File Streams

Now, we shall learn how to communicate with files using C++.

WE'LL COVER THE FOLLOWING

- Random Access Random access enables you to set the file position pointer arbitrarily.

File streams enable you to work with files. They need the header `<fstream>`. The file streams automatically manage the lifetime of their file.

Whether you use a file stream for input or output or with the character type `char` or `wchar_t` there are various file stream classes:

Class	Use
`std::ifstream` and `std::wifstream`	File stream for the input of data of type `char` and `wchar_t`.
`std::ofstream` and `std::wofstream`	File stream for the output of data of type `char` and `wchar_t`
`std::fstream` and `std::wfstream`	File stream for the input and output of data of type `char` and `wchar_t`.
`std::filebuf` and `std::wfilebuf`	Data buffer of type `char` and `wchar_t`.

⚠ **Set the file position pointer**

> File streams used for reading and writing have to set the file position pointer after the contests change.

Flags enable you to set the opening mode of a file stream.

Flag	Description
`std::ios::in`	Opens the file stream for reading (default for `std::ifstream` and `std::wifstream`).
`std::ios::out`	Opens the file stream for writing (default for `std::ofstream` and `std::wofstream`).
`std::ios::app`	Appends the character to the end of the file stream.
`std::ios::ate`	Sets the initial position of the file position pointer on the end of the file stream.
`std::ios::trunc`	Deletes the original file.
`std::ios::binary`	Suppresses the interpretation of an escape sequence in the file stream.

Flags for the opening of a file stream

It's quite easy to copy the file named `in` to the file named `out` with the file buffer `in.rdbuf()`. The error handling is missing in this short example.

```
#include <fstream>
...
std::ifstream in("inFile.txt");
std::ofstream out("outFile.txt");
out << in.rdbuf();
```

If you combine the C++ flags, you can compare the C++ and C modes to open a file.

C++ mode	Description	C mode
`std::ios::in`	Reads the file.	`"r"`
`std::ios::out`	Writes the file.	`"w"`
`std::ios::out|std::ios::app`	Appends to the file.	`"a"`
`std::ios::in|std::ios::out`	Reads and writes the file.	`"r+"`
`std::ios::in|std::ios::out|std::ios::trunc`	Writes and reads the file.	`"w+"`

Opening of a file with C++ and C

The file has to exist with the mode `"r"` and `"r+"`. In contrary, the file is be created with `"a"` and `"w+"`. The file is overwritten with `"w"`.

You can explicitly manage the lifetime of a file stream.

Flag	Description
`infile.open(name)`	Opens the file `name` for reading.
`infile.open(name, flags)`	Opens the file `name` with the flags `flags` for reading.
`infile.close()`	Closes the file `name`.

`infile.is_open()`	Checks if the file is open.

Managing the lifetime of a file stream

Random Access Random access enables you to set the file position pointer arbitrarily.

When a file stream is constructed, the files position pointer points to the beginning of the file. You can adjust the position with the methods of the file stream `file`.

Method	Description
`file.tellg()`	Returns the read position of `file`.
`file.tellp()`	Returns the write position of `file`.
`file.seekg(pos)`	Sets the read position of `file` to `pos`.
`file.seekp(pos)`	Sets the write position of `file` to `pos`.
`file.seekg(off, rpos)`	Sets the read position of `file` to the offset `off` relative to `rpos`.
`file.seekp(off, rpos)`	Sets the write position of `file` to the offset `off` relative to `rpos`.

Navigate in a file stream

`off` has to be a number. `rpos` can have three values:

`rpos` value	Description
`std::ios::beg`	Position at the beginning of the file.

`std::ios::cur`	Position at the current position
`std::ios::end`	Position at the end of the file.

> ⚠ **Respect the file boundaries**
> If you randomly access a file, the C++ runtime does not check the file boundaries. Reading or writing data outside the boundaries is *undefined behaviour*.

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

int writeFile(const std::string name){

  std::ofstream outFile(name);

  if (!outFile){
    std::cerr << "Could not open file " << name << std::endl;
    exit(1);
  }

  for ( unsigned int i=0; i < 10; ++i){
    outFile << i << "        0123456789" << std::endl;
  }
}



int main(){

  std::cout << std::endl;

  std::string random{"random.txt"};

  writeFile(random);

  std::ifstream inFile(random);

  if (!inFile){
    std::cerr << "Could not open file " << random << std::endl;
    exit(1);
  }

  std::string line;

  std::cout << "The whole file : " << std::endl;
  std::cout << inFile.rdbuf();
  std::cout <<  "inFile.tellg(): " << inFile.tellg()  << std::endl;

  std::cout <<  std::endl;
  inFile.seekg(0);
  inFile.seekg(0, std::ios::beg);   // redundant
  getline(inFile, line);
  std::cout << line << std::endl;

  inFile.seekg(20, std::ios::cur);
  getline(inFile, line);
  std::cout << line << std::endl;

  inFile.seekg(-20, std::ios::end);
  getline(inFile, line);
  std::cout << line << std::endl;

  std::cout << std::endl;

}
```

Random access