Lecture 1 Notes

- We want a program what will do the following calculations
- Note that the text in **bold** are the outputs, and the normal text are your inputs

```
How many hours did you work? 40
What is your hourly rate of pay? 16.13

You earned $645.20
$64.52 will be withheld.
```

```
#include <iostream>
using namespace std;

int main()
{
cout << "How many hours did you work? ";
double hoursworked;
cin >> hoursworked;

cout << "What is your hourly rate of pay? ";
double payRate;
cin >> payRate;

double amtEarned = hoursworked * payRate;
cout.setf(ios::fixed);
cout.precision(2);
cout << "You earned $" << amtEarned << endl;
cout << "$" << (0.10 * amtEarned) << " will be withheld." << endl;
}
```

- Along the way, you can run the following to see if the computer registered the correct inputs

```
cout << "hours worked is " << hoursworked << endl;
cout << "pay rate is " << payRate << endl;
```

- Note that `endl` stands for “end line” and is used to move onto the next line
- `#include <iostream>` allows the C++ compiler to use the input and output stream
    - For `cout` (output) and `cin` (input)
- `std` stands for the standard library

Identifiers

- There are two ways to declare identifiers:

```
type identifier;
type identifier = expression;
```

- Rules for identifiers:
    - The first character must be a letter, either uppercase or lowercase
    - Any subsequent characters are optional, but can be a letter, digit, or underscore
- Examples:

```
fred
covid19
covid_19
hours_worked
hoursworked
```

- Types
    - `double` is used to define values that hold numbers and decimal points
        - Can be positive, negative, or zero
        - 15 to 16 decimal digits of precision
        - From 10^{-308} to 10^{308}
    - `int` is used to define whole numbers from -2 billion to 2 billion

- The following code tells the program that all subsequent outputs are to be 2 decimal places in precision

```
cout.setf(ios::fixed);
cout.precision(2);
```

Mathematics

- Most order of operations PEMDAS from algebra carry over to C++
- Multiplication and division have higher precedence than addition and subtraction
  - Operators with equal precedence are read from left to right
- However, the star operator **must** be used for multiplication
  - `2(3+5)` must be written as `2*(3+5)`
- Four possible cases for division operations, and one case for the remainder modulus:
  - Double ÷ Double = Double

    `14.3/5.0 = 2.86`
  - Double ÷ Int = Double

    `8.65/3 = 2.8833`
  - Int ÷ Double = Double

    `14/5.0 = 2.8`
  - Int ÷ Int = Int

    `14/5 = 2`
  - Remainder Modulus

    `14%5 = 4`
- Most compilers will restrict outputs to **four decimal places**
- If less than four decimal places, doubles will be **truncated** to the last nonzero digit

The following examples are logic errors:

- Sometimes, there might be a 0 in the denominator, and the program might crash

```
int a = 10;
int b = a * a;
int c = 25 / (b-100);
```

- The following example is undefined because `e` is one billion and `f` exceeds the maximum value allowed for an int
    - The output is some random integer

```
int d = 1000;
int e  = d * d * d;
int f = d * e;
```