Secure Coding

CHAPTER 8

THE C++ INPUT/OUTPUT CLASS HIERARCHY

- 8.1 Overview
- 8.2 The High-Level Input/Output Classes
- 8.3 Manipulators
- 8.4 The File Input/Output Classes
- 8.5 The Character Stream Input/Output Classes

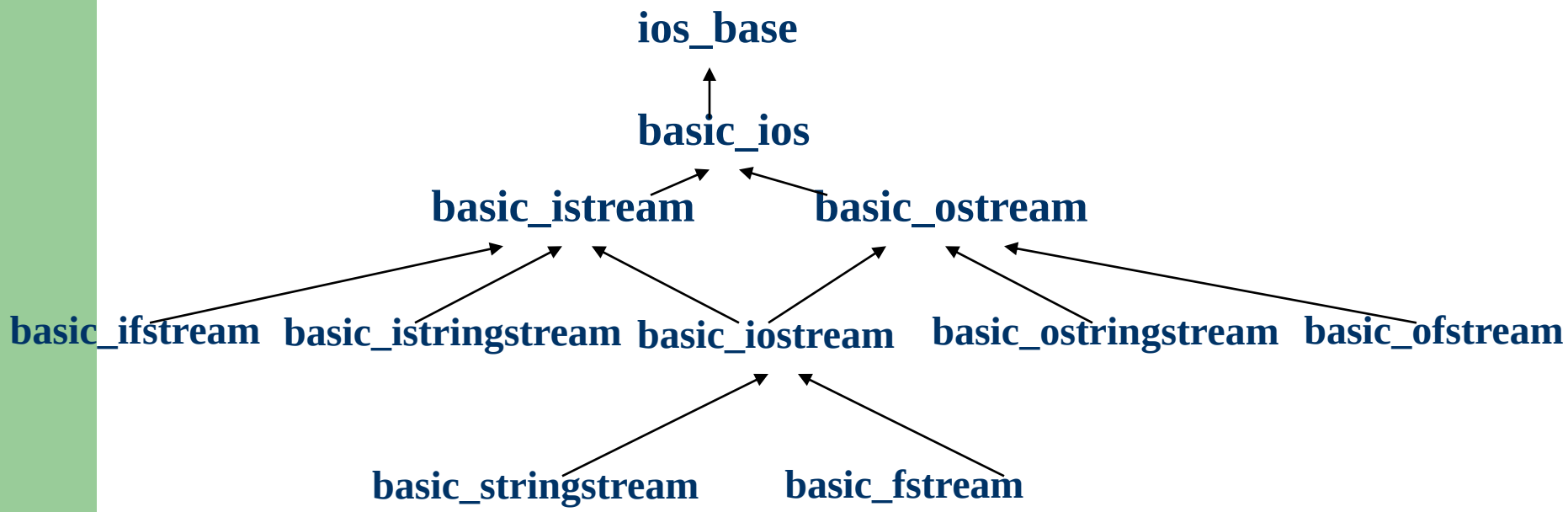
8.1 Overview

Input and output facilities are not part of C++ language but instead of furnished through a class library.

In C++ input and output, a central object is the stream, which is a sequence of bytes. There is a common base class for two derived stream classed basic_istream, an input stream class, and basic_ostream, an output stream class.

8.1 Overview

The standard input/output hierarch.



8.1 Overview

Several input/output headers and describes their purpose.

Header	Partial Description
iosfwd	Contains forward declarations
iostream	Declares cin, cout, etc.
ios	Declares ios_base and basic_ios
streambuf	Declares basic_streambuf
istream	Declares basic_istream
ostream	Declares basic_ostream
iostream	Declares basic_istream and basic_ostream
iomanip	Declares parameterized manipulators
sstream	Declares basic_stringbuf and the stringstream classes
fstream	Declares basic_filebuf and the fstream classes

8.2 The High-Level Input/Output Classes

In C++, we use the classes basic_istream, basic_ostream, and basic_iostream to get a high-level interface to input and output.

Header	Partial Description
iostream	Declares basic_istream and basic_iostream
ostream	Declares basic_ostream
istream	Declares basic_istream

8.2 The High-Level Input/Output Classes basic_istream

- basic_istream

cin>>val

cin reads the stream, without white space, and save it to the val.

Example:

```
int a; char c; char ca[20];
cin>>a>>c>>ca;
```

8.2 The High-Level Input/Output Classes basic_istream

Example:

```
#include <iostream.h>
void main( )
{
   int n;        cin>>n;
   char ch;    cin>>ch;
   float pi;     cin>>pi;
   char str[20];     cin>>str;
   cout<<"n="<<n<<endl;
   cout<<"ch="<<ch<<endl;
   cout<<"pi="<<pi<<endl;
   cout<<"string="<<str<<endl;
}
```

Input:
"5 c 3.14159 hello world!"

Output:
n=5
ch=c
pi=3.14159
string=hello

8.2 The High-Level Input/Output Classes basic_istream

- **Operator Overloaded**

```
istream & operator >> ( signed char);
istream & operator >> (unsigned char);
istream & operator >> (short);
istream & operator >> (unsigned short);
istream & operator >> (int);
istream & operator >> (unsigned int);
istream & operator >> (long);
istream & operator >> (unsigned long);
istream & operator >> (float);
istream & operator >> (double);
istream & operator >> (long double);
```

8.2 The High-Level Input/Output Classes basic_istream

- **Methods**

A) get( );

When get( ) is invoked, the next character in the stream, white space or not, is returned.

Example:

```
int c;
while( ( c=cin.get( ) )!=EOF )
        cout<<c;
```

8.2 The High-Level Input/Output Classes basic_istream

B) getline(char_type* b, streamsize s, char_type d);

b: an array, into which to write character

s: an integer value that bounds the number of characters

d: an end-of-line marker

8.2 The High-Level Input/Output Classes basic_istream

Method getline reads characters until it

1. **Reaches end-of-file**
2. **Encounters an end-of-line marker**
3. **Stores s-1 characters**

Example:

```
#include   <iostream>

using namespace std;
void main (void)
{
    char city[80];
    char country[80];
    int  i;
    for (i = 0; i < 2; i++)
    {
        cin.getline(city,80,',');
        cin.get(country,80,'\n');
        cout << "City: " << city  << "   Country: " << country<< endl;


     }
}
```

Input:

Beijing,China

Shanghai,China

Output:

City: Beijing   Country: China

City:

Shanghai   Country: China

Example:

```
#include   <iostream>
#include   <limits>
using namespace std;
void main (void)
{
    char city[80];
    char country[80];
    int  i;
    for (i = 0; i < 2; i++)
    {
        cin.getline(city,80,',');
        cin.get(country,80,'\n');
        cout << "City: " << city  << "   Country: " << country<< endl;
        cin.clear( );
        cin.ignore( numeric_limits<streamsize>::max(), '\n' );
     }
}
```

Input:

```
Beijing,China
Shanghai,China
```

Output:

```
City: Beijing   Country: China
City: Shanghai   Country: China
```

8.2 The High-Level Input/Output Classes basic_istream

C) read(char_type* a, streamsize n);

Method read reads characters and stores them in the array a until n characters are read or end-of-file occurs.

8.2 The High-Level Input/Output Classes basic_ostream

- **basic_ostream**

cout<<val

cout puts the stream, in val, to the screen.

Example:

cout<<“Hello： ”<<123<<endl;

8.2 The High-Level Input/Output Classes basic_ostream

Example:

```
#include <iostream.h>
void main()
{
    float pi=3.14159;
    cout<<"pi=";
    cout<<pi;
    cout<<endl;
}
```

Output :
pi=3.14159

8.2 The High-Level Input/Output Classes basic_ostream

- **Operator Overloaded**

```
ostream & operator<< ( signed char);
ostream & operator<< (unsigned char);
ostream & operator<< (short);
ostream & operator<< (unsigned short);
ostream & operator<< (int);
ostream & operator<< (unsigned int);
ostream & operator<< (long);
ostream & operator<< (unsigned long);
ostream & operator<< (float);
ostream & operator<< (double);
ostream & operator<< (long double);
```

8.2 The High-Level Input/Output Classes basic_ostream

- **Methods**

A) put(char_type);

Method put writes the character passed to the output stream.

Example:

```
cout.put( c );
```

8.2 The High-Level Input/Output Classes basic_ostream

B) write(const char_type* a, streamsize m);

Method write writes m characters from the array a to the output stream.

8.2 The High-Level Input/Output Classes basic_ostream

Example:

```
#include <iostream.h>
#include<string.h>
void main( )
{
    char* pc="this is a test!";
    cout.put('A');
    cout.put('\n');
    cout.write(pc,strlen(pc));
 }
```

Output:

A

this is a test!

8.2 The High-Level Input/Output Classes basic_iostream

- basic_iostream

Class basic_iostream is publicly inherited from both basic_istream and basic_ostream.

8.2 The High-Level Input/Output Classes basic_iostream

```
Example:
class Complex
{
public:
    Complex( ){real=0.0;imag=0.0;}
    Complex(double r){real=r;imag=0.0;}
    Complex(double r, double i){real=r;imag=i;}
    Complex operator + (const Complex& c);
    Complex operator - (const Complex& c);
    Complex operator * (const Complex& c);
    Complex operator / (const Complex& c);
    friend ostream& operator <<(ostream& Out, Complex& x);
    friend istream& operator >>(istream& In, Complex& x);
protected:
    double real, imag;
};
```

8.2 The High-Level Input/Output Classes basic_iostream

```
ostream& operator<<(ostream& Out, Complex& x)
{
    Out<<x.real<<"+"<<x.imag<<"i"<<endl;
    return Out;
}
istream& operator>>(istream& In, Complex& x)
{
    In>>x.real>>x.imag;
    return In;
}
//……
```

8.2 The High-Level Input/Output Classes basic_iostream

```
void main()
{
    Complex a(1,2),b(3,4),c;
    c=a+b;
    cout<<c;
}
```

Output:

4+6i

8.3 Manipulators

A manipulators is a function that either directly or indirectly modifies a stream.

Header	Partial Description
iomanip	Declares parameterized manipulators

8.3 Manipulators

Several manipulators with arguments are predefined.

Manipulator	Acts On	Purpose
setBase(int n)	basic_ostream	Set integer base to n (0 means default)
setfill(char_type c)	basic_ostream	Set fill character to c
setprecision(int n)	basic_ostream	Set precision to n
setw(int n)	basic_ostream	Set field width to n
setiosflags(mask)	ios_base	Set specified format bits
resetiosflags(mask)	ios_base	Clear specified format bits

8.3 Manipulators

- **setBase(int)**

Example:

```
cout<<setbase(8)
        <<9<<endl;
```

Output:

```
   11
```

8.3 Manipulators

- setw(int)

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main( )
{
    double values[ ]={1.23,35.36,653.7,4358.24};
    char *names[ ]={"Zoot","Jimmy","Al","Stan"};
    for(int i=0;i<4;i++)
    cout<<setw(6)<<names[i]
        <<setw(10)<<values[i]
        <<endl;
}
```

```
Output:
  Zoot      1.23
 Jimmy     35.36
    Al     653.7
  Stan   4358.24
```

6 10

6 characters

8.3 Manipulators

- setfill(char)

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main( )
{
    double values[ ]={1.23,35.36,653.7,4358.24};
    for(int i=0; i<4; i++)
    {
        cout.width(10);
        cout<<setfill('*')<<values[i]<<'\n';
    }
}
```

Fill with '*'

Output:

```
******1.23
*****35.36
*****653.7
***4358.24
```

8.3 Manipulators

- setiosflags and rsetiosflags

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
  double values[]={1.23,35.36,653.7,4358.24};
  char *names[]={"Zoot","Jimmy","Al","Stan"};
  for(int i=0;i<4;i++)
   cout<<setiosflags(ios::left)     // flush left
       <<setw(6)<<names[i]
       <<resetiosflags(ios::left)   // remove flush left
       <<setw(10)<<values[i]
       <<endl;
}
```

Output:

```
Zoot        1.23
Jimmy      35.36
Al          653.7
Stan     4358.24
```

- **setprecision**

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
  double values[]={1.23,35.36,653.7,4358.24};
  char *names[]={"Zoot","Jimmy","Al","Stan"};
  cout<<setiosflags(ios::scientific);
  for(int i=0;i<4;i++)
   cout<<setiosflags(ios::left)
         <<setw(6)<<names[i]
         <<resetiosflags(ios::left)
         <<setw(10)<<setprecision(1)
         << values[i]<<endl;
}
```

Output:

```
Zoot    1.2e+000
Jimmy   3.5e+001
Al      6.5e+002
Stan    4.4e+003
```

Precision 1

8.4 The File Input/Output Classes

The classes basic_filebuf, basic_ofstream, basic_ifstream and basic_fstream are declared in fstream.

Header	Partial Description
fstream	Declares basic_filebuf and the fstream classes

There are two kinds of files: text and binary text.

8.4 The File Input/Output Classes basic_ofstream

- basic_ofstream

```
basic_ofstream( const char* filename,
                ios_base::openmode mode=io_base::out);
```

The constructor is used to associate a basic_ofstream object with the file filename, which is opened in mode mode.

The type openmode is a bitmask type. The default mode is output.

8.4 The File Input/Output Classes basic_ofstream

A)

```
ofstream outfile("outfile",iosmode);
```

B)

```
ofstream myFile;
myFile.open("filename",iosmode);
```

C)

```
ofstream* pmyFile = new ofstream;
pmyFile->open("filename",iosmode);
```

8.4 The File Input/Output Classes basic_ofstream

Example:

```
#include "fstream.h"
#include "iostream.h"
void main( )
{
    char buf[80];
    ofstream out("out.txt");

    while (cin.getline(buf,80,' '))
    {
        cout<<buf<<endl;
        out<<buf<<endl;
    }
}
```

Output file

Get data

8.4 The File Input/Output Classes basic_ifstream

- basic_ifstream

```
basic_ifstream( const char* filename,
                ios_base::openmode mode=ios_base::in);
```

The constructor is used to associate a basic_ifstream object with the file filename, which is opened in mode mode. The default mode is input.

8.4 The File Input/Output Classes basic_ifstream

A)

```
ifstream outfile("infile",iosmode);
```

B)

```
ifstream myFile;
myFile.open("filename",iosmode);
```

C)

```
ifstream* pmyFile = new ifstream;
pmyFile->open("filename",iosmode);
```

8.4 The File Input/Output Classes basic_ifstream

Example:

```
ifstream fin("data.in" );
if( !fin )
        cerr<<"Can't open data.in\n";
```

Example:

```
ifstream fin;
fin.open("data.in" );
```

8.4 The File Input/Output Classes basic_ifstream

Example:

```
#include "fstream.h "
#include "iostream.h"
void main( )
{
    int line=1;
    char buf[80];
    ifstream in("in.txt");
    ofstream out("out.txt");

    while (in.getline(buf,80))
    {
        cout<<line++<<":"<<buf<<endl;
        out<<buf<<endl;
    }
}
```

Input file

Output file

Read data

8.4 The File Input/Output Classes basic_fstream

- basic_fstream

basic_fstream(const char* filename,

ios_base::openmode mode=ios_base::in

|ios_base::out);

The constructor is used to associate a basic_fstream object with the file filename, which is opened in mode mode. The default mode is input and output.

8.4 The File Input/Output Classes basic_fstream

A)

```
fstream outfile("file",iosmode);
```

B)

```
fstream myFile;
myFile.open("filename",iosmode);
```

C)

```
fstream* pmyFile = new fstream;
pmyFile->open("filename",iosmode);
```

8.4 The File Input/Output Classes basic_fstream

Example:

fstream finout(“data.txt”);

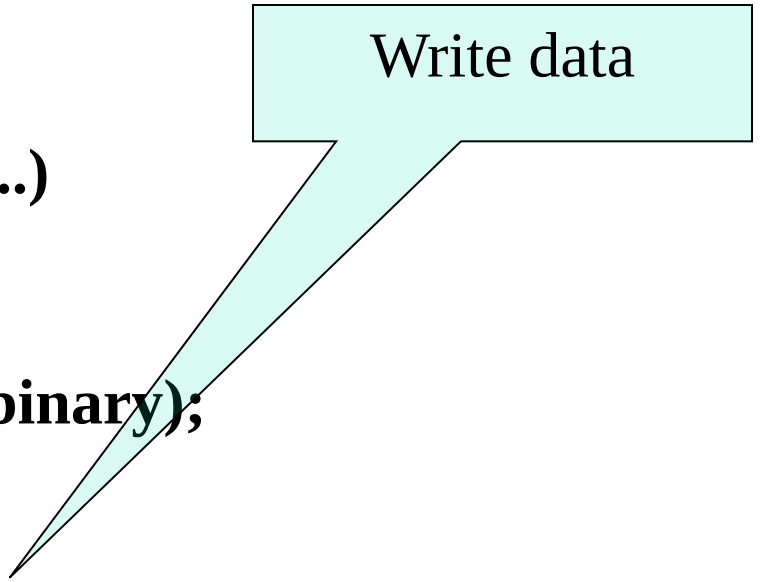
In this case the file data.txt is for both input and output.

Binio.cpp

```
#include <fstream>          //for file streams
#include <iostream>
using namespace std;
const int MAX = 100;        //size of buffer
int buff[MAX];              //buffer for integers
int main( )
{
   //fill buffer with data
   for(int j=0; j<MAX; j++)
        buff[j] = j;          //(0, 1, 2, ...)

   //create output stream
   ofstream os("edata.txt", ios::binary);

   //write to it
   os.write( (char*)buff, MAX*sizeof(int) );
   os.close( );           //must close it
```

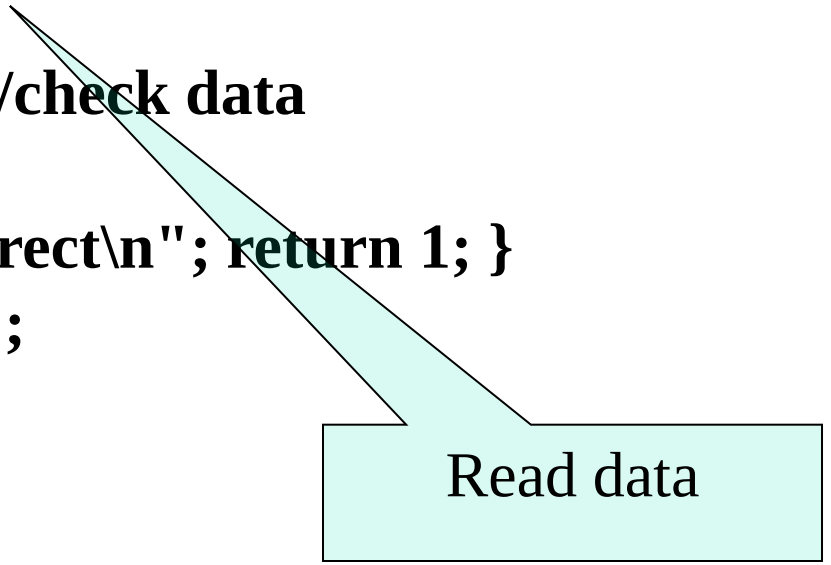


```
  for(j=0; j<MAX; j++)     //erase buffer
       buff[j] = 0;

  //create input stream
  ifstream is("edata.txt", ios::binary);

  //read from it
  is.read((char*)(buff), MAX*sizeof(int) );

  for(j=0; j<MAX; j++)       //check data
     if( buff[j] != j )
        { cerr << "Data is incorrect\n"; return 1; }
  cout << "Data is correct\n";
  return 0;
}
```



Output: data.txt

```
  ! " # $ % & ' ( ) * + , - .
/ 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N
O P Q R S T U V W X Y Z [ \ ] ^
_ ` a b c
```

8.4 The File Input/Output Classes Methods

- **Methods**

basic_ofstream:

1 open

2 close

3 tellp: Gets the value for the stream's put pointer.

8.4 The File Input/Output Classes basic_fstream

4 seekp(streampos pos);

5 seekp(streamoff off, ios::seek_dir dir);

pos: The new position value; streampos is a typedef equivalent to long.

off: The new offset value; streamoff is a typedef equivalent to long.

dir: The seek direction specified by the enumerated type ios:: seek_dir, with values including:

ios:: beg Seek from the beginning of the stream.

ios:: cur Seek from the current position in the stream.

ios:: end Seek from the end of the stream.

8.4 The File Input/Output Classes basic_fstream

basic_ifstream:

1 open

2 close

3 get

4 getline

5 read

6 tellg: Gets the value for the stream's get pointer.

8.4 The File Input/Output Classes basic_fstream

7 seekg(streampos pos)

8 seekg(streamoff off, ios::seek_dir dir)

pos: The new position value; streampos is a typedef equivalent to long.

off: The new offset value; streamoff is a typedef equivalent to long.

dir: The seek direction. Must be one of the following enumerators:

ios::beg Seek from the beginning of the stream.

ios::cur Seek from the current position in the stream.

ios::end Seek from the end of the stream.

Example: diskfun.cpp

```
#include <fstream>              //for file streams
#include <iostream>
using namespace std;
class person                    //class of persons
{
protected:
        char name[80];          //person's name
        int age;                //person's age
public:
        void getData( )         //get person's data
        {
                cout << "\n   Enter name: "; cin >> name;
                cout << "   Enter age: "; cin >> age;
        }
        void showData(void)     //display person's data
        {
                cout << "\n   Name: " << name;
                cout << "\n   Age: " << age;
        }
};
```

```
int main( )
{
   char ch;
   person pers;                  //create person object
   fstream file;                 //create input/output file
   file.open("GROUP.DAT", ios::app | ios::out
                                  | ios::in | ios::binary );
   do                            //data from user to file
   {
      cout << "\nEnter person's data:";
      pers.getData( );           //get one person's data
      file.write( reinterpret_cast<char*>(&pers), sizeof(pers) );
      //write to file
      cout << "Enter another person (y/n)? ";
      cin >> ch;
   }while(ch=='y');              //quit on 'n'
```

reinterpret_cast<char*>(&pers)?

```
   file.seekg(0);                //reset to start of file

   //read first person
   file.read( reinterpret_cast<char*>(&pers), sizeof(pers) );
   while( !file.eof( ) )         //quit on EOF
   {
      cout << "\nPerson:";       //display person
      pers.showData();           //read another person
      file.read( reinterpret_cast<char*>(&pers), sizeof(pers) );
   }
   cout << endl;
   return 0;
}
```

Example:2

```
int main( )
{
   person pers;                  //create person object
   ifstream infile;              //create input file

   infile.open("GROUP.DAT", ios::in | ios::binary); //open file

   infile.seekg(0, ios::end);     //go to 0 bytes from end
   int endposition = infile.tellg();       //find where we are
   int n = endposition / sizeof(person);    //number of persons
   cout << "\nThere are " << n << " persons in file";
```

```
   cout << "\nEnter person number: ";
   cin >> n;
   int position = (n-1) * sizeof(person);  //number times size
   infile.seekg(position);        //bytes from start

   //read one person
   infile.read( reinterpret_cast<char*>(&pers), sizeof(pers) );
   pers.showData( );              //display the person
   cout << endl;
   return 0;
}
```

8.5 The Character Stream Input/Output Classes

The classes basic_stringbuf, basic_ostringstream, basic_istringstream and basic_stringstream are declared in sstream.

Header	Partial Description
sstream	Declares basic_stringbuf and the stringstream classes

8.5 The Character Stream Input/Output Classes basic_ostringstream

- basic_ostringstream

The class basic_ostringstream is used to write characters to an internal buffer that can be copied to a basic_string.

basic_ostringstream(iso_base::openmode
=ios_base::out);

8.5 The Character Stream Input/Output Classes basic_ostringstream

Example:

```
string name="monica";
ostringstream sout( name );
```

Example:

```
ostringstream sout;
sout<<"monica"<<123;
cout<<sout.str( )<<endl;
```

Output:

```
monica123
```

8.5 The Character Stream Input/Output Classes basic_istringstream

- **basic_istringstream**

The class basic_istringstream is used to read characters from an internal buffer.

basic_istringstream(ios_base::openmode
=ios_base::in);

8.5 The Character Stream Input/Output Classes basic_istringstream

Example:

```
float x;
string val="37.805";
istringstream sin(val);
sin>>x;
cout<<"x="<<val<<endl;
```

Output:

```
x=37.805
```

8.5 The Character Stream Input/Output Classes basic_stringstream

- **basic_stringstream**

The class basic_stringstream is used to read and write an internal buffer that can be copied to a basic_string.

basic_stringstream(ios_base::openmode

=io_base::in | ios_base::out);

Summarize

- Manipulators
- basic_istream, basic_ostream,basic_iostream
- basic_ofstream, basic_ifstream, basic_fstream
- basic_ostringstream, basic_istringstream, basic_stringstream

Exercise:

Read from file “in.txt”, and write it into file “D:\out.dat” in binary. For every line in “D:\out.dat”, please add a mask(0,1,2,……).