C++ Refresher

In this lesson, we'll get a quick C++ refresher.

We'll cover the following

Refresher

Let's quickly go over a few concepts, most of which you should already know if you have some experience with C++.

These will come handy and help you avoid common mistakes that beginners make.

Environment

You can always use a simple text editor and terminal to compile and run C++ programs.

I personally like using **codeblocks**. You can download the IDE for most of the platforms here.

#Include

Including libraries can quickly become a pain. You can always Google the required library for a built-in function or data type, but a great alternative is to use a single include statement that includes all libraries or to use a long list of include statements in your code.

```
#include <bits/stdc++.h>
```

If the above doesn't work for you, try this fix.

Global and local variables

Here is how you declare global and local variables.

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

int global = 10;

int main() {
  int local = 5;
  return 0;
}
```

Important: When you want to declare an array of say, a million integers, declaring it as local might not work depending on various factors, like memory issues for one.

But as a general rule, you will see most everyone declares large arrays that the program will need as global, thus creating the array in heap memory. I would suggest doing this.

Note: Declaring variables using the `new` operator or using STL structures like vectors allocates the memory to heap in this case, meaning it can be declared locally as well.

Overflow

Data types `int` and `long long int` are 32-bit and 64-bit integers, respectively.

While doing arithmetic operations, always know the data type's limit.

For example, int (signed) can store values a little over 2 billion, so squaring 5 million will definitely overflow for `int` but not for `long long int`.

```
int x = 5000000;
int y = x * x; //Overflow
long long int = (long long int)x * x; //This works, type casting to long lon
g int
```

In the next lesson, we'll study some handy built-in C++ methods.