a = 2, b = 4?

x = 3 * ((a - ++b)%3);

x = 3 * ((a - 5)%3);

x = 3 * ((-3)%3);

x = 3 * 0;

x = 0;

- Given the following expression, what will be the value of x, a and b once the expression be evaluated? Given the following values: *a = 2, b = 4?*

```
x = 3 * ++a – b--%3;
x = 3 * ++a – b--%3;
x = 3 * 3 – b--%3;
x = 3 * 3 – b--%3;
x =     9 – b--%3;
x =     9 – b--%3;
x =     8;
```

b = b -1;
b = 3;

a 2 b 4

a 3 b 4

a 3 b 3

Mathematical Functions

- Engineering problem solving usually requires the use of formula beyond addition, subtraction, multiplication and division.
- Many expressions require the use of exponentiation, logarithms, exponentials and trigonometric functions.
- Use the following preprocessor directive to use the elementary and trigonometric math functions in C.

```
#include <math.h>
```

- Available functions include fabs(x), sqrt(x), pow(x,y), ceil(x), floor(x), exp(x), log(x) and log10(x), sin(x), cos(x) and tan(x).

Equality and Relational Operators

- Equality Operators:

Operator	Example	Meaning
==	x == y	x is equal to y
!=	x != y	x is not equal to y

- Relational Operators:

Operator	Example	Meaning
>	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

Logical Operators

- Logical operators are useful when we want to test multiple conditions.
- There are 3 types of logical operators and they work the same way as the boolean AND, OR and NOT operators.
- && - Logical AND
  - All the conditions must be true for the whole expression to be true.
  - Example: `if (a == 10 && b == 9 && d == 1)`

    means that the *if* statement is only true when *a* == 10 **and** *b* == 9 **and** *d* == 1.

Logical Operators cont…

- || - Logical OR
  - The truth of one condition is enough to make the whole expression true.
  - Example: `if (a == 10 || b == 9 || d == 1)`

    means the *if* statement is true when **either one** of *a*, *b* or *d* has the right value.
- ! - Logical NOT (also called logical negation)
  - Reverse the meaning of a condition
  - Example: `if (!(points > 90))`

    means if points not bigger than 90.

Conditional Operator

- The conditional operator (?:) is used to simplify an if/else statement.
- Syntax:

  Condition ? Expression1 : Expression2
- The statement above is equivalent to:

```
if (Condition)
    Expression1
else
    Expression2
```

Conditional Operator cont…

- Example 1:

if/else statement:

```
 if (total > 60)
      grade = 'P'
else
      grade = 'F';
```

conditional statement:

```
(total > 60) ? grade = 'P': grade = 'F';
```

OR

```
grade =( total > 60) ? 'P': 'F';
```

Conditional Operator cont…

- Example 2:

  <u>if/else statement:</u>

```
if (total > 60)
        printf("Passed!!\n");
else
        printf("Failed!!\n");
```

  <u>Conditional Statement:</u>

```
printf("%s!!\n", total > 60? "Passed": "Failed");
```

Practice 1

- Write a program to compute the area of a rectangle with sides *a* and *b*. The values of *a* and *b* are entered from standard input.

- Write a program to compute the radius of a circle having the same area as that of a rhombus with diagonals d_1 and d_2. The values of d_1 and d_2 are entered from standard input.

Practice 2

- Write the program for height estimation problem solved in Chapter 2 earlier.

SUMMARY

- This chapter exposed you the operators used in C
  - Arithmetic operators
  - Assignment operators
  - Equalities and relational operators
  - Logical operators
  - Conditional operator
- Precedence levels come into play when there is a mixed of arithmetic operators in one statement.
- Pre/post fix - effects the result of statement