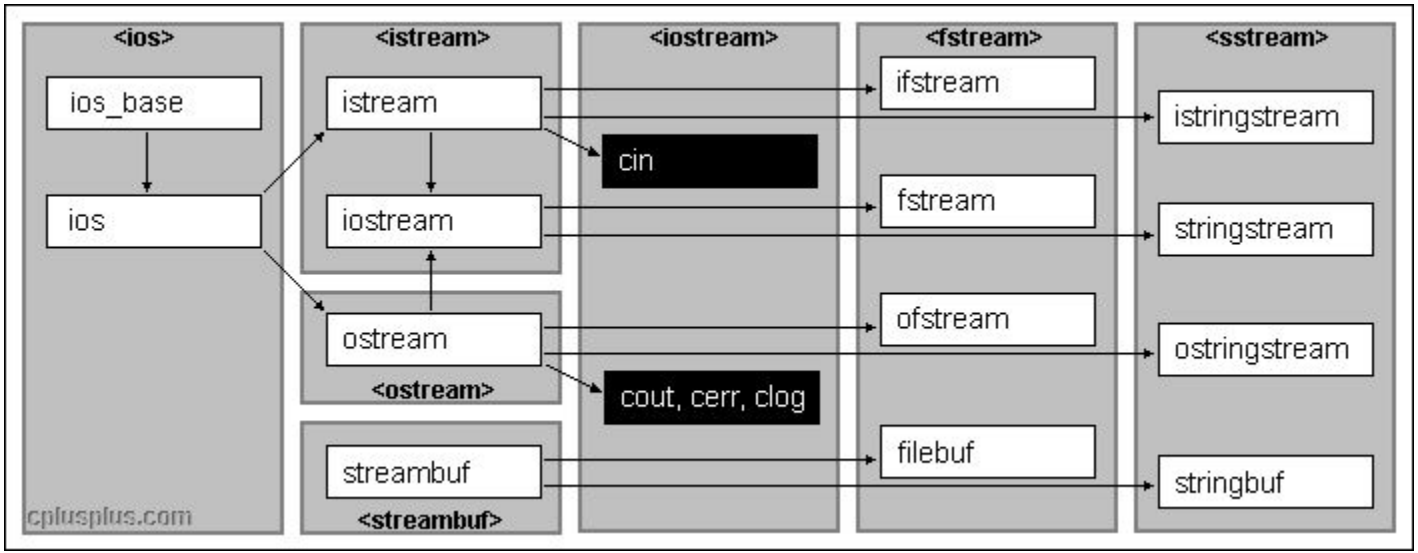
Object-oriented programming CS10003

Lecturer: Do Nguyen Kha

Contents

- C++ Input/Output: Stream
- C++ File Programming
- Team Project: Milestone 2

C++ Input/Output: Stream



https://cplusplus.com/reference/iolibrary/

C++ Input/Output: Stream

The `iostream` library is an object-oriented library that provides input and output functionality using streams.

A stream is an abstraction that represents a device on which input and output operations are performed. A stream can basically be represented as a source or destination of characters of indefinite length.

If using with Unicode, add prefix `w` to the classes, such as `wios` or `wistream`

C++ Input/Output: Stream

- `ios`: support both input and output classes
- `istream`: inherited from `ios`, support reading from input object
- `ostream`: inherited from `ios`, support writing to output object
- `iostream`: inherited from `istream` and `ostream`, support both reading and writing

File Stream

- `ifstream`: inherited from `istream`, support reading from file
- `ofstream`: inherited from `ostream`, support writing to file
- `fstream`: inherited from `iostream`, support both reading from and writing to file

String Stream

- `istringstream`: inherited from `istream`, support reading and getting data from input string object
- `ostringstream`: inherited from `ostream`, support writing and printing to output string object
- `stringstream`: inherited from `iostream`, support all operations of string object

Standard Input/Output Stream

Operator << outputs data to the object of ostream or its derived classes

Operator >> inputs/receives data from the object of istream or its derived classes

C++ I/O defines 4 objects:

- `cin`: input data from input-device (default keyboard)
- `cout`, `cerr`, `clog`: output data to output-device (default console screen)

Stream Operator Overloading

```
#include <iostream>
using namespace std;
class Point3D {
 int x, y, z;
 public:
   Point3D(int a, int b, int c) { x = a; y = b; z = c; }
   friend ostream& operator<<(ostream& os, Point3D &P);
};
ostream& operator<<(ostream& os, Point3D &P){
 os << "(" << P.x << ", " << P.y << ", " << P.z << ")";
 return os;
}
void main(){
 Point3D Q(3, 2, 5);
 cout << "Q: " << Q << endl;
}
```

cin.fail()

```
#include <iostream>
using namespace std;
void main(){
  int x;
  cin>>x;
  if(cin.fail()){
    cout << "Error…" << endl;
  }
}
```

Stream State Flag

- `eof()`: Returns true if the eofbit is set (the stream is at the end of a file)
- `fail()`: Returns true if the failbit is set (a non-fatal error occurred)
- `bad()`: Returns true if the badbit is set (a fatal error occurred)
- `good()`: Returns true if everything is ok

C++ File Programming

Some operations in file: open a file, read/write in a file and close a file

Operations	Methods
Open file	Using constructors of ofstream, ifstream, fstream Using `open()` method
Read file	Using extraction operator >> Using `get()`, `getline()` or `read()` methods
Write file	Using insertion operator << Using `put()` or `write()` methods
Close file	Automatically closed when `ofstream`, `ifstream` or `fstream` object is destroyed Using `close()` method

C++ File Programming

Operations	Methods
Reading position	Methods of class `istream` (also `ifstream`) ● `tellg()` : return current reading-position ● `seekg()` : move reading-position to new position
Writing position	Methods of class `ostream` (also `ofstream`) ● `tellp()` : return current writing-position ● `seekp()` : move writing-position to new position
Check end of file	Methods of classes `ifstream`, `ofstream` and `fstream` ● `eof()` : check if reading/writing-position moves to end of file or not

C++ File Programming

Reading file technique: perform following steps to read a file

- Open the file
- Check if opening is success or fail
- Transfer the content from a file to output-device
- When transferring, check if we come to end of file.
- Close the file

C++ File Programming

To read a file:

- create an instance of “ifstream” class
- use `void open(const char* filename, ios::openmode mode = ios::in)`
  - `filename`: the name of file needed to open
  - `mode`: mode of opening
    - `ios::in`: open to read
    - `ios::binary`: open binary file
    - `ios::nocreate`: error when file not available

C++ File Programming

```
#include <fstream>
using namespace std;
void main(){
  ifstream inputFile;
  inputFile.open("data.txt");
  // or using constructor
  ifstream inputFile("data.txt");
  // reading data…
}
```

C++ File Programming

Read a character or a byte

```
int get(); or istream& get(char& c);
```

Reading a *n*-character string, stopped when encountering termination-character stored in `delim` variable (Default termination-character is `\n`)

```
istream& get(char* s, streamsize n);
istream& get(char* s, streamsize n, char delim);
istream& get(streambuf& sb);
istream& get(streambuf& sb, char delim);
```

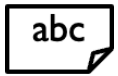
Reading a n-character line, termination-character is stored in `delim` variable (default `\n`)

```
istream& getline(char* s, int n);
istream& getline(char* s, int n, char delim);
```

Reading a binary file with maximum `nbufferSize` bytes

```
istream& read(char* buffer, int nbufferSize);
```

C++ File Programming




```
void main(){
 char a, b, c; fstream f("data.txt", ios::in);
 cout << f.tellg() << endl; // 0
 f >> a; // a
 cout << f.tellg() << endl; // 1
 f >> b; // b
 cout << f.tellg() << endl; // 2
 f.seekg(1);
 f >> c; // b
 cout << f.tellg() << endl; // 2
 cout << a << endl << b << endl << c << endl;
 // a
 // b
 // b
 f.close();
}
```

C++ File Programming

Writing file technique: perform following steps to write a file

- Open the file
- Check if opening is success or fail
- Transfer the content from input-device to output-device
- Close the file

C++ File Programming

To write a file:

- create an instance of “ofstream” class
- `use void open(const char* filename, ios::openmode mode = ios::in)`
  - `filename`: the name of the file to open
  - `mode`: mode of opening
    - `ios::app`: open to append at the end of file
    - `iso::ate`: move a writing-pointer to end of file
    - `ios::in`: if file is available, its contents won’t be deleted when opened
    - `ios::out`: open to write
    - `ios::trunc`: if file is available, its contents will be deleted when opened
    - `ios::binary`: open to write binary file
    - `ios::nocreate`: error when file not available
    - `ios::noreplace`: error when file available

C++ File Programming

```
#include <fstream>
#include <iostream>
using namespace std;
void main(){
  ofstream f("data.txt", ios::app);
  char* str = "Hello World!!!";
  while(*str){
    f.put(*str);
    str++;
  }
  f.close();
}
```

Team Project: Milestone 2

- Submission deadline: **14/12/2023**
- Add command line argument for opening SVG file path
- Each group must post result showcase (https://drive.google.com/drive/folders/1hSOCsf1_E_yu7IPLExYMvW_IgJqSo-Mh?usp=drive_link) of 18 test cases to Facebook Group by this weekend (03/12/2023)