Input and Output Functions

Apart from 'cin' and 'cout', there are many other functions we can use to perform input/output operations.

WE'LL COVER THE FOLLOWING

Input

You can read in C++ in two way from the input stream: Formatted with the extractor `>>` and unformatted with explicit methods.

Formatted Input

The extraction operator `>>`

- is predefined for all built-in types and strings,
- can be implemented for user-defined data types,
- can be configured by format specifiers.

std::cin ignores by default leading whitespace

```
#include <iostream>
//...
int a, b;
std::cout << "Two natural numbers: " << std::endl;
std::cin >> a >> b; // < 2000 11>
std::cout << "a: " << a << " b: " << b;
```

Unformatted Input

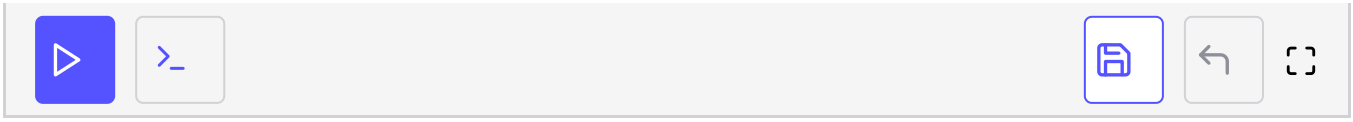
There are many methods for the unformatted input from an input stream `is`.

Method	Description
`is.get(ch)`	Reads one character into `ch`.
`is.get(buf, num)`	Reads at most `num` characters into the buffer `buf`.
`is.getline(buf, num[, delim])`	Reads at most `num` characters into the buffer `buf`. Uses optionally the line-delimiter `delim` (default `\n`).
`is.gcount()`	Returns the number of characters that were last extracted from `is` by an unformatted operation.
`is.ignore(streamsize sz= 1, int delim= end-of-file)`	Ignores `sz` characters until `delim`.
`is.peek()`	Gets one characters from `is` without consuming it.
`is.unget()`	Pushes the last read character back to `is`.
`is.putback(ch)`	Pushes the character `ch` onto the stream `is`.

Unformatted input from an input stream

> 🔍 **std::string has a getline function**
> The `getline` function of std::string has a big advantage above the `getline` function of the `istream`. The std::string automatically takes care of its memory. On the contrary, you have to reserve the memory for the buffer `buf` in the `is.get(buf, num)` function.

```
#include <iostream>
#include <string>

int main(){

  std::cout << std::endl;

  std::string line;
  std::cout << "Write a line: " << std::endl;
  std::getline(std::cin, line);
  std::cout << line << std::endl;

  std::cout << std::endl;

  std::cout << "Write numbers, separated by;" << std::endl;
  while ( std::getline(std::cin, line, ';') ) {
    std::cout << line << std::endl;
  }

  std::cout << std::endl;

}
```



Unformatted input

Output

You can push characters with the insert operator `<<` onto the output stream.

The insert operator `<<`

- is predefined for all built-in types and strings,
- can be implemented for user-defined data types,
- can be adjusted by format specifiers.