C PROGRAMMING FOR ENGINEERS

Arithmetic Operations

Assignment Statements

- General form:
  variable = expression;

Can be any constant, variable, or algebraic expression.

= sign is the assignment operator.

Variable identifies the memory location.

Initializing a variable when declared:

```
int x1 = 5;
```

Declaring a variable and then assigning a value:

```
int x1;
x1 = 5;
```

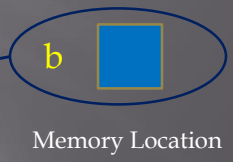
Assigning the same value to more than one variable:

```
int x1 = x2 = x3 = 5;
```

Examples (cont.)

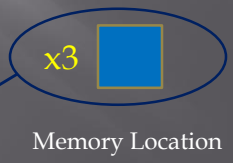
- Assigning values to variables of different data type:
- Example 1:

```
int b;
b = 15.5;
```



- Example 2:

```
int x1;
double x2, x3;
x1 = 9.8;
x2 = 1.5;
x3 = x1 + x2;
```



Arithmetic Operators

- Binary Operators:
  - Performs an operation using two values.
  - Types:

Addition	+
Subtract	-
Multiplication	*
Division	/
Modulus	%

The modulus operator gives the remainder in a division between two integers, such as:

5 % 2 = 1
6 % 3 = 0
2 % 5 = 2

- Unary Operators:
  - Performs an operation using one value:
  - Types:

Plus sign	+
Minus sign	-

Example: y = -(x+7);

Examples of Binary Operations

- Example 1:

```
x = 1;
x = x + 1;
```

This is not a valid statement in algebra.
It is read as x is assigned the value of x+1.

- Example 2:

```
y = 2;
y = y * x;
```

Assuming that x is the value from the above example, what would be the value of y?

y 

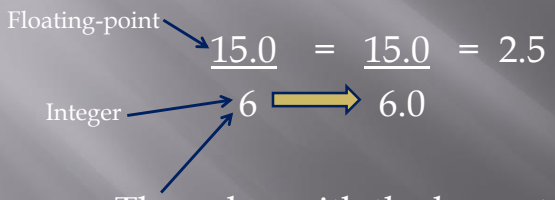
Integer & Floating-point Operations

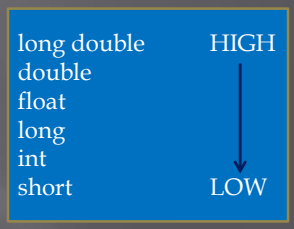
- Examples:

Integers:	Floating-point:
15+2 = 17	1.2+0.8 = 2.0
2*6 = 12	3.5*3.0 = 10.5
7–10 = -3	1.5-2.0 = -0.5
6/2 = 3	6.0/2.0 = 3.0
3/2 =	3.0/2.0 =
3/6 =	3.0/6.0 =

Mixed Mode Operations

- Arithmetic operations that are performed between two values of different types.
- Example:





The value with the lower type is converted to the higher type before the computation is performed.

A Programming Example

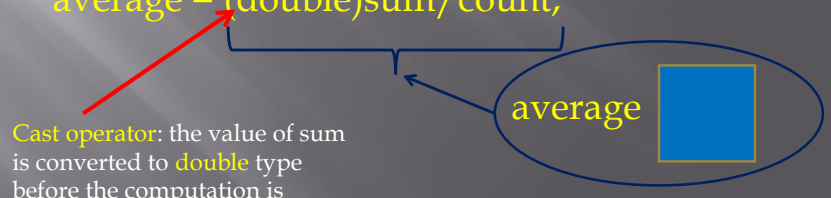
```
int sum, count;
double average;
sum = 18;
count = 5;
average = sum/count;
```

Memory Location

Cast Operator

- Program Example (floating-point):

```
int sum, count;
double average;
sum = 18;
count = 5;
average = (double)sum/count;
```



Cast operator: the value of sum is converted to double type before the computation is performed.

Memory Location

Cast Operator (cont.)

- Program Example (integer):

```
int b = 10;
double a, c=200.65;
a = (int)c/b;
```

Memory Location

Increment & Decrement Operations

- Considered unary operations.
- Two types: Prefix and Postfix

Prefix Examples:

++y --y

Postfix Examples:

y++ y--

Equivalent Evaluation

- Example 1:

y++; y = y+1;

- Example 2:

x--; x = x-1;

Prefix and Postfix Evaluation Order

- Example 1 (prefix):

```
x = 5, y = 3;
w = ++x – y;
```

Evaluation Order

```
x = 5, y = 3;
x = x+1;
w = x-y;
```

- Example 2 (postfix):

```
x = 5, y = 3;
w = x++ - y;
```

Evaluation Order

```
x = 5, y = 3;
w = x-y;
x = x+1;
```

Abbreviated Assignment Operations

- General form:

  variable operator = expression

- Examples:

Normal Code	Abbreviated Code
x = x+3;	x += 3;
d = d/4.5;	d /= 4.5;
r = r*0.5;	r *= 0.5;

Hierarchy (priority) of Arithmetic Operators

- From highest to lowest:

Operator Type	Evaluation Order
Parentheses ( )	inner most first
Unary operators + - ++ --	right to left
Binary operators * / %	left to right
Binary operators + -	left to right
Assignment Operators = += -= *= /= %=	right to left

- Always use parentheses for complex expressions.
- Split long equations into multiple statements; do not attempt to condense them into a single statement.

Order of Evaluation of an Arithmetic Expression

- Algebraic form:

  $d = 90.0+2.125t^2 - 0.00125t^4$

- C programming form:

  d = 90.0 + 2.125*(t*t) – 0.00125*(t*t*t*t);