Streams

Terms

Binary files
Buffer
File streams
Input streams
Output streams
Streams
String streams
Text files

Summary

- A *stream* is an abstraction for a data source or destination. Using streams, we can read data or write it to a variety of places (eg terminal, files, network, etc) in the same way.
- In the C++ STL, we have many stream classes for different purposes. All these classes inherit their functionality from **ios_base**.
- A *buffer* is a temporary storage in memory used for reading or writing data to streams.
- If an error occurs while reading data from a stream, the invalid data stays in the buffer and will be used for subsequent reads. In such situations, first we have to put the stream into a clean state using the **clear()** method. Then, we should clear the data in the buffer using the **ignore()** method.
- In the C++ STL, we have three stream classes for working with files. (**ifstream** for reading from files, **ofstream** for writing to files, and **fstream** for reading and writing to files).
- *Binary files* store data the same way it is stored in memory. They are more efficient for storing large amount of numeric data but they're not human readable.
- Using *string streams* we can convert data to a string or vice versa.


```cpp
// Writing to a stream
cout << "Hello World";

// Reading from a stream
int number;
cin >> number;

// Handling read errors
if (cin.fail()) {
    cout << "Error";
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

// Writing to a text file
ofstream file;
file.open("file.txt", ios::app);
if (file.is_open()) {
    file << "Hello World";
    file.close();
}
```



```cpp
// Reading from a text file
ifstream file;
file.open("file.txt");
if (file.is_open()) {
    string str;
    while(!file.eof()) {
        getline(file, str);
    }
    file.close();
}


// Writing to a binary file
int numbers[3] = { 1, 2, 3 };

ofstream file;
file.open("file.bin", ios::app | ios::binary);
if (file.is_open()) {
    file.write(
        reinterpret_cast<char*>(&numbers),
        sizeof(numbers));
    file.close();
}
```



```
// Reading from a binary file
ifstream file;
file.open("file.bin", ios::binary);
if (file.is_open()) {
    int number;
    while (file.read(
        reinterpret_cast<char*>(&number),
        sizeof(number))) {
        cout << number << endl;
    }
    file.close();
}
```

