

```
Embedded Assignment Expressions

• Assignments can also be embedded Example cout << (num2 = 10);

This performs 2 tasks

->1. it assigns the value 10 into the variable num2
->2. it displays the contents of the variable num2 on the screen

Assignments are expressions NOT statements

• They can be used anywhere an expression can be used
```

```
Example

num2 = 3;
num3 = num2 + 5 * (num1 = 7);

This statement is evaluated as follows:

1. num1 is assigned the value 7
num3 = num2 + 5 * 7

2. The multiplication is evaluated
num3 = num2 + 35

3. The addition is evaluated
num3 = 38

Two assignment statements were made in the 2<sup>nd</sup> statement.

The value 7 was stored in num1

The value 38 was stored into num3

WARNING

Doing this in practice can cause you needless hours debugging!
And your friends who help you debug will not appreciate it!
This makes your code confusing to understand
Bad style
```

```
Evaluate the following assignment expressions.

Int in1
Int in2;
float fn3;
in1 = ( fn3 = (in2 = 5) * 4 / 8.0 ) * 2;
cout << in1 << endl << in2 << endl << fn3 << endl;

in1 = ( fn3 = (in2 = 5) * 4 / 8 ) * 2;
cout << in1 << endl << in2 << endl << fn3 << endl;

Evaluate this one on your own
```

```
Evaluate the following assignment expressions.

int in1,in2;
float fn3,fn4;

in1 = ( fn3 = (in2 = 5) * (4 / 8.0) ) * 2;

if ((fn4 = (in1 = (in2 * 2) + fn3)) > 10) {
      cout << fn4;
    }
else
{
      cout << "Test val is 10 or less";
    }
```

```
Evaluate the following assignment expressions.

int in1, in2;
float fn3,fn4;

in1 = ( fn3 = (in2 = 5) * (4 / 8) ) * 2;

if ((fn4 = (in1 = (in2 * 2) + fn3)) > 10)
{
    cout << fn4;
}
else
{
    cout << "Test val is 10 or less";

    Advanced Arithmetic 7
```

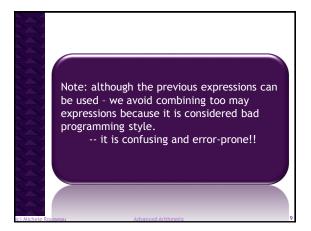
```
Exercises

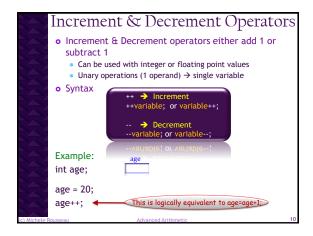
Rewrite the following statement → one operation at a time if ((x = y) < z)

How will this be evaluated?

if (x = y < z)

Remember order of precendence? Which comes first?
```





```
Prefix & Postfix

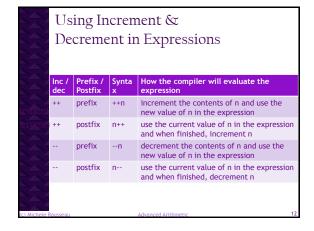
Prefix → ++age; (or --age;)

Postfix → age++; (or age--;)

• Using these operators alone will produce the same results

• Using them as part of a larger expression may not

• → the compiler does not evaluate them the same way
```



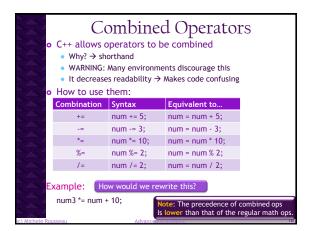
```
int prelnc;
                         Increment Examples
int postlnc;
                                 VARIABLES
int lcv;
                                  lcv preinc postinc
preInc = 1;
postInc = 1;
cout<<"lcv Pre-Inc Test Post-Inc Test\n";
for (lcv = 1; lcv <= 3; ++lcv)
  cout << lcv/<< '\t';
  cout << ++preInc << "\t\t";
  cout << postInc++ << endl;
                 Inc after
cout << "\nIn the end they are the same: ";
cout << preInc << '\t' << postInc;
                                 Let's do a desk check
```

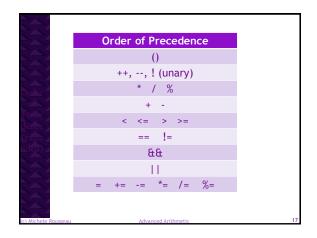
```
Increment & Decrement Examples

int preDecTest, postIncTest, preDecTest, postDecTest,;

preIncTest = 3;
postIncTest = 3;
preDecTest = 3;
preDecTest = 3;
postDecTest = 3;
result = 4* ++preIncTest;
result = 4 * postIncTest++;
What are the values of preIncTest and postInctest now?

result = 4 * -- preDecTest;
result = 4 * postDecTest--;
What are the values of preDecTest and postDectest now?
```





```
Combined Operators Examples

Write statements using combined assignment operators to perform the following:

a) Subtract 5 from n1 & store the result in n1

b) Add n1 * 8 to n2 & store the result in n2

c) Get the remainder of n3 divided by 5 and store the result in n3
```

```
# include <iostream>
using namespace std;
int main()
{
    int a, b, c, d, e, f;
    c = 2;
    d = 5;
    e = 2;
    f = 8;

    b = (c++) + c;
    a = (b = c++) * --d / (e += f++);

    cout << a << endl << b << endl;
    cout << c << endl << d << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << endl << endl;
    cout << e << endl << f << endl;
    cout << e << endl << f << endl;
    cout << endl << endl;
    cout << e << endl << endl;
    cout << e << endl << endl;
    cout << endl << endl << endl;
    cout << endl << endl
```

```
# include <iostream>
                         Exercise #2
using namespace std;
int main()
                       b c d e f
{
  int a, b, c, d, e, f;
  a = 2;
  b = 5;
  c = 10;
  b *= c;
  d = --b * c++;
  e = --c * (a += 5);
  f = --a + --b * c++;
  cout << a << endl << b << endl;
  cout << c << endl << d << endl;
  cout << e << endl << f << endl;
  return 0;
```