Lesson 2: Simple Math

Hello and welcome back to the Teachers Edition of a Guided Intro to Programming. This guide will mirror the exact same guide that the students will be using except this guide will have tips and advice on how to help students if they encounter errors.

lesson will be about math. I know some of you are probably rolling your eyes right now and saying "I hate doing math". Well fear not because we don't actually have to do any math. Today we will be learning how to make a computer do the math for us. By the end of the lesson you will be able to do simple math equations such as add, subtract, multiply, and divide.

To start todays lesson we will once again make our way to the online compiler we used in the previous lesson. To do so either click or copy this link https://repl.it/languages/c. You will not need the "Hello World" function in this lesson so feel free to delete what is inside the braces of the int main(void) { }, but remember to not delete anything outside of that otherwise our code will not work.

Today's lesson will be a great introduction to those of you who have not yet begun algebra and a good review for those of you who have. This is because we will be working with variables and the order of operations that they are involved in.

To start todays lesson we will go over a brief explanation of what data types are and what they do. You were actually introduced to data types in the last lesson in the form of int and char. These are called data types because they define what traits a variable has. For example, an int makes a variable hold a number value, while a char can only hold characters. This is important to know in the future because each data type has their own traits that can change how a program functions. For this lesson we will need two data types: int which you should be familiar with and double. The double data type doesn't do what you may be thinking it does, it does not double the value in it, rather it holds a larger maximum number than int does and it also can contain decimal points, unlike int.

For future reference here is a short list of data types, what they do, how to reference them and how to declare them.

Data Type	What they do	Reference	Declaration
Int	Contains an integer that can be either positive or negative. Cannot have decimal points	%d	int x = 0;
Char	Contains characters that are either the alphabet or punctuation.	%c	char x = ‘A’;
Long	Has the same properties as an integer but has a vastly larger maximum value.	%li	long x = 0;
Double	Referred to as a floating point, which makes its max value much greater than an int, can also contain decimal points.	%d	double x = 0;

Now that we know a bit more about data types and how they affect variables we can start learning how to make our computer do math for us. It is important to know that just like the math you do there are rules to math on a computer. Just like you a computer follows the rules of PEMDAS where it will do the commands you input in parentheses first, then multiplication and division, and finally addition and subtraction. We will not need to do exponents today so not to

worry about that. However, unlike you and I, a computer cannot just look at a complex equation like $x = \frac{(9*10)+2}{12-8}$. We can easily read it and take the steps to complete the problem with little issue. But unlike us a computer is methodical, meaning it reads everything from left to right. This is what the computer will see $x = (9 * 10) + {}^{2}/_{12} - 8$. As you might notice it is very different from what we see. This is because we need to be specific when doing arithmetic like this. To make the equation come out to the same answer we will put parts of it in parentheses like so, $x = ((9 * 10) + 2)/(12 - 8)$. This is an equation that the computer can read and it will give us the same answer. It is also important that we place our sum or the answer that we want on the left side of the equal sign. This is because in computing and programming making a variable hold a value looks like this, Variable name = value you want to store.

Remember that a computer must read from left to right in order to math arithmetic, if a student is having an issue and is not getting the right output check their syntax in the equation, this will be the most likely spot where the issue occurs.

Now that we have an idea of how a computer does math, let's have you try it. We can start with simple addition.

Addition in *C* looks like this:

Sum = num1 + num2;

Very simple, and using what we learned in the last lesson we can make the computer tell us what it did using the printf function like so.

printf("%d + %d = %d\n", num1, num2, sum);

Test what you learned in this lesson so far to declare 3 int variables (num1, num2, and sum) and apply what you learned in the last lesson with the printf and scanf functions to prompt a user to enter 2 integers and add them together and show the output. If you need get stuck you can ask your teacher, look back at the last guide as a reference, or you can look online for some tips and tricks to complete this code.

Your screen when you input the code should look like this

```
clang-7 -pthread -lm -o main main.c
./main
Please enter an integer: 6

Please enter another integer: 5
The numbers you chose are 6 and 5
6 + 5 = 11
```

The Student's Code should look something like this:

```
#include <stdio.h>

int main(void) {

int sum = 0;
int num1 = 0;
int num2 = 0;

printf("Please enter an integer: ");
scanf("%d", &num1);
printf("Please enter another integer: ");
scanf("%d", &num2);

sum = num1 + num2;

printf("%d + %d = %d", num1, num2, sum);

return 0;
}
```

If students are struggling feel free to give them hints as to what to do by referencing this code. Be sure to focus on the highlighted areas as this is where students may make an error and once they get one of the highlighted areas working correctly the rest should be easy to get. On top of all of this be sure to check that they have their semicolons in the right place and that they're code is inside the int main(void) { CODE }.

If you have achieved this output

Congratulations

Now that you have completed that lets try what you just learned with subtracting two variables to get an answer. To get from what you just did to show subtraction is very easy. You will only need to change a few things in your code.

The output for this code should look like this:

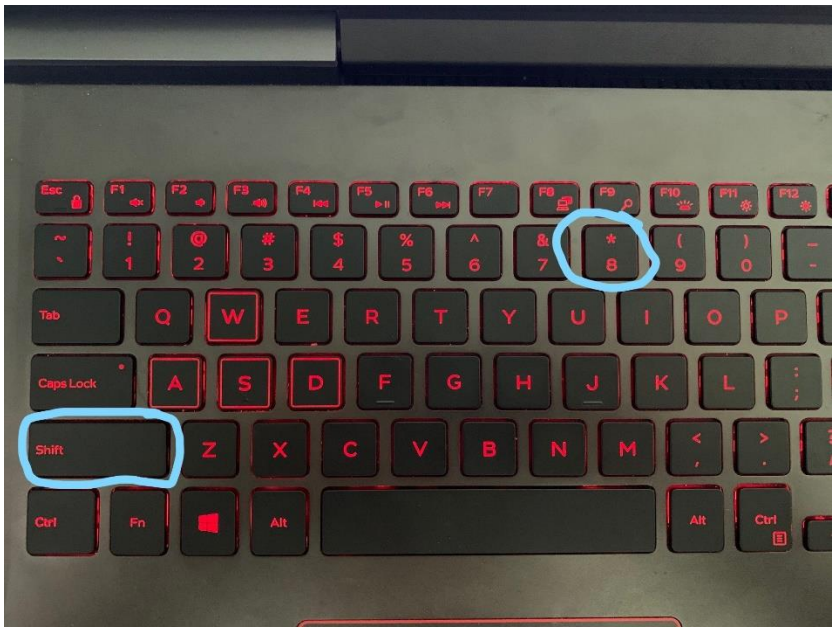
```
clang-7 -pthread -lm -o main main.c
./main
Please enter an integer: 8
Please enter another integer: 5
8 - 5 = 3
```

The code for this output should look like this:

```c
#include <stdio.h>

int main(void) {

int sum = 0;
int num1 = 0;
int num2 = 0;

printf("Please enter an integer: ");
scanf("%d", &num1);
printf("Please enter another integer: ");
scanf("%d", &num2);

sum = num1 - num2;

printf("%d - %d = %d", num1, num2, sum);

return 0;
}
```

The highlighted areas are the only place where the students needed to change anything and that change was replacing the '+' with '-'. If they kept the original code they made and just added this to their new code and the subtraction output is correct then everything is fine.
Now that we have learned about addition and subtraction in programming we will now move on to multiplication and division.

In order to do multiplication in *C* we use the '*' key which is done by holding down the shift key and hitting the 8 on the top of the keyboard while still holding the shift key, pictured here.



This will allow us to use multiplication which will look like this:

Sum = num1 * num2;

Now for division we will use '/' which is located next to the arrow keys on the keyboard and will look like this in code:

Sum = num1 / num2;

When doing division we will want to make the variable that we wish to hold our answer in a Double type variable to hold decimals. This can be done by using the code shown in the data type table that was shown previously or by using this code:

double sum = 0;

Now try it out using what you learned before to multiply and divide 2 numbers to get the answer.

Your output should be similar to this:

```
clang-7 -pthread -lm -o main main.c
./main
Please enter an integer: 8
Please enter another integer: 2
8 * 2 = 16
8 / 2 = 4
```

Congratulations

You have now learned to do simple math in *C*.

There only one last thing you need to learn to complete this lesson and that is how to do slightly more advanced equations using parentheses. Do you remember that small equation from earlier?
$x = \frac{(9*10)+2}{12-8}$

That's the one, well this equation is too complicated as it is now for a computer to do so if we want to find the answer, we will need to split up parts of the equation into parts that a computer can solve. We do this using parenthesis as such:

$$x = (9 * 10) + \frac{2}{12} - 8$$

This is what the computer would see if we put the equation in as it is written so let's use parentheses to make this easier, first let's do the top of the fraction:

$$x = \frac{((9 * 10) + 2)}{12} - 8$$

Now that we have closed the top in parentheses this is what the computer will see but we still want to subtract 8 from 12 on the bottom of the fraction so let's do the bottom now:

$$x = \frac{((9 * 10) + 2)}{(12 - 8)}$$

Now this what the computer will see when all the parts of the equation are closed in parentheses now in code it will look like this:

X = ((9*10) +2) / (12-8);

And with that you have completed lesson 2 of a Guided Intro to Programming, I hope to see you again in the next lesson.

As an idea if you wish to have the students try something on their own as an assignment you could have the students read a number from input and square it, or give them an equation and have them program the equation to give an answer when given 2 numbers to read from input.

For example:

Squaring a number

Num1 = 3

Square = num1 * num1;

Equation

Num1 = 8

Num2 = 5

$$x = \frac{num1 * 2}{num2 - 3}$$

Code

X = (num1 * 2)/(num2 – 3);

These are just ideas for having students practice their problem-solving skills alongside their programming. It is important to develop and highlight the importance of problem-solving skills early as these skills can help students further down the line in education or at a job.

Glossary

Arithmetic: The branch of mathematics dealing with the properties and manipulation of numbers.

Char: variablecaller = %c , A char, which is short for character, is a variable that holds character values like letters or punctuation.

Data Types: A particular kind of data item that is defined by the values it can accept

Declare: The act of creating a variable in programming, Example: int x = 0; creates the variable x and sets it equal to 0 until we change it.

Double: variablecaller = %f , A double is a data type that is like an int except it has a larger maximum value and can hold decimal points

Int: variablecaller = %d , An int is a variable that holds numerical values that are either positive or negative, has a maximum value of 2,147,483,647

PEMDAS: Stands for Parentheses, Exponent, Multiplication, Division, Addition, and Subtraction, a mnemonic for the order of operations in a mathematical equation, computers also follow this same order.

Created By:
Ryan Floyd, Computer Science, ‘22

Thank you for downloading this lesson, please take a moment to complete our survey