# C++ I/O

In this lesson, we'll discuss how to handle input and output in C++ code.

**We'll cover the following**



## I/O in contests #

In competitions, the input for your code has to read from standard input (*most of the time*) and you print the output to standout output.

In `C++`, two most common choices to do is

- **iosteams** - `cin` / `cout` **(NEW)**
- **stdio** - `scanf` / `printf` **(C style)**

Though it seems inconsequential, handling I/O correctly can make your solution *milliseconds* or even *a few seconds*. This becomes relevant when the input is huge (for example, reading $10^6$ integers).

Before moving on to I/O optimization, let's quickly see how to use each I/O function.

---

## Stdio #

**Include statement**: `#include <cstdio>`

`scanf` and `printf` need to know the type of the variable you are reading/printing, denoted by using a format specifier (`%`).

Here are the most common ones, for a complete list you can go here

```
scanf("%d", &x); // read int
scanf("%lld", &x); // read long long int
scanf("%s", &s); // read string
scanf("%f", &d); // read float
```

Similarly,

```
printf("%d", &x); // print int
printf("%lld", &x); // print long long int
printf("%s", &s); // print string
printf("%f", &d); // print float
```

Two space separated integers can also be read in a single `scanf` statement

```
scanf("%d %d", &a, &b);
```

---

# IOStream #

**Include statement**: `#include <iostream>`

Unlike `cstdio` functions, it's not required to define the type of the variable. I/O becomes much simpler as `x` can be any data type here.

```
cin>>x; // read x
cout<<x; // print x
```

# Synchronization #

In a single program, `scanf` and `cin` can be interleaved to read from input because synchronization is on by default.

Without going too much into the language architecture, this synchronization comes at a cost. Turning this off will cause unexpected bugs when `scanf` and `cin` and used together for the same stream but will speed up cin.

Add this statement at the start of the program:

```
ios_base::sync_with_stdio(false);
```

**Note**: With synchronization turned off, only use `cin` to read input.

Another option is to not use `iostreams` at all. `scanf` is fast enough and should almost never cause issues.

---

We'll kick off the next chapter with algorithm analysis.