C++ Notes on Sheet 1

Mohamed Emary

September 13, 2025

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

This header file includes most of the standard C++ libraries. It is often used in competitive programming to save time on including headers.

`bits/stdc++.h` is a **non-standard header** file of GNU C++ library. So, if you try to compile your code with some compiler other than GCC it might fail; e.g. MSVC do not have this header.

0.2 Fast Input/Output

```cpp
#define fast_io ios::sync_with_stdio(false);
cin.tie(nullptr);
```

0.3 Faster `endl`

You can make `endl` faster by using `#define endl '\n'` instead. This avoids the flushing of the output buffer that `endl` performs.

- `std::endl` is like saying: "Print a new line, AND make sure everything I've printed so far is visible on the screen *right now*."
- `'\n'` is like saying: "Just print a new line." The program will still send the output to the screen eventually, but it will wait until it has enough data to send efficiently, or until it's absolutely necessary (like when the program ends, or when you ask for input).

0.4 String Reverse (Problem-R)

To reverse a string in C++, you can use the `reverse` function from the `<algorithm>` header.

```cpp
#include <algorithm>

// Then inside main()
std::string str = "Hello, World!";
std::reverse(str.begin(), str.end());
std::cout << str << std::endl; // "!dlroW ,olleH"
```

> Note: `std::` can be omitted if you use `using namespace std;` at the beginning of the code.

0.5 Maximum of Three Numbers (Problem-N)

You can find the maximum of three numbers using the `max` function from the `<algorithm>` header.

```
#include <algorithm>

// Then inside main()
int a = 10, b = 20, c = 15;

// Send all numbers to `max` as initializer list
int maximum = std::max({a, b, c});

// OR get the maximum between the first two numbers
// and compare with the third
int maximum = std::max(a, std::max(b, c));
```

0.6 `ceil`, `floor`, and `round` (Problem-Q)

To find the ceiling, floor, or round a number in C++, you can use the `ceil`, `floor`, `round` functions from the `<cmath>` header.

```
#include <cmath>

// Then inside main()
float a = 5, b = 2;
cout << std::ceil(a / b) << endl; // 3
cout << std::floor(a / b) << endl; // 2
cout << std::round(a / b) << endl; // 3
cout << a / b << endl; // 2.5
```

0.7 String `.find` function

To find a substring within a string in C++, you can use the `.find` member function of the `std::string` class.

```
std::string str = "Hello, World!";
size_t pos = str.find("World");
if (pos != std::string::npos) {
    std::cout << "Found 'World' at position: " << pos << std::endl;
} else {
    std::cout << "'World' not found" << std::endl;
}
```

The function will either return:

- The position (index) of the first occurrence of the substring, if found.
- `std::string::npos`, if the substring is not found.

0.7.1 `std::string::npos`

`std::string::npos` is a constant representing a value that indicates "not found" or "no position". It is typically used as the return value of string search functions like `find` when the substring is

not found.

0.8 `tolower` and `toupper` functions

To convert a character to lowercase or uppercase in C++, you can use the `tolower` and `toupper` functions from the `<cctype>` header.

```
#include <cctype>

// Then inside main()
char ch = 'A';
char lower = std::tolower(ch);
char upper = std::toupper(lower);
std::cout << lower << " " << upper << std::endl; // a A
```

0.9 Array in C++

To create a **fixed size** array in C++:

```
int arr[5]; // Array of 5 integers
```

To create a **dynamic array** in C++ use `vector` from the `<vector>` header:

```
#include <vector>

// Then inside main()
std::vector<int> vec; // Dynamic array (vector) of integers
```

0.9.1 Looping Over Array Values (Problem-L)

To loop over an array or vector, we can either use an **index-based loop** or a **range-based loop**.

```
vector<int> v = {10, 20, 30, 40, 50};

// Index-based loop
for (int i = 0; i < v.size(); i++) {
    cout << v[i] << " ";
}

// Range-based for
for (int x : v) {
    cout << x << " ";
}
```

In the range-based loop we can also use `auto` which allows the compiler to automatically deduce the type of the variable based on the value being assigned to it.

```
for (auto x : v) {
    cout << x << " ";
}
```

0.10 Ternary Operator

Just like JavaScript, C++ also supports the ternary operator `? :` for conditional expressions.

```
int a = 10, b = 20;
int maximum = (a > b) ? a : b; // If a > b,
cout << maximum; // 20
```

0.11 Precision

By default, C++ outputs floating-point numbers with 6 significant digits (including numbers before the decimal point).

We can control the precision of floating-point output in C++ using `setprecision` function from `<iomanip>` header.

```
float x = 1.0f;
double pi = 3.1415926535;
double large_num = 12345.6789;
double larger_num = 123456.789;

// Default precision: usually 6 significant digits
cout << "x / 3: " << x / 3 << endl;           // 0.333333
cout << "pi: " << pi << endl;                 // 3.14159
cout << "large_num: " << large_num << endl;   // 12345.7
cout << "larger_num: " << larger_num << endl; // 123457
```

Using `setprecision(n)` controls the total number of significant digits displayed (including those before the decimal point). The output may switch to scientific notation for very large or small numbers.

```
// Using setprecision(3): shows total significant digits
cout << setprecision(3);
cout << "pi: " << pi << endl;        // 3.14
cout << "large_num: " << large_num << endl; // 1.23e+04 (scientific notation)
```

Using `fixed` with `setprecision(n)` makes `setprecision` starts counting number of digits only after the decimal point.

```
// Using fixed and setprecision(3): shows 3 digits after decimal point
cout << fixed << setprecision(3);
cout << "pi: " << pi << endl;        // 3.142 (rounded)
cout << "large_num: " << large_num << endl; // 12345.679 (rounded)
```

To revert to default behavior, use `defaultfloat`:

```
cout << defaultfloat;
cout << "Back to default (pi): " << pi << endl; // 3.14159
```

> Note: When using `fixed` with `setprecision(n)`, if a floating-point number has fewer than `n` digits after the decimal point, the output will be padded with trailing zeros to reach exactly `n` decimal places.

```
float x = 1.234;
cout << fixed << setprecision(6) << x; // 1.234000
```

0.12 Data Types

Be extra careful with variable data types and their limits. This caused multiple issues in Theatre Square Problem in the sheet.

Type	Size (bytes)	Range (Approx)
`bool`	1	`true` / `false`
`char`	1	`-128` to `127`
`unsigned char`	1	`0` to `255`
`short`	2	`-32,768` to `32,767`
`unsigned short`	2	`0` to `65,535`
`int`	4	`-2,147,483,648` to `2,147,483,647`
`unsigned int`	4	`0` to `4,294,967,295`
`long`	8	`-9.22e18` to `9.22e18`
`unsigned long`	8	`0` to `1.84e19`
`long long`	8	`-9.22e18` to `9.22e18`
`unsigned long long`	8	`0` to `1.84e19`
`float`	4	~±3.4e38 (**7 digits precision**)
`double`	8	~±1.7e308 (**15 digits precision**)
`long double`	16	~±1.2e4932 (**18–19 digits precision**)