Programming Notes 3: Math

Always remember, I will provide examples of the notes in the example program.

1. Intro to Math

It's time to go back to basic arithmetic for a while, because while basic math works more or less how you'd think, there are some important things you have to remember when doing math in your programs.

2. Integers – Addition, Subtraction, Multiplication

Well, integer math is pretty much exactly the same as what you're familiar with. Adding two integers adds them, subtracting two integers subtracts them, and multiplying two integers multiplies them. To add two values, put a plus sign between them, to subtract two values put a dash between them, and to multiply two values put an asterisk between them. You can chain as many operations together as you want. However, do remember that your math operation will not do anything itself—you must assign the result to another variable. Examples:

```
addResult = valueOne + 15;

multiply = x * y * z;

result = 5 - numSets + 23;
```

3. Integers – Division

Now, this is where programming math starts to vary from the way you've learned it in your math classes. You'll remember that integers cannot store any fractional values, so what happens if you divide two integers that don't divide evenly? Well, the short answer is the fractional part is chopped off, or truncated. If you think of the answer you should get, you'll just get the part before the decimal point. To be clear–this doesn't round your value, it just truncates it. This doesn't only apply to division with integers—this will happen any time you try to assign a non-integer value to an integer. Examples:

```
result = 10 / 7;
```

This will make result hold the value 1, since 10 divided by 7 is about 1.43.

```
divResult = 34 / 50;
```

This will make divResult hold the value 0, since 34 / 50 is 0.68.

```
int value = -1.5;
```

This will make value hold the value -1, since it cannot hold the .5 part.

4. Floating Point Math

Here's a quick foray back into the realm of normal—floating point math works exactly as you'd expect it. However, remember that the larger values a floating point variable holds, the less precise those values are.

5. Modulo

There is another basic mathematical operation that you may not have used before—the modulo. The modulo operation can only be used between two integers, and it gives you the remainder of dividing the left value into the right one. To use it, it is the same as the other basic operations of addition and the like, except you use the operator '%'. Examples:

modResult = 12 % 13;

This evaluates to 12, as it is the remainder from dividing 12 into 13.

anotherMod = 15 % 5;

This evaluates to zero, as five divides cleanly into 15.

finalMod = 7 % 3;

This evaluates to one, as it is the remainder from dividing 3 into 7.

6. Order of Operations

The order of operations, more commonly referred to as operator precedence, in C++ is mostly the same as it is in "normal" math, with a few exceptions. Additionally, just as in normal math, you can force anything to become top priority by wrapping it in parenthesis. If you're not sure of the order of a statement, just use parenthesis.

Order	Rule
1	Parenthesized operations, using all of these rules.
2	Multiplication, division, and modulo from left to right.
3	Addition and subtraction from left to right.
4	The assignment operator is always last.

7. Combining Math and Assignment

There will often be times when you want to do something to a variable itself, and not assign the new value to a different variable. With the previous information, you could write statements like

var = var + 5;

but there is a much easier and quicker way to do this. This is with combined math and assignment operators. These include '+=', '-=', '/=', '*=', and '%='. (There are also some more, but we'll learn them later.) These new operators are simply contractions of longer statements. For example:

var += 5 is equivalent to writing var = var + 5

result *= value + 1 is equivalent to writing result = result * (value + 1)

And so on. Finally, there is an even quicker way to add or subtract one from a variable: the increment and decrement operators. These are '++' and '--' respectively. For example:

value++ is equivalent to writing value += 1 which is also equivalent to writing value = value + 1

var-- is equivalent to writing var -= 1 which is also equivalent to writing var = var - 1

8. Character Math

There is one more short topic that you might not expect. Remember that characters are technically a form of integer? Well, you can use them with all of these mathematical operators if you'd like. While you don't need to know every number that corresponds to each character, you should know that the alphabet is a range of 26 integers. This means that if you were to increment the character 'b', you would get the character 'c'. The same rule holds true for capital letters as well, but there isn't a super simple relationship between the lowercase and capital letters. However, as an example of character math, you can transform a lowercased letter to a capital letter with the following statement:

capital = lowercase – 'a' + 'A';

By subtracting 'a' from the variable lowercase, you get the offset from 'a', meaning if the lowercase letter was a 'c', you would get two, because it is two away from 'a'. Then, the offset is added to 'A', producing the same letter as before, but as a capital. For example, if the offset was two, you would go two letters from 'A', and get 'C'.

9. Cmath

Finally, you may have noticed that I haven't mentioned anything about ways to do other operations like take a number to a power, or take the sine of a number, or round a number, or the like. That's because these functions aren't actually part of the C++ language. However, that doesn't mean you have to program them yourself (for now). There is a library (just like iostream) called "cmath" that includes pretty much all general math functions you could need. Here's a list of some of the ones you might use:

cos() – cosine

sin() – sine

tan() – tangent

acos() – arc cosine or inverse cosine

asin() – arc sine or inverse sine

atan() – arc tangent or inverse tangent

exp() – the exponential function

log() – natural logarithm

log10() – logarithm base 10

pow() – a number to a power

sqrt() – square root

ceil() – round up

floor() – round down

round() – round value

abs() – absolute value