# 8. InputOutput

June 21, 2023

## 1 C++ Input and Output

### 1.1 1. IO streams

#### 1.1.1 1.1 Formatted output to screen <iostream>

The standard input/output stream is part of the <iostream> header file and the standard namespace std. cout is the standard output stream cin is the standard input stream endl flushes the stream

Simplest programm

[ ]:
```
#include <iostream>
int main(){
std::cout << "Hello World!" << std::endl;
return 0;
}
```

Here is how a compilation of these lines looks in the terminal:

$ g++ -Wall -std=c++11 -o hello hello.cpp $ ./hello Hello World!

Works with all built in types

[ ]:
```
#include <iostream>
int a = 3;
double b = 3.1415;
double *c = &b; //pointer
char s[] = "Hello!"; //array of characters
std::cout << a << std::endl;
std::cout << b << std::endl;
std::cout << c << std::endl;
std::cout << s << std::endl;
```

Some more information here http://www.cplusplus.com/reference/iostream/

The output stream can be formatted, for example to print numbers in scientific notation using std::scientific, in a hexadecimal form std::hex, or boolean std::boolalpha. You can find more formats at the link below. (These also work with file streams!)

[ ]:
```
#include <iostream>
double a = 3.1415;
```

```
std::cout << std::scientific << a << std::endl;
std::cout << std::fixed << a << std::endl;
```

More here http://www.cplusplus.com/reference/ios/ (e.g. std::hex)

#### 1.1.2 1.2 Formatted Input from stdin <iostream>

[ ]:
```
#include <iostream>
int a;
double b;
std::cout << "Enter an integer: " << std::endl;
std::cin >> a;
std::cout << "Enter a double: " << std::endl;
std::cin >> b;
std::cout << a << " " << b << std::endl;
```

---

## 1.2 2. File Streams <fstream>

There are three data types for files stream:

ofstream is the output file stream data type. It is used to create and write to files.

ifstream is the input file stream data type. It is used to read data from files.

fstream is the generic file strea, data types, used for both writing and reading.

The standard function open(), which is a member of all file streams above, defines the mode in which the files is to be opened. void open(const char *filename, ios::openmode mode);

ios:app – append mode

ios::ate – opens a file for output and moved the read/write control to the end of the file

ios::in – reading mode

ios::out – writing mode

ios::trunc – if the file exists, it's will be truncated (dleted) before opening

For example ofstream outfile; outfile.open("file.dat", ios::out | ios::trunc ); will open a file for writing and erase its contents, if the files already exists. fstream afile; afile.open("file.dat", ios::out | ios::in ); will open a file for reading and for writing.

### 1.2.1 2.1 Output file stream example

[ ]:
```
#include <iostream>
#include <fstream>
std::ofstream myfile;
myfile.open("text.txt",std::ios::trunc);
for(auto i = 0; i < 5; i++)
    myfile << i << " " << (double)i + 0.5 << "\n"; // notice the type casting
```

```
myfile.close(); //usually a good idea :)
```

### 1.2.2 2.2 Input file stream example

For Input streams, perform error handling (e.g. check is file exists)

[ ]:
```
#include <iostream>
#include <fstream>
std::ifstream myfile("text.txt");
if(!myfile) std::cout << "could not open file" << std::endl; // check if file␣
 ↪exists
double a; double b;
while(myfile >> a >> b)
    std::cout << a << " " << b << std::endl;
myfile.close();
```

## 1.3 3. Strings

### 1.3.1 3.1 Character strings std::string

Strings std::string are objects that represent sequences of characters. The string class has constructors and destructors, as well as other memebr functions. Here are some examples:

size() returns the length of the string

clear() clears the contents of the string

begin() returns a the iterator to the beginning of the string

empty() test if the sring is empty

at() access am element of the string

c_str gets the c string equivalent

For more iterators, modifiers, and string operations, have a look at https://cplusplus.com/reference/string/string/

[ ]:
```
#include <string>
#include <iostream>
std::string s1 = "Hello";
std::string s2 = std::string("Hello");
std::string s3(s1);
std::string s4 = s1;
std::string s5;
s5 = s1; // assignment
std::cout << s1 << std::endl;
std::cout << s2 << std::endl;
std::cout << s3 << std::endl;
std::cout << s4 << std::endl;
std::cout << s5 << std::endl;
```

Strings are like character strings:

```
[ ]: #include <string>
     std::string s = "data.txt";
     const char *c = s.data(); // C string with '\0' at the end, marking the end of␣
      ↪the string
     c = s.c_str(); // C string with '\0' at the end, marking the end of the string
```

Have important extentions:

```
[ ]: #include <string>
     std::string s = "Hello";
     char c = s[3]; // l
     c = s.at(4); // o, bound check, indexing starts at 0
     int slen = s.length(); // 5
```

Indices start with 0, don't forget

Strings are also like std::vector, so that vector operations are also applicable: push_back(), clear(), erase(), append(), resize()

```
[ ]: #include <string>
     #include <iostream>
     std::string s = "This is a string";
     s.push_back('!');
     std::cout << s << std::endl;
     s.erase(3);
     std::cout << s << "\n" << s.length() << std::endl;
```

### 1.3.2 3.2 String arithmetics

Addition (concatenation)

```
[ ]: #include <string>
     #include <iostream>
     std::string s1 = "Hello ";
     std::string s2 = s1 + "World";
     std::cout << s2 << std::endl;
     s2 += " Bye!";
     std::cout << s2 << std::endl;
```

Comparison (lexicographcal)

```
[ ]: #include <string>
     #include <iostream>
     std::string foo = "alpha";
     std::string bar = "beta";
     if (foo==bar) std::cout << "foo == bar \n" << std::endl;
     if (foo!=bar) std::cout << "foo != bar \n" << std::endl;
     if (foo< bar) std::cout << "foo <  bar \n" << std::endl;
     if (foo> bar) std::cout << "foo >  bar \n" << std::endl;
     if (foo<=bar) std::cout << "foo <= bar \n" << std::endl;
     if (foo>=bar) std::cout << "foo >= bar \n" << std::endl;
```

### 1.3.3 3.3 Sub-strings and string manipulation

The member function string substr (size_t pos = 0, size_t len = npos) const; returns a newly constructed string object, that stars at character position pos and has length len characters.

[ ]:
```
#include <string>
#include <iostream>
std::string s = "This is a string";
std::string s1 = s.substr(10); // string
std::string s2 = s.substr(5,2); // is
std::cout << s1 << std::endl;
std::cout << s2 << std::endl;
```

String conversion to/from numbers with std::to_string C++11 (std::to_chars and std::from_chars #include <charconv> C++17, more features in C++23) Convert number 123 to a std::string:

[ ]:
```
#include <string>
#include <iostream>
const int n = 123;
std::string s = std::to_string(n);
std::cout << s << std::endl;
```

[ ]:
```
#include <string>
#include <iostream>
#include <charconv>
const int n = 123;
std::string s;
s.resize(3);
const auto res = std::to_chars(s.data(), s.data() + s.size(), n);
```

Convert a std::string with value “123” to an integer:

[ ]:
```
#include <string>
#include <iostream>
#include <charconv>
const std::string str{ "123" };
int n = 0;
const auto res = std::from_chars(s.data(),s.data() + s.size(), n);
```

### 1.3.4 3.4 IO related to strings (through streams)

The function istream& getline (istream& is, string& str); extract characters from the standard input is and saves them into the string str. This function has also been overloaded to take additional input parameters. std::getline() works with any input stream: istream, ifstream, istringstream For more information, please have a look at https://cplusplus.com/reference/string/string/getline/

[ ]:
```
#include <iostream>
#include <string>
std::string line = "";
do{
    if(line.length() > 0)
        std::cout << "you entered " << line << "\n";
    std::cout << "enter something: ";
} while(std::getline(std::cin,line));
std::cout << std::endl;
```

Let’s now use std::getline() to read from out file text.txt

[ ]:
```
#include <iostream>
#include <string>
#include <fstream>
std::string line = "";
std::ifstream myfile("text.txt");
do{
    if(line.length() > 0)
        std::cout << "you entered " << line << "\n";
} while(std::getline(myfile,line));
std::cout << std::endl;
myfile.close();
```

## 2 4. Stringstream <sstream>

std::stringstream is a class, that allows you to manupulate strings, as if they were input or output streams (using operator« and operator»). It is included in the header <sstream>. Aside from the generic std::stringstream, there is also an output string stream std::ostringstream and an input sting stream std::istringstream.

#### 2.0.1 4.1 Output string stream std::ostringstream

You can insert formated output using the «operator

[ ]:
```
#include <iostream>
#include <string>
#include <sstream>
int ival = 5;
double dval = 1.23;
bool b = true;
std::string s = "some text";

std::ostringstream oss;

oss << "i = " << ival << " d = " << dval <<
    " b = " << std::boolalpha << b <<
    " s = " << s << std::endl;
```

```
std::cout << oss.str() << std::endl;
```

### 2.0.2 4.2 Input string stream std::istringstream

You can read formated data using the »operator

[ ]:
```
#include <iostream>
#include <sstream>
#include <string>
std::string stringvalues = "125 320 512 750 333";
std::istringstream iss(stringvalues);

for (auto n=0; n<5; n++)
{
    int val;
    iss >> val;
    std::cout << val << '\n';
}
```

Let's now use the input stringstream to read data from out file "text.txt"

[ ]:
```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
std::string line = "";

std::ifstream myfile("text.txt");
while(std::getline(myfile,line)){
    std::istringstream iss(line);
    int a; double b;
    iss >> a >> b;
    std::cout << a << " " << b << std::endl;
}
myfile.close();
```

[ ]:
```
```