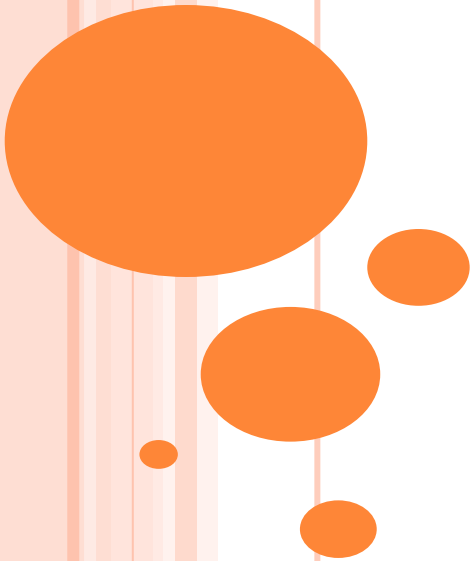


CSE 107: Object Oriented Programming Language

Dr. Tanzima Hashem
Professor
CSE, BUET

C++ Basic I/O

- **Stream**
  - a logical device that either produces or consumes information
  - linked to a physical device
- **ios:**
  - contains many member functions and variables that control or monitor the fundamental operation of a stream
  - **istream** and **ostream** inherit from **ios**
  - **iostream** inherits from **istream** and **ostream**

C++ Basic I/O

- **<<** (left-shift operator)
  - Overloaded as *stream insertion (output) operator*
  - Associated with cout, a predefined stream linked to the console output (monitor)

- **>>** (right-shift operator)
  - Overloaded as *stream extraction (input) operator*
  - Associated with cin, a predefined stream linked to the console input (keyboard)
- In order to use either << or >> operator for console I/O programs must include iostream

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;
}
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 ostream &operator<<(ostream &out, myclass ob);
        friend istream &operator>>(istream &in, myclass &ob);
};

ostream &operator<<(ostream &out, myclass ob){
        out<<“x: “<<ob.x<<“, y: “<<ob.y<<endl;
        return out;
}
```

Custom Extractor >>

```
istream &operator>>(istream &in, myclass &ob){
        cout<<“Enter x: “;
        in>>ob.x;
        cout<<“Enter y: “;
        in>>ob.y;
        return in;
}
int main(){
        myclass ob(120, 130);
        cout<<ob;
        cin>>ob;
        cout<<ob;
        return 0;
}
```

C++ File I/O

- A computer file is stored on a secondary storage device (e.g., disk)
  - permanent
  - can be used to
    - provide input data to a program
    - or receive output data from a program
    - or both;
  - should reside in project directory for easy access;
  - must be opened before it is used

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
- Done with the same operations (insertion, extraction) as keyboard input and monitor output
- Simply open input or output object with connection to a file and use it where you would use cin or cout

C++ File I/O

- To work with file you need to
  - include <fstream>
  - create input object of type *ifstream*
  - or output object of type *ofstream*

C++ File I/O

- 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: Writing/Reading

```
#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;
}
```

```
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: Writing/Reading (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;
}
```

```
#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;
  in.close();
  return 0;
}
```

C++ File I/O: Writing/Reading (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);  // binary is used cause of get & put
        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))

- **istream &read(char **buf*, streamsize *num*)**
  - Member function of **fstream** or **ifstream**
  - 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))

- **ostream &write(const char **buf*, streamsize *num*)**
  - Member function of **fstream** or **ofstream**
  - Writes *num* number of bytes from *buf* in the the stream

C++ File I/O: Writing/Reading (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))

- Overloaded **get()** member function of **fstream** or **ifstream**

  - **istream &get(char **buf*, streamsize *num*)**
  - **istream &get(char **buf*, streamsize *num*, char *delim*)**
  - **int get()**

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

- **istream &get(char **buf*, streamsize *num*)**
  - Reads *num-1* characters into *buf* until
    - *num-1* characters have been read
    - a newline is found
    - the end of the file has been encountered
  - Character sequence in the *buf* is null terminated
  - If newline character is encountered, it is not read and removed from the stream

- **istream &get(char **buf*, streamsize *num*, char *delim*)**
  - Reads *num-1* characters into *buf* until
    - *num-1* characters have been read
    - a delim character is found
    - the end of the file has been encountered
  - Character sequence in the *buf* is null terminated
  - If delim character is encountered, it is not read and removed from the stream

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

- **int get()**
  - Returns the next character from the stream
  - Returns EOF if end-of-file is encountered
  - Similar to C's getc() function.

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

- Overloaded **getline()** member function of **fstream** or **ifstream**

- **istream &getline(char **buf*, streamsize *num*)**
- **istream &getline(char **buf*, streamsize *num*, char *delim*)**

- Same as get, except extract and remove the newline or delim character from the input stream

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

- **istream &putback(char ch)**
  - Puts **ch** back to the input stream so the next extracted character will be **ch**

- **int peek()**
  - Returns the next character in the input stream without removing it

- **ostream &flush()**
  - Force the information to be physically written to disk before the internal buffer is full

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

- **istream &seekg(off_type *offset*, seekdir *origin*)**
  - 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*)**
  - 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 (Unformatted (Binary)) Random Access

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

using namespace std;

int main (int argc, char** argv)
{
fstream myFile("test.txt", ios::in | ios::out | ios::trunc);

myFile << "Hello World";

myFile.seekg(6, ios::beg);
```

```
char A[6];
myFile.read(A, 5);
A[5] = '\0';

cout << A << endl;

myFile.close();
}
```

Output:

World

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

- **pos_type tellg()**
  - Returns current position of get pointer
- **pos_type tellp()**
  - Returns current position of put pointer

- **istream &seekg(pos_type position)**
- **ostream &seekp(pos_type position)**
  - Ovreloaded versions of seekg and seekp

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

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

int main()
{
   long position;
   fstream file;

   file.open("myfile.txt");

   file.write("this is an apple", 16);
   position = file.tellp();

   file.seekp(position - 7);
   file.write(" sam", 4);
   file.close();
}
```

Output:

this is a sample

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

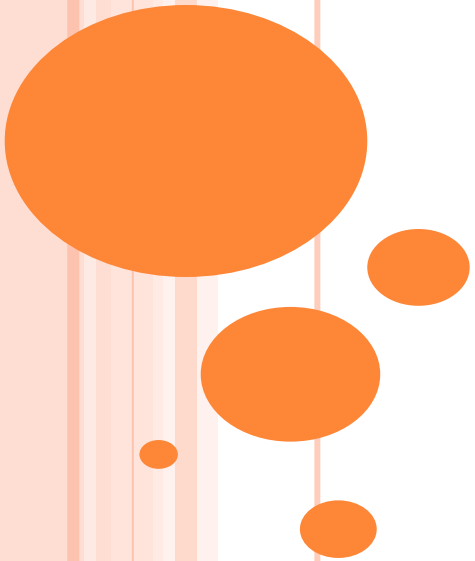
```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
void checkstatus(ifstream &in);
int main(){
        char ch;
        ifstream in("myfile.txt");
        if(!in.is_open()){
                cout<<"Cannot open file"<<endl;
                exit(1);
        }
        while(!in.eof()){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

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

- The status can be determined by 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;
}
```



Acknowledgement

http://faizulbari.buet.ac.bd/Courses.html

http://mhkabir.buet.ac.bd/cse201/index.html

The End

Topic Covered: Sections 8.5, 8.6, 9.2, 9.3, 9.4, 9.5, 9.6