Basics of C++

Compiling C++

- Command:
  `g++ source.cpp -o output -Wall -O2 --std=c++11`
  - `source.cpp`: the source file to compile
  - `-o output`: the output filename
  - `-Wall`: enables ‘all’ warnings
  - `-O2`: optimizes the code so it runs faster
  - `--std=c++11`: uses modern C++ features

- To run: `./output`

The basic structure

```
1   #include <iostream>         //alternatively, if you don't mind long compile time:
2   #include <bits/stdc++.h>    //imports everything
3
4   using namespace std;        //without this, you have to type std::cout, etc.
5
6   int main()
7   {
8       cout << "Hello, world!\n";
9       return 0;
10  }
11
```

Input and Output

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);//this line makes I/O faster

    int a;
    int b;
    //int a, b; <-- same thing as above

    cin >> a >> b;//takes in A first and then B (seperated by spaces/newline)

    int ans = a * b;
    cout << a << " times " << b << " is equal to " << ans << '\n';

    return 0;
}

```

Functions and strings

```cpp
#include <iostream>

using namespace std;

//functions must go before (above) where they're used in C++

string reverse(string& input)//pass a pointer, don't copy "input"
{
    string output = "";

    for(int i = 0; i != input.size() + 1; i++)
    {
        output += input[input.size() - i];
    }
    return output;
}

int main()
{
    ios::sync_with_stdio(false);

    string text;
    cin >> text;//reads first word

    string reversed = reverse(text);
    cout << reversed << '\n';

    return 0;
}
```

Variables and Arrays

```cpp
#include <iostream>

//finds and replaces 'INF" with 1000000000
#define INF 1000000000
#define DEBUG

//both of these would do the same thing:
//make 'll' shorthand for 'long long'
#define ll long long
typedef long long ll;

using namespace std;

int main()
{
    //both of these are the same thing
    long long bigNumberA;
    ll bigNumberB;

    double values[100];
    double nextVar = 12.3;

#ifdef DEBUG
    //alternatively, define DEBUG to be true and use if(DEBUG){...}
    cout << "initial value of index 0 is " << values[0] << endl;
    cout << "index 100 is " << values[100]  << endl;
#endif

    return 0;
}
```

STL Basics

```cpp
#include <iostream>
#include <vector>
#include <algorithm>//needed for sort()

using namespace std;

typedef vector<int> vInt;

int main()
{
    vector<string> textList;
    vInt intList;//same as vector<int> intList;


    intList.push_back(2);
    intList.push_back(10);
    intList.push_back(7);

    if(intList.back() == intList[2] && intList[2] == 7)
    {
        cout << intList.front() << '\n';
    }

    auto iter = inList.begin();//get an iterator, figure out the type

    sort(intList.begin(), intList.end());//uses iterators
    return 0;
}
```