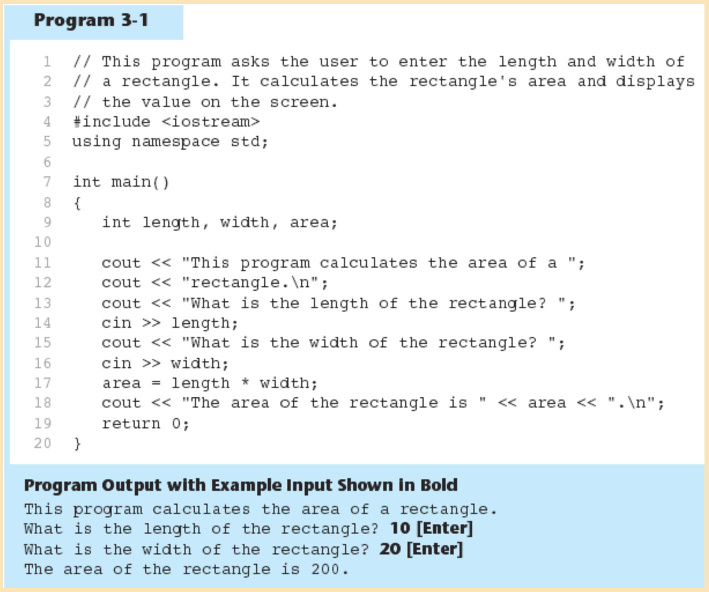
Arithmetic Expressions

The `cin` Object

- Standard input object
- Like **`cout`**, requires **`iostream`** file
- Used to read input from keyboard
- Information retrieved from **`cin`** with **>>**
- Input is stored in one or more variables



Program 3-1

```cpp
// This program asks the user to enter the length and width of
// a rectangle. It calculates the rectangle's area and displays
// the value on the screen.
#include <iostream>
using namespace std;

int main()
{
   int length, width, area;

   cout << "This program calculates the area of a ";
   cout << "rectangle.\n";
   cout << "What is the length of the rectangle? ";
   cin >> length;
   cout << "What is the width of the rectangle? ";
   cin >> width;
   area = length * width;
   cout << "The area of the rectangle is " << area << ".\n";
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
What is the length of the rectangle? 10 [Enter]
What is the width of the rectangle? 20 [Enter]
The area of the rectangle is 200.
```

The `cin` Object

- **`cin`** converts data to the type that matches the variable:

```cpp
int height;
cout << "How tall is the room? ";
cin >> height;
```

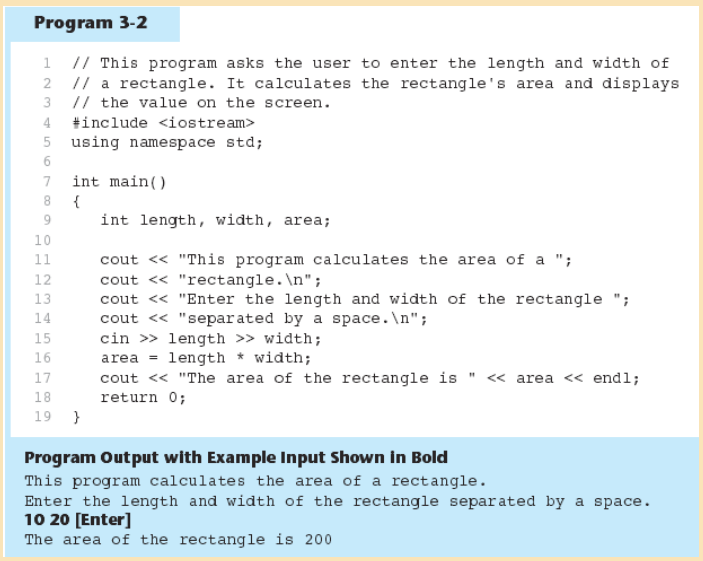
Displaying a Prompt

- A prompt is a message that instructs the user to enter data.
- You should always use **`cout`** to display a prompt before each cin statement.

```
cout << "How tall is the room? ";
cin >> height;
```

The `cin` Object

- Can be used to input more than one value:
  ```
  cin >> height >> width;
  ```
- Multiple values from keyboard must be separated by spaces
- Order is important: first value entered goes to first variable, etc.



Program 3-2

```
// This program asks the user to enter the length and width of
// a rectangle. It calculates the rectangle's area and displays
// the value on the screen.
#include <iostream>
using namespace std;

int main()
{
   int length, width, area;

   cout << "This program calculates the area of a ";
   cout << "rectangle.\n";
   cout << "Enter the length and width of the rectangle ";
   cout << "separated by a space.\n";
   cin >> length >> width;
   area = length * width;
   cout << "The area of the rectangle is " << area << endl;
   return 0;
}
```

Program Output with Example Input Shown in Bold

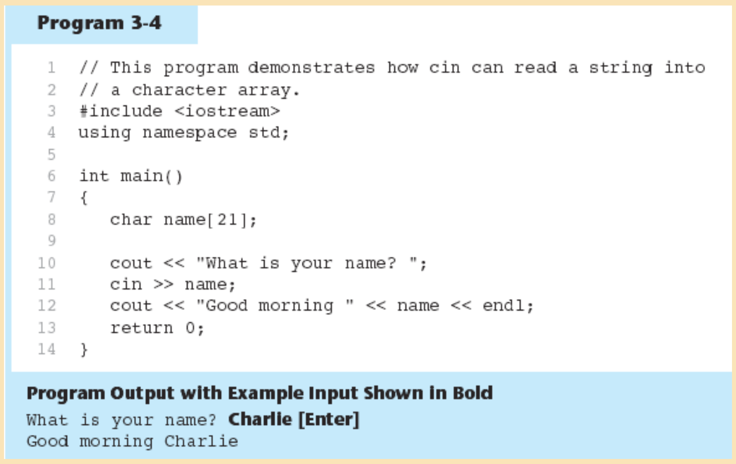
```
This program calculates the area of a rectangle.
Enter the length and width of the rectangle separated by a space.
10 20 [Enter]
The area of the rectangle is 200
```

Reading Strings with `cin`

- Can be used to read in a string
- Must first declare an array to hold characters in string:

  `char myName[21];`
- `myName` is a name of an array, `21` is the number of characters that can be stored (the size of the array), including the NULL character at the end
- Can be used with `cin` to assign a value:

  `cin >> myName;`



Program 3-4

```
// This program demonstrates how cin can read a string into
// a character array.
#include <iostream>
using namespace std;

int main()
{
   char name[21];

   cout << "What is your name? ";
   cin >> name;
   cout << "Good morning " << name << endl;
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
What is your name? Charlie [Enter]
Good morning Charlie
```

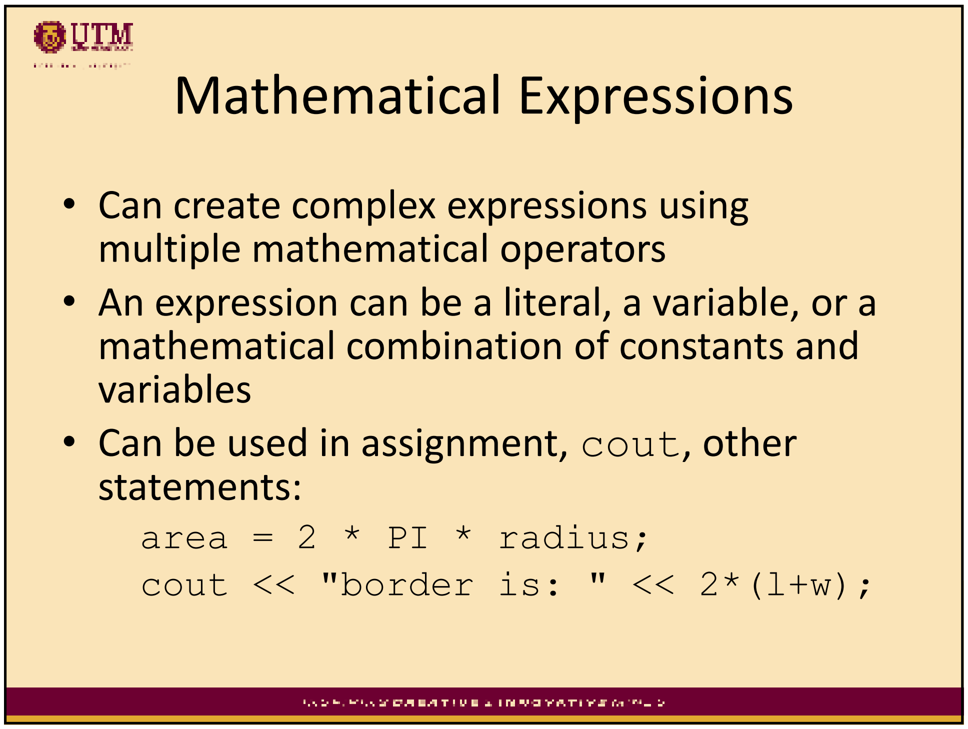
In-Class Exercise

- Refer to Lab 6, Exercise 3 No. 1 (pg. 79).

- Solve the problem.
- Add array of characters to the output.

Sample of output:

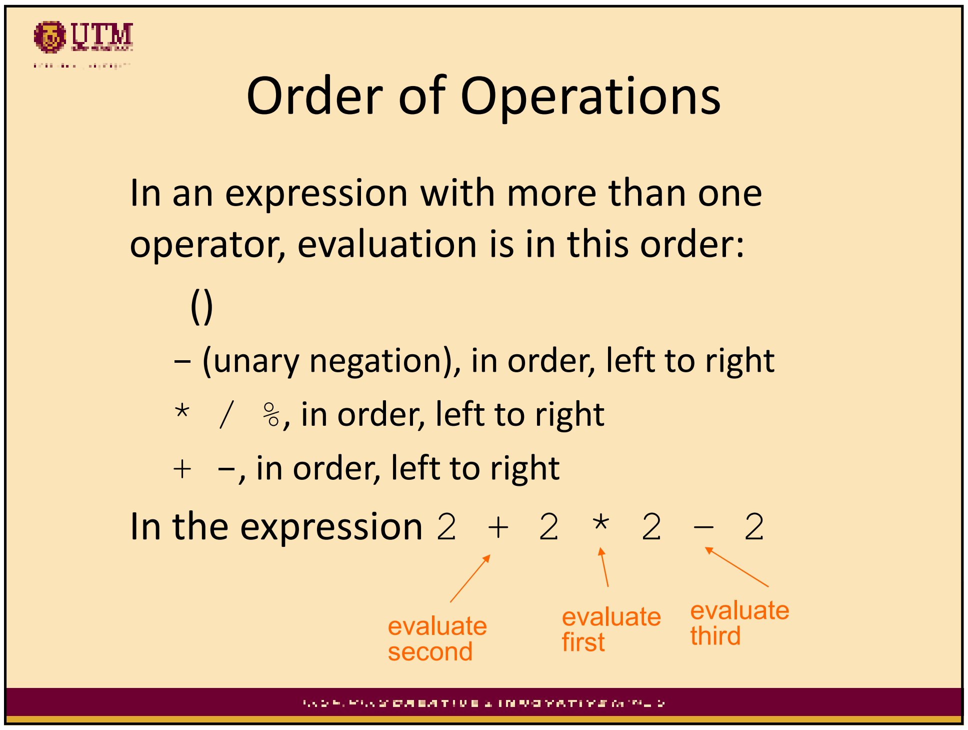
Enter an integer: 7
Enter a decimal number : 2.25
Enter a single character : R
Enter an array of characters: Programming



Mathematical Expressions

- Can create complex expressions using multiple mathematical operators
- An expression can be a literal, a variable, or a mathematical combination of constants and variables
- Can be used in assignment, `cout`, other statements:

```
area = 2 * PI * radius;
cout << "border is: " << 2*(l+w);
```



Order of Operations

In an expression with more than one operator, evaluation is in this order:

()

– (unary negation), in order, left to right

`*  /  %`, in order, left to right

`+  –`, in order, left to right

In the expression `2 + 2 * 2 – 2`

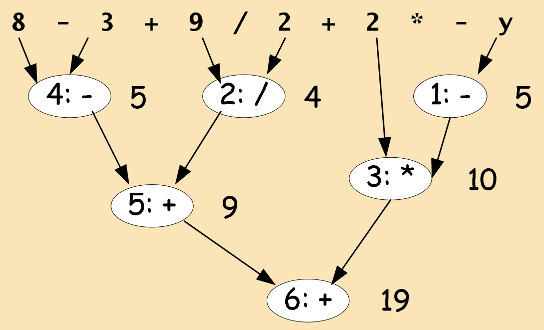
evaluate second

evaluate first

evaluate third

Example

```
int z, y=-5;
z= 8 - 3 + 9 / 2 + 2 * - y;
z= 8 - (3 + 9 / 2) + 2 * - y;// try this
```



Order of Operations

Show prove for the following expression

Table 3-2 Some Expressions

Expression	Value
5 + 2 * 4	13
10 / 2 - 3	2
8 + 12 * 2 - 4	28
4 + 17 % 2 - 1	4
6 - 3 * 2 + 7 - 1	6

Associativity of Operators

- − (unary negation) associates right to left
- `*, /, %, +, −` associate left to right
- parentheses ( ) can be used to override the order of operations:

```
 2 + 2  *  2 − 2  = 4
(2 + 2) *  2 − 2  = 6
 2 + 2  * (2 − 2) = 2
(2 + 2) * (2 − 2) = 0
```

Grouping with Parentheses

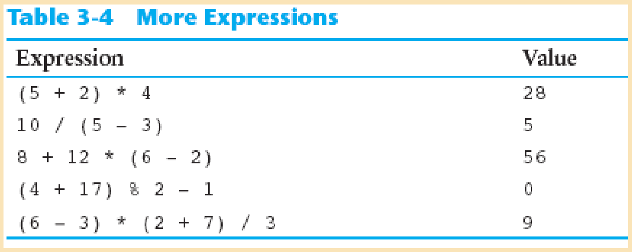


Table 3-4 More Expressions

Expression	Value
`(5 + 2) * 4`	28
`10 / (5 - 3)`	5
`8 + 12 * (6 - 2)`	56
`(4 + 17) % 2 - 1`	0
`(6 - 3) * (2 + 7) / 3`	9

Algebraic Expressions

- Multiplication requires an operator:

  $Area=lw$ is written as `Area = l * w;`
- There is no exponentiation operator:

  $Area=s^2$ is written as `Area = pow(s, 2);`
- Parentheses may be needed to maintain order of operations:

  $$m = \frac{y_2 - y_1}{x_2 - x_1}$$ is written as `m = (y2-y1) /(x2-x1);`

Algebraic Expressions

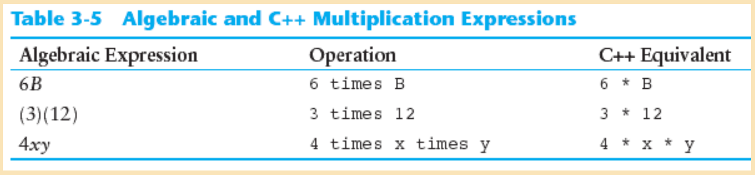
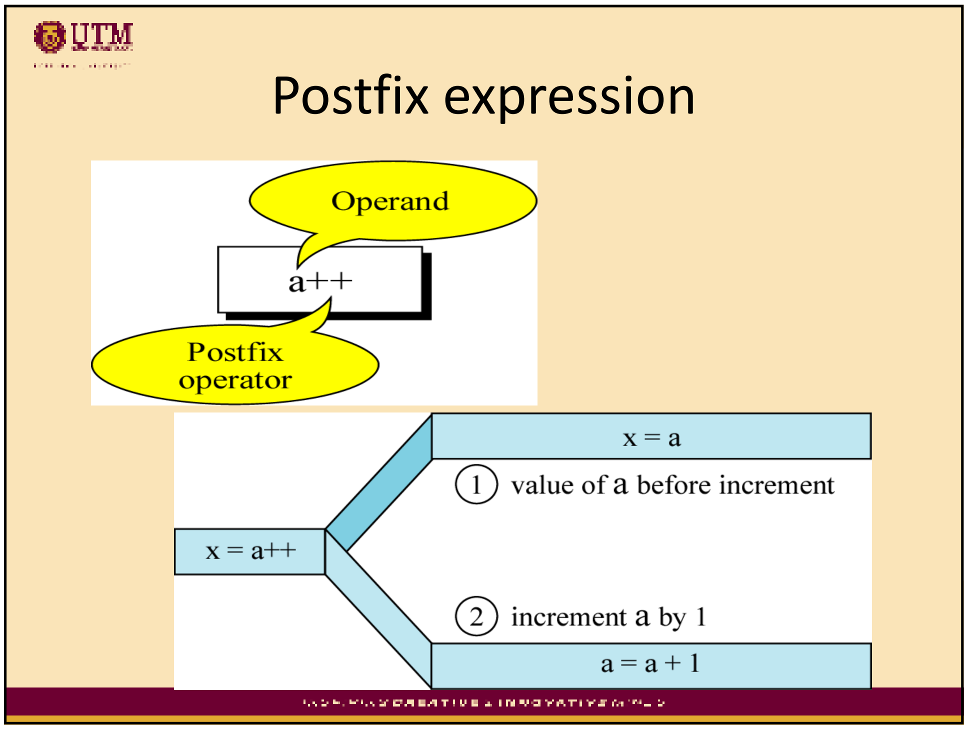
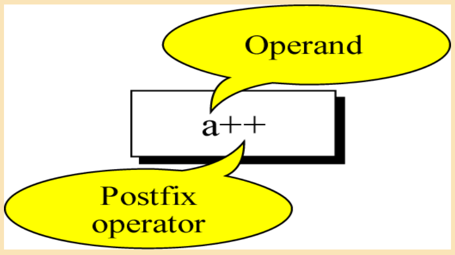


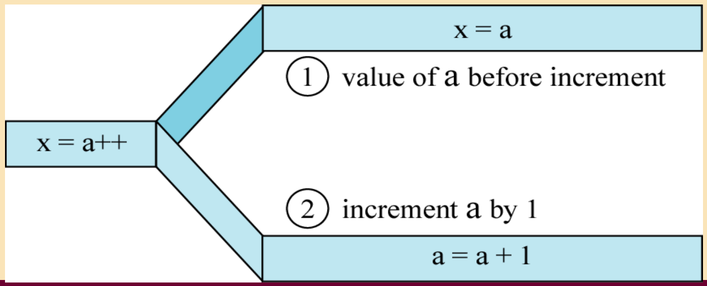
Table 3-5 Algebraic and C++ Multiplication Expressions

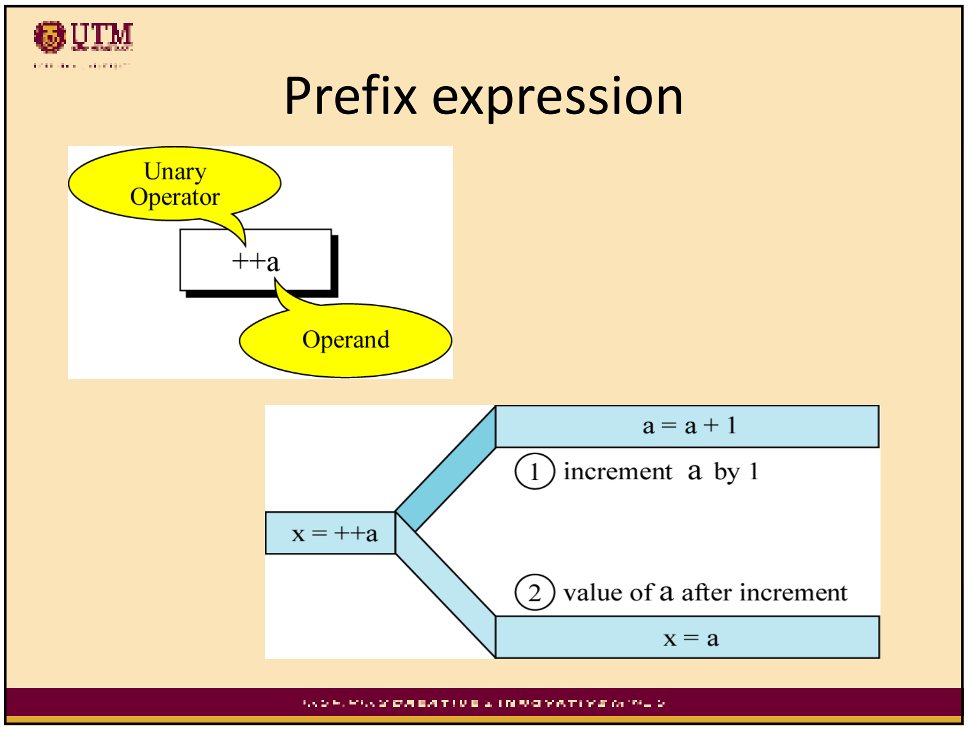
Algebraic Expression	Operation	C++ Equivalent
$6B$	`6 times B`	`6 * B`
$(3)(12)$	`3 times 12`	`3 * 12`
$4xy$	`4 times x times y`	`4 * x * y`



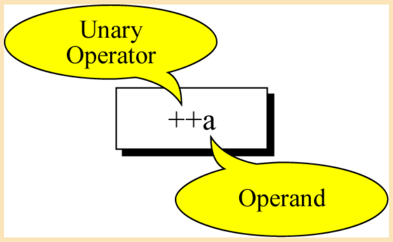
Postfix expression

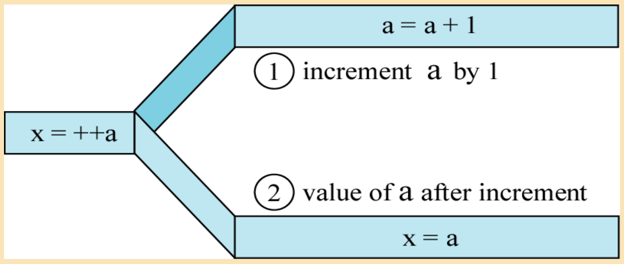






Prefix expression





In-Class Exercise

- What would be the value of nilai_kedua:

```
int kira = 5;
int nilai_pertama = 10, nilai_kedua;

nilai_kedua= 5* kira-- + nilai_pertama;
nilai_kedua = 5* --kira +nilai+pertama;
```

Mathematical function library

- Required header: `#include <math.h>`
- Refer to “predefinefunction” notes in your elearning.
- Example:

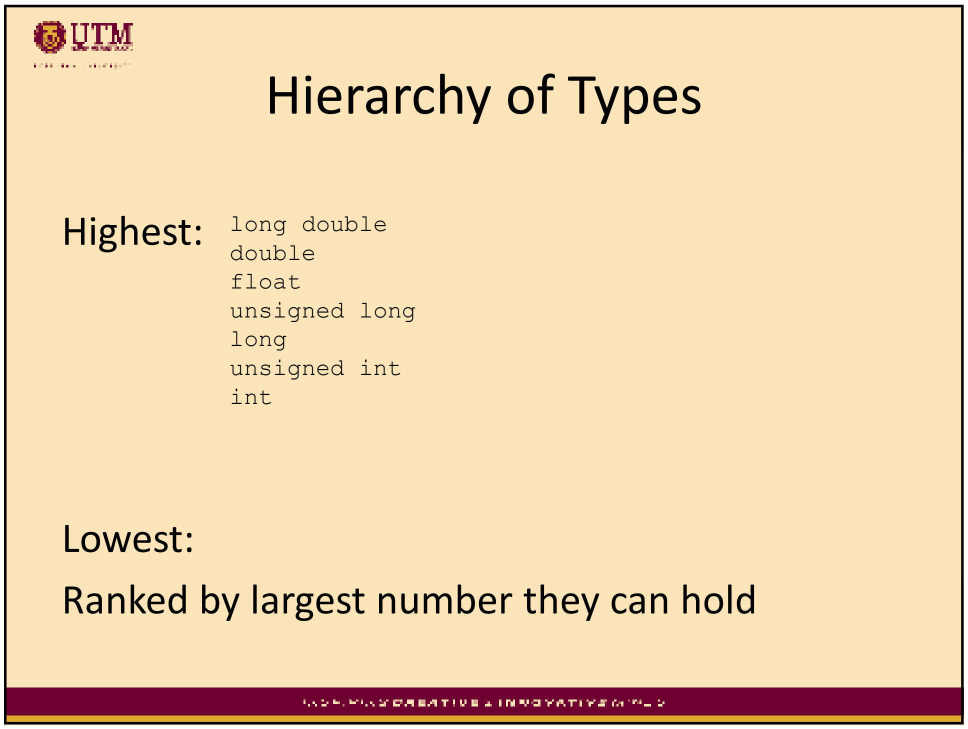
```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{    int x;
     cin>>x;
     cout<<pow(x,2);
     system("PAUSE");
     return 0;
}
```

In-class Exercise

- Do Lab 5, Exercise 1, No. 5 (pg. 59)
- Do Lab 5, Exercise 1, No. 6 (pg. 60)
- Do Lab 5, Exercise 1, No. 7 (pg. 60)
- Do Lab 5, Exercise 2, No. 2 (pg. 63)

When You Mix Apples and Oranges: *Type Conversion*

- Operations are performed between operands of the same type.
- If not of the same type, C++ will convert one to be the type of the other
- This can impact the results of calculations.

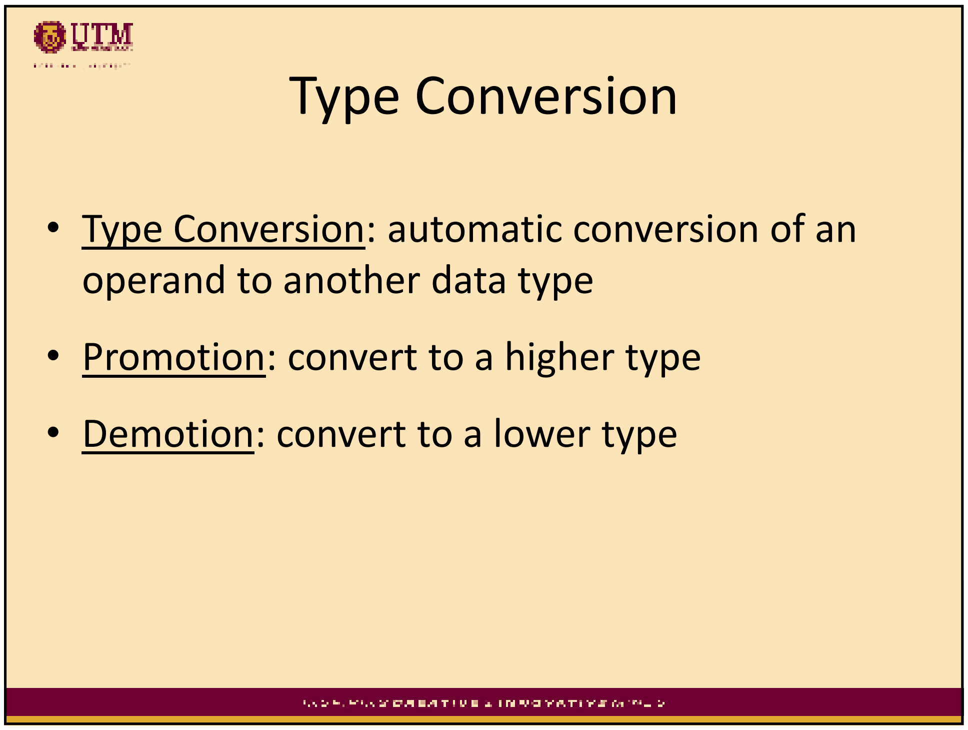


Hierarchy of Types

Highest:
```
long double
double
float
unsigned long
long
unsigned int
int
```

Lowest:

Ranked by largest number they can hold



Type Conversion

- Type Conversion: automatic conversion of an operand to another data type
- Promotion: convert to a higher type
- Demotion: convert to a lower type

Conversion Rules

1) `char`, `short`, `unsigned short` automatically promoted to `int`
   - For arithmetic operation

   `char c='A'; cout<<6+c;` // int

2) When operating on values of different data types, the lower one is promoted to the type of the higher one.

   `int i=25; cout<<6.1+i;` // float

3) When using the = operator, the type of expression on right will be converted to type of variable on left

```
int x, y =25; float z=2.5;
x=y+z; //int
```

In-Class Exercise

- Do Lab 5, Exercise 1, No. 8 (pg. 61)

Type Casting

- Used for manual data type conversion
- Useful for floating point division using ints:

```
double m;
m = static_cast<double>(y2-y1)
                      /(x2-x1);
```

- Useful to see `int` value of a `char` variable:

```
char ch = 'C';
cout << ch << " is "
     << static_cast<int>(ch);
```

Example

Program 3-10

```
// This program uses a type cast to avoid integer division.
#include <iostream>
using namespace std;

int main()
{
   int books;          // Number of books to read
   int months;         // Number of months spent reading
   double perMonth;    // Average number of books per month

   cout << "How many books do you plan to read? ";
   cin >> books;
   cout << "How many months will it take you to read them? ";
   cin >> months;
   perMonth = static_cast<double>(books) / months;
   cout << "That is " << perMonth << " books per month.\n";
   return 0;
}
```

Program Output with Example Input Shown in Bold

```
How many books do you plan to read? 30 [Enter]
How many months will it take you to read them? 7 [Enter]
That is 4.28571 books per month.
```

C-Style and Prestandard Type Cast Expressions

- C-Style cast: data type name in `()`

```
cout << ch << " is " << (int)ch;
```

- Prestandard C++ cast: value in `()`

```
cout << ch << " is " << int(ch);
```

- Both are still supported in C++, although `static_cast` is preferred

In-Class Exercise

- Do Lab 5, Exercise 2, No. 3 (pg. 63)
- Do Lab 5, Exercise 2, No. 4 (pg. 64)
- Do Lab 5, Exercise 1, No. 5 (pg. 62)

Multiple Assignment and Combined Assignment

- The = can be used to assign a value to multiple variables:

```
x = y = z = 5;
```

- Value of = is the value that is assigned
- Associates right to left:

```
x = (y = (z = 5));
```

value is 5 value is 5 value is 5

Combined Assignment

- Look at the following statement:

```
sum = sum + 1;
```

This adds 1 to the variable **`sum`**.

Other Similar Statements

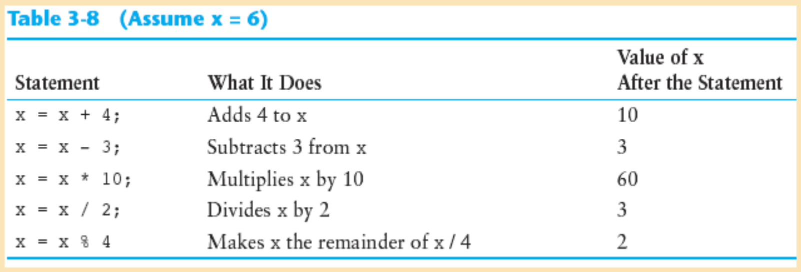


Table 3-8 (Assume x = 6)

Statement	What It Does	Value of x After the Statement
`x = x + 4;`	Adds 4 to x	10
`x = x - 3;`	Subtracts 3 from x	3
`x = x * 10;`	Multiplies x by 10	60
`x = x / 2;`	Divides x by 2	3
`x = x % 4`	Makes x the remainder of x / 4	2

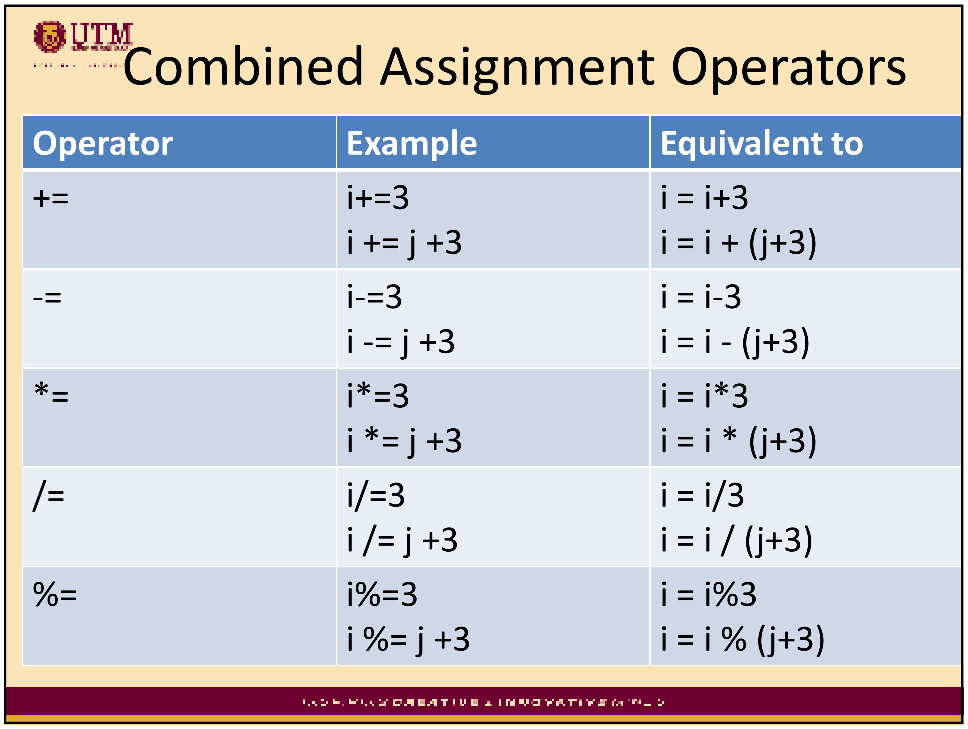
Combined Assignment

- The combined assignment operators provide a shorthand for these types of statements.
- The statement

```
sum = sum + 1;
```

is equivalent to

```
sum += 1;
```



Combined Assignment Operators

Operator	Example	Equivalent to

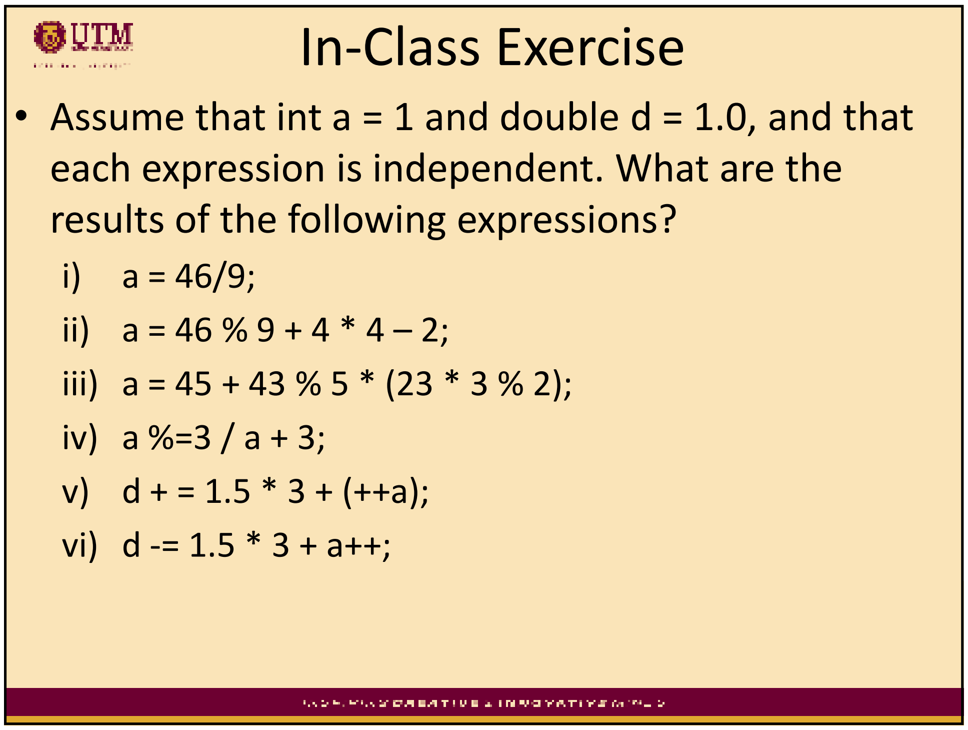
+=	i+=3
	i += j +3
		i = i+3
		i = i + (j+3)

-=	i-=3
	i -= j +3
		i = i-3
		i = i - (j+3)

*=	i*=3
	i *= j +3
		i = i*3
		i = i * (j+3)

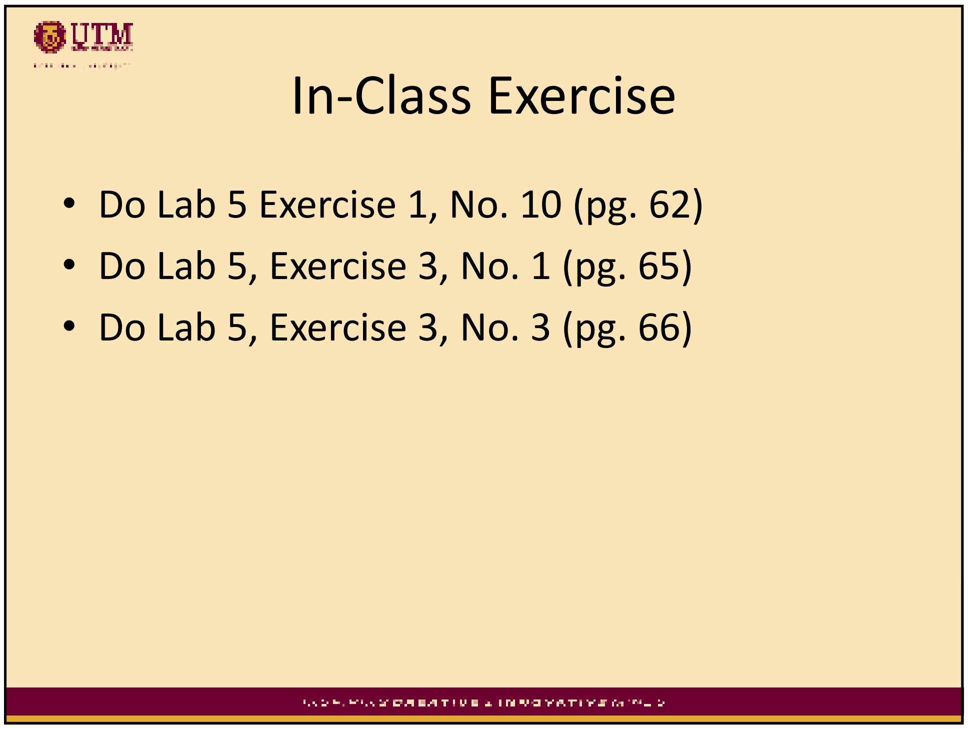
/=	i/=3
	i /= j +3
		i = i/3
		i = i / (j+3)

%=	i%=3
	i %= j +3
		i = i%3
		i = i % (j+3)



In-Class Exercise

- Assume that int a = 1 and double d = 1.0, and that each expression is independent. What are the results of the following expressions?
  i) a = 46/9;
  ii) a = 46 % 9 + 4 * 4 – 2;
  iii) a = 45 + 43 % 5 * (23 * 3 % 2);
  iv) a %=3 / a + 3;
  v) d + = 1.5 * 3 + (++a);
  vi) d -= 1.5 * 3 + a++;



In-Class Exercise

- Do Lab 5 Exercise 1, No. 10 (pg. 62)
- Do Lab 5, Exercise 3, No. 1 (pg. 65)
- Do Lab 5, Exercise 3, No. 3 (pg. 66)