Input and Output

Strings are well-known for their use in input and output of data. Let's look at how this is done.

A string can read from an input stream via `>>` and write to an output stream via `<<`.

The global function `getline` empowers you to read from an input stream line by line until the *end-of-file* character.

There are four variations of the `getline` function available. The first two arguments are the input stream `is` and the string `line` holding the line read. Optionally you can specify a special line separator. The function returns by reference the input stream.

```
istream& getline (istream& is, string& line, char delim);
istream& getline (istream&& is, string& line, char delim);
istream& getline (istream& is, string& line);
istream& getline (istream&& is, string& line);
```

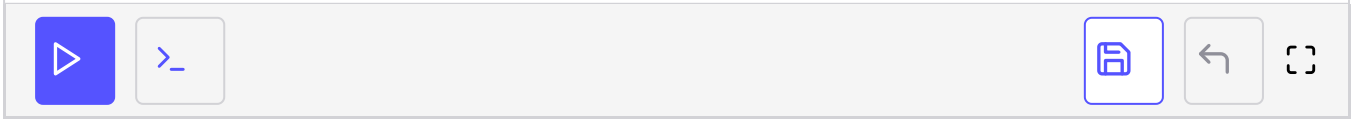
`getline` consumes the whole line including empty spaces. Only the line separator is ignored. The function needs the header `<string>`.

main.cpp

string.txt

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

std::vector<std::string> readFromFile(const char* fileName){

  std::ifstream file(fileName);

  if ( !file ){
    std::cerr << "Could not open the file " << fileName << ".";
    exit(EXIT_FAILURE);
  }
```

```
  std::vector<std::string> lines;
  std::string line;

  while ( getline(file , line) ) lines.push_back(line);

  return lines;

}

int main(){

  std::cout << std::endl;

  std::string fileName;
  std::cout << "Your filename: ";
  std::cin >> fileName;

  std::vector<std::string> lines=readFromFile(fileName.c_str());

  int num{0};
  for ( auto line: lines ) std::cout << ++num << ": " <<  line <<  std::endl;

  std::cout << std::endl;

}
```



The program displays the lines of an arbitrary file including their line number. The expression `std::cin >> fileName` reads the file name. The function `readFromFile` reads with `getline` all file lines and pushes them onto the vector.