C++ I/O and File

Dr. Md. Humayun Kabir
CSE Department, BUET

Custom Inserter<<

```
#include <iostream>
using namespace std;
class myclass{
    int x, y;
public:
        myclass(int x, int y){this->x=x; this->y=y;}
        friend ostream &operator<<(ostream &out, myclass ob);
};
ostream &operator<<(ostream &out, myclass ob){
    out<<“x: “<<ob.x<<“, y: “<<ob.y<<endl;
    return out;
}
```

Custom Inserter<<

```
int main(){
    myclass ob(120, 130);
    cout<<ob;
    return 0;
}
```

Custom Extractor >>

```
#include <iostream>
using namespace std;
class myclass {
      int x, y;
public:
            myclass (int x, int y){this->x=x; this->y=y;}
            friend istream &operator>>(istream &in, myclass &ob);
};
istream &operator>>(istream &in, myclass &ob){
      cout<<“Enter x: “;
      in>>ob.x;
      cout<<“Enter y: “;
      in>>ob.y;
      return in;
}
```

Custom Extractor >>

```
int main(){
    myclass ob(120, 130);
    cout<<ob;
    cin>>ob;
    cout>>ob;
    return 0;
}
```

C++ File I/O

- C++ provides three classes to perform output and input of characters to/from files:
  - **ofstream:** Stream class to write on files
  - **ifstream:** Stream class to read from files
  - **fstream:** Stream class to both read and write from/to files.
- These classes are derived directly or indirectly from the classes istream and ostream.

C++ File I/O

- To work with file you need to include <fstream>
- A file needs to be opened by ifstream before read or ofstream to write or fstream to read and write
  - void ifstream::open(const char *filename, openmode mode=ios::in)
  - void ofstream::open(const char *filename, openmode mode=ios::out|ios::trunc)
  - void fstream::open(const char *filename, openmode mode=ios::in|ios::out)

C++ File I/O

- The value of mode determines how the file is opened
- Type **openmode** is an enumeration defined by **ios** that contains the following values:
  - ios::app
  - ios::ate
  - ios::binary
  - ios::in
  - ios::out
  - ios::trunc
- You can combine two or more of these value together by ORing them.
- If open() fails the stream will evaluate to false.

C++ File I/O

- You can also use **fstream(), ifstream()** and **ofstream()** constructors to open the files in one step.
- These constructors have the same parameters and defaults as their corresponding open() function
  - ifstream(const char *filename, openmode mode=ios::in)
  - ofstream(const char *filename, openmode mode=ios::out|ios::trunc)
  - fstream(const char *filename, openmode mode=ios::in|ios::out)
- All the streams have **bool is_open()** member function to check whether the file has been successfully opened or not.

C++ File I/O

- You can detect when the end of an input file has been reached by using **eof()** member function.
- You must close a file **close()** member function once you have done with the file.
  - fstream mystream(“myfile.txt)
  - if(mystream) or if(mystream.is_open())
  - while(!mystream.eof())
  - mystream<<“Testing file”
  - mystream>>str; //char *str
  - mystream.close()

C++ File I/O: Writing

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
      ofstream myfile;
      myfile.open ("example.txt");
      myfile << "Writing this to a file.\n";
      myfile.close();
      return 0;
}
```

C++ File I/O: Reading

```
#include <iostream>
#include <fstream>
using namespace std;
```

C++ File I/O: Reading

```
int main () {
      char ch;
      ifstream myfile ("example.txt");
      if (myfile.is_open()) {
          while ( !myfile.eof() ) {
              myfile>>ch;
              cout << ch;
          }
          cout<<endl;
          myfile.close();
      }
      else
      cout << "Unable to open
      file";
      return 0;
}
```

C++ File I/O: Unformatted (Binary)

- **istream &get(char *ch*)**
  - Member function of **fstream** and **ifstream**
  - Associated stream must be opened with ios::binary openmode option
  - Reads a single character from the stream and puts the value in *ch*
  - Returns the reference to the stream
  - If the end-of-file is reached returned stream will be evaluated false

C++ File I/O: Unformatted (Binary)

- **ostream &put(char *ch*)**
  - Member function of **fstream** and **ofstream**
  - Associated stream must be opened with ios::binary openmode option
  - Writes a single character from *ch* to the stream
  - Returns the reference to the stream

C++ File I/O: Unformatted (Binary)

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main(){
	char str[100]="I love Bangladesh";
	ofstream out("myfile.txt", ios::out|ios::binary);
	if(!out.is_open()){
		cout<<"Cannot open file"<<endl;
		exit(1);
	}
	for(int i=0; str[i]; i++)
		out.put(str[i]);
	out.close();
	return 0;
}
```

C++ File I/O: Unformatted (Binary)

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main(){
	char ch;
	ifstream in("myfile.txt", ios::in|ios::binary);
	if(!in.is_open()){
		cout<<"Cannot open file"<<endl; exit(1);
	}
	while(!in.eof()){
		in.get(ch);
		cout<<ch;
	}
	cout<<endl;
	return 0;
	in.close();
}
```

C++ File I/O: Unformatted (Binary)

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main(){
	char str[100]="I love Bangladesh", ch;
	fstream mystream("myfile.txt", ios::out|ios::in|ios::binary);
	if(!mystream.is_open()){
		cout<<"Cannot open file"<<endl; exit(1);
	}
	for(int i=0; str[i]; i++)  mystream.put(str[i]);
	mystream.seekg(0, ios::beg);
	while(!mystream.eof()){
		mystream.get(ch); cout<<ch;
	}
	cout<<endl;
	mystream.close();
	return 0;
}
```

C++ File I/O: Unformatted (Binary)

- To read blocks of unformatted (binary) data from a file you need to use **read()** member function of **fstream** or **ifstream**.
  - **istream &read(char **buf*, streamsize *num*)**
  - Reads *num* number of bytes from the stream and puts them in *buf*
  - Might read less than *num* number of bytes if the end-of-file is reached ahead
  - How many bytes have been read can be determined by using **gcount()** member function.

C++ File I/O: Unformatted (Binary)

- To write blocks of unformatted (binary) data in a file you need to use **write()** member function of **fstream** or **ofstream**.
  - **ostream &write(const char *_buf_, streamsize _num_)**
  - Writes *num* number of bytes from *buf* in the the stream

C++ File I/O: Unformatted (Binary)

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
using namespace std;
int main(){
	char str1[100]="I love Bangladesh", str2[100];
	fstream mystream("myfile.txt", ios::out|ios::in|ios::binary);
	if(!mystream.is_open()){
		cout<<"Cannot open file"<<endl; exit(1);
	}
	mystream.write(str1, strlen(str1)+1);
	mystream.seekg(0, ios::beg);
	mystream.read(str2, strlen(str1)+1);
	cout<<str2<<endl;
	mystream.close();
	return 0;
}
```

C++ File I/O: Unformatted (Binary)

- To read blocks of unformatted (binary) data from a file you can also use overloaded **get()** member function of **fstream** or **ifstream**.
  - **istream &get(char **buf*, streamsize *num*)**
  - **istream &get(char **buf*, streamsize *num*, char *delim*)**
  - Reads *num-1* number of bytes from the stream and puts them in *buf*
  - Character sequence in the *buf* is null terminated.
  - If newline or delim character is encountered before *num-1* characters it is not read and removed from the stream.

C++ File I/O: Unformatted (Binary)

- To read blocks of unformatted (binary) data from a file you can also use overloaded **getline()** member function of **fstream** or **ifstream**.
  - **istream &getline(char **buf*, streamsize *num*)**
  - **istream &getline(char **buf*, streamsize *num*, char *delim*)**
  - Reads *num-1* number of bytes from the stream and puts them in *buf*
  - Character sequence in the *buf* is null terminated.
  - If newline delim character is encountered before *num-1* characters it is read and removed from the stream.

C++ File I/O: Unformatted (Binary)

- Another overloaded **get()** member function of **fstream** or **ifstream** works as follows.
  - **int get()**
  - Returns the next character from the stream.
  - Returns EOF if end-of-file is encountered
  - Similar to C's getc() function.
- You can return the last character read from an input stream using **putback()** member function.
  - **istream &putback(char c)**
- You can obtain the next character in the input stream without removing it using **peek()** member function.
  - **int peek()**

C++ File I/O: Unformatted (Binary) Random Access

- **istream &seekg(off_type *offset*, seekdir *origin*)**
  - is a member function of **ifstream** and **fstream**
  - Moves the current get pointer *offset* number of bytes from the specified *origin*
- **ostream &seekp(off_type *offset*, seekdir *origin*)**
  - is a member function of **ofstream** and **fstream**
  - Moves the current put pointer *offset* number of bytes from the specified *origin*
- seekdir is an enumeration defined in **ios** with the following values:
  - ios::beg
  - ios::cur
  - ios::end

C++ File I/O: Checking the Status

- C++ I/O System maintains status information about the outcome of each I/O operation.
- The current status of an I/O stream is described in an object of type **iostate**, which is an enumeration defined in **ios** with the following values:
  - ios::goodbit
  - ios::eofbit
  - ios::failbit
  - ios::badbit
- To read the I/O status you can use **rdstate()** function
  - **iostate rdstate()**
  - It is a member of **ios** and inherited by all the streams

C++ File I/O: Checking the Status

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
void checkstatus(ifstream &in);
int main(){
	char ch;
	ifstream in("myfile.txt");
	if(!in.is_open()){
		cout<<"Cannot open file"<<endl;
		exit(1);
	}
	while(!in.eof()){
		ch=in.get();
		checkstatus(ifstream &in);
		cout<<ch;
	}
	cout<<endl;
	in.close();
	return 0;
}
```

C++ File I/O: Checking the Status

```
void checkstatus(ifstream &in){
    ios::iostate s;
    s=in.rdstate();
    if(s&ios::eofbit)
        cout<<“EOF encountered”<<endl;
    else if(s&ios::failbit)
        cout<<“Non-Fatal I/O error encountered”<<endl;
    else if(s&ios::badbit)
        cout<<“Fatal I/O error encountered”<<endl;
}
```

C++ File I/O: Checking the Status

- You can also determine the status using following **ios** member functions those have also been inherited by all the streams
  - **bool eof()**
  - **bool good()**
  - **bool fail()**
  - **bool bad()**

C++ File I/O: Checking the Status

```
while(!in.eof()){
    ch=in.get();
    if(in.fail()||in.bad()){
        cout<<“I/O Error … terminating”<<endl;
        return 1;
    cout<<ch;
}
```

C++ Array Based I/O

- #include <strstream>
  - istrstream to read from the array
  - ostrstream to write in the array
  - strstream to read and write

C++ Array Based I/O

- istrstream istr(const char *buf)
- ostrstream ostr(char *buf, streamsize size, openmode mode=ios::out)
- strstream iostr(char *buf, streamsize size, openmode mode=ios::in|ios::out)
- streamsize no_characters=ostr.pcount();

C++ Array Based I/O

```
#include <iostream>
#include <strstream>
using namespace std;
int main(){
    char buf[225];
    ostrstream ostr(buf, sizeof(buf));
    ostr<<"Array-like I/O uses stream just like";
    ostr<<"normal I/O"<<endl;
    cout<<ostr.pcount()<<endl;
    cout<<buf;
    return 0;
}
```

C++ Array Based I/O

```
#include <iostream>
#include <strstream>
using namespace std;
int main(){
    char buf=“Hello 100 3.14 a”;
    istrstream istr(buf);
    int i;
    char str[80];
    float f;
    char c;
    istr>>str>>i>>f>>c;
    cout<<str<<“ “<<f<< “ “<<i<<“ “<<c<<endl;
    return 0;
}
```

C++ Array Based I/O

```
#include <iostream>
#include <strstream>
using namespace std;
int main(){
    char buf="Hello 100 3.14 a";
    istrstream istr(buf);
    char c;
    while(!istr.eof()){
        istr.get(c);
        cout<<c;
    }
    cout<<endl;
    return 0;
}
```

cout<<“Thank You”<<endl;

cout<<“**Have a Good Day**”<<endl;