WEEK 12
STREAMS AND FILES

Dr. Yuzana Win

OUTLINE

- Persistent Data
- Stream Classes
- File Handling
  - Text File
  - Binary File

PERSISTENT DATA

- Persistence is a very valuable trait in any person, it is particularly important in programmers
- The data should survive when the program is finished

WHAT IS A FILE?

- A file is a collection of data, and is located on persistent storage such as a hard drive, a CD-ROM, or other storage device.
- The typical computer system has much less memory storage than hard disk storage.
- Disk drive holds much more data than can fit in the computer's RAM.
- The disk memory, because it is *nonvolatile*, lasts longer because the disk retains its contents when you power-off your computer.

DISK FILE ACCESS

- **Read, write, change, and delete data from the file**
- **Types of Disk File Access**
  - Sequential File Access: must be accessed in the same order the file was written.
  - Random File Access : can access random access files in any order you want.

SEQUENTIAL FILE CONCEPTS

Three operations

- Read from disk files (Reading a file)
- Create disk files (Writing a file)
- Add to disk files (Appending to a file)

FILE HANDLING

TWO TYPES OF FILES

- **Text File**
- **Binary File**

FILE HANDLING

- In C++, you open a file by linking it to a **stream.**
- Before you can open a file, you must first obtain a stream.
- There are three classes in C++ which are used for File Read/Write operations.

  They are

  - **ifstream** - To create an input stream
  - **ofstream** - To create an output stream
  - **fstream** - To create a stream that will be performing both input and output operations

STREAM CLASSES

- A stream is a general name given to a flow of data.
- A stream is represented by an object of a particular class.
- Different streams are used to represent different kinds of data flow.
- For example, the ***ifstream*** class represents data flow from input disk files.

THE STREAM CLASS HIERARCHY

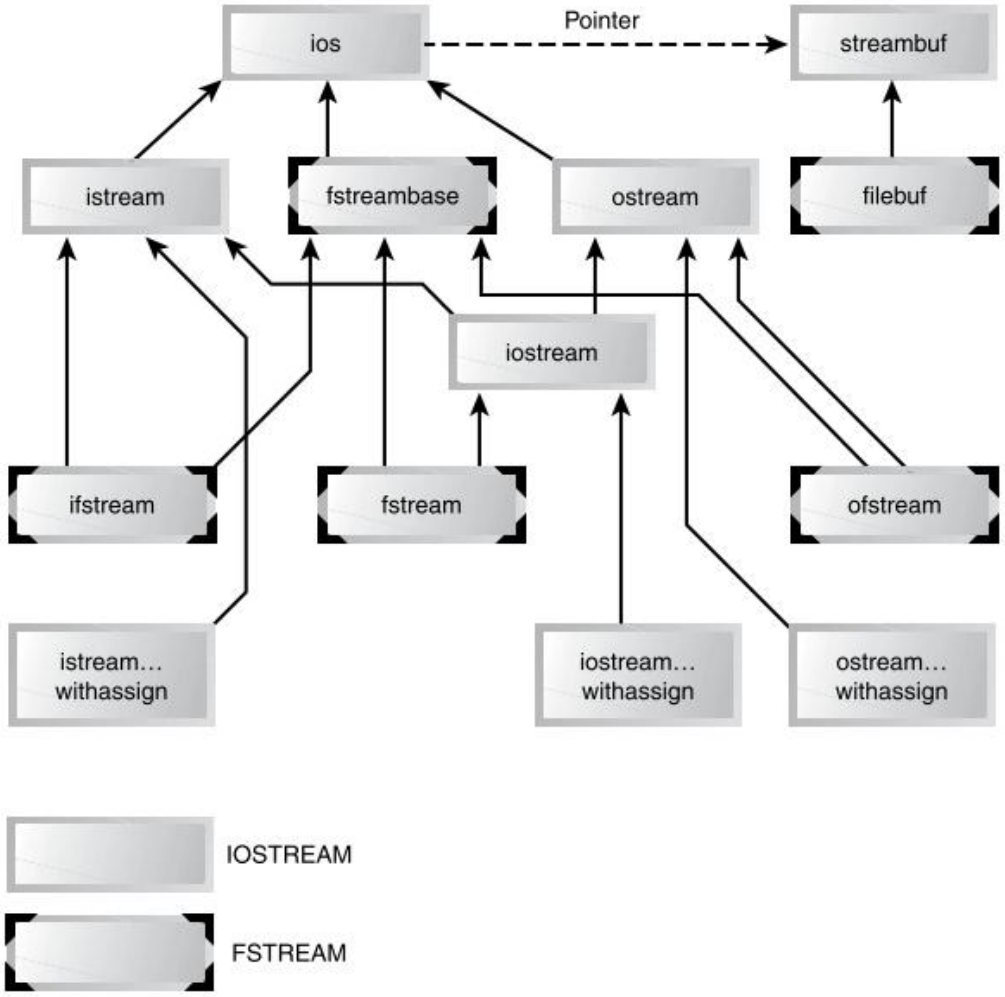


Figure: Stream Class Hierarchy

FILES IN C++

1. Location of the file on the disk
2. Type of the file
3. Allowed operations
4. Present position in the file

OPENING A FILE

The **open()** function is a member of each of the three stream classes.

The prototype for each is as follows:

```
void ifstream::open(const char *filename, ios::openmode mode = ios::in);
void ofstream::open(const char *filename, ios::openmode mode = ios::out | ios::trunc);
void fstream::open(const char *filename, ios::openmode mode = ios::in | ios::out);
```

OPENING A FILE

- *Filename* - is the name of the file; it can include a path specifier.

- Open mode - The value of mode determines how the file is opened.
  - **ios::app // start writing at end of file (append)**
  - **ios::ate // start reading and writing at end of file**
  - **ios::binary // open file in binary mode**
  - **ios::in // open for reading**
  - **ios::out // open for writing**
  - **ios::trunc // truncate file to zero length if it exists**

EXAMPLE OPENING A FILE

- **Using open() method,**

```
ofstream out;
out.open("test", ios::out);
(OR)
out.open("test"); // defaults to output and normal file
```

- **Using constructor of ofstream**

```
ofstream myfile("test", ios::out);
 ifstream myfile("test", ios::in);
```

SOME FUNCTIONS OF OSTREAM

- <<
- put(ch)
- get(str)
- write(str,size)
- seekp(position)
- seekp(position,seek_dir) // ios::beg, ios::cur, ios::end
- pos=tellp() // get the current file pointer's position

FUNCTIONS OF ISTREAM

- >>
- get(ch)
- get(str)
- get(str,max)
- get(str, max, delim)
- getline(str,max)
- getline(str,max,delim)
- read(str,max)
- seekg()
- seekg(pos,seek_dir)// ios::beg, ios::cur, ios::end
- pos=tellg(pos) // get the current file pointer's position

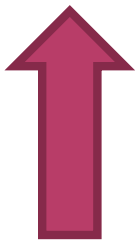
READING FROM A FILE

- Steps
  - Declare file variable
  - Open the file (read mode)
  - Read from the file
  - Close the file

EXAMPLE: READING TEXT FILES

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ifstream in(" C:\\OUTFILE.txt "); // input
    if(!in) {
            cout << "Cannot open INFILE file.\n";
            return 1;}
    char item[20];
    float cost;
    in >> item >> cost;     cout << item << " " << cost << "\n";
    in >> item >> cost;     cout << item << " " << cost << "\n";
    in >> item >> cost;     cout << item << " " << cost << "\n";
    in.close();     return 0;   }
```

```
Radios 39.95
Toasters 19.95
Mixers 24.8
Press any key to continue . . .
```

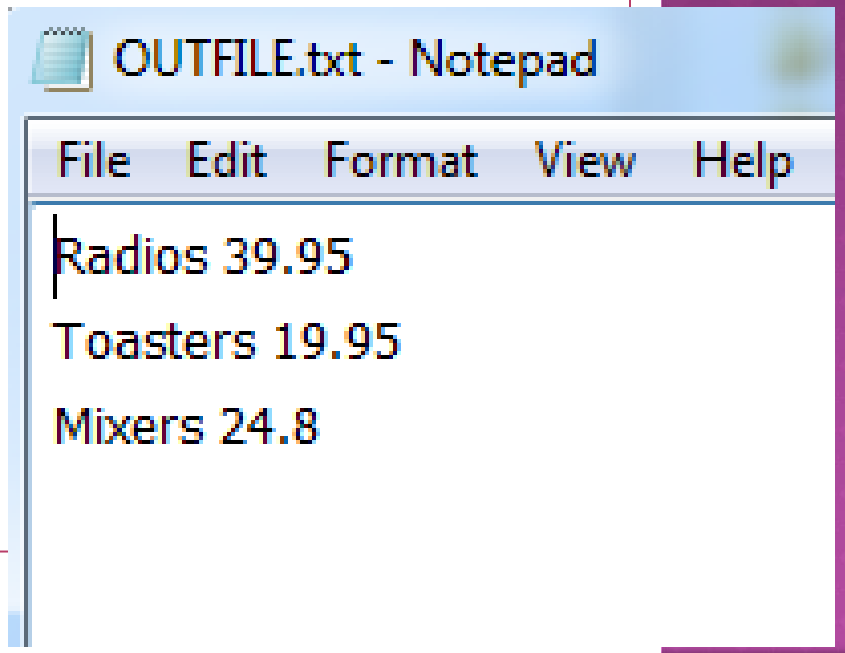


Writing to a File

- Steps
  - Declare file variable
    - Same as before
  - Open the file (write mode)
  - Write to the file
  - Close the file
    - Same as before

EXAMPLE: WRITING TEXT FILES

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{  ofstream out("C:\\OUTFILE.txt"); // output, normal file
   if(!out) {
         cout << "Cannot open OUTFILE file.\n";
         return 1;
   }
   out << "Radios " << 39.95 << endl;
   out << "Toasters " << 19.95 << endl;
   out << "Mixers " << 24.80 << endl;
   out.close();
   return 0;  }
```



EXAMPLE : USING PUT( ) FUNCTION

```
#include <iostream>
#include <fstream>
using namespace std;
int main( ){
    int i;
    ofstream out("CHARS", ios::out);
    if(!out) { cout << "Cannot open output file.\n";
                    return 1; }
    // write all characters to disk
    for(i=0; i<256; i++)
            out.put((char) i);
    out.close();
    return 0;
}
```

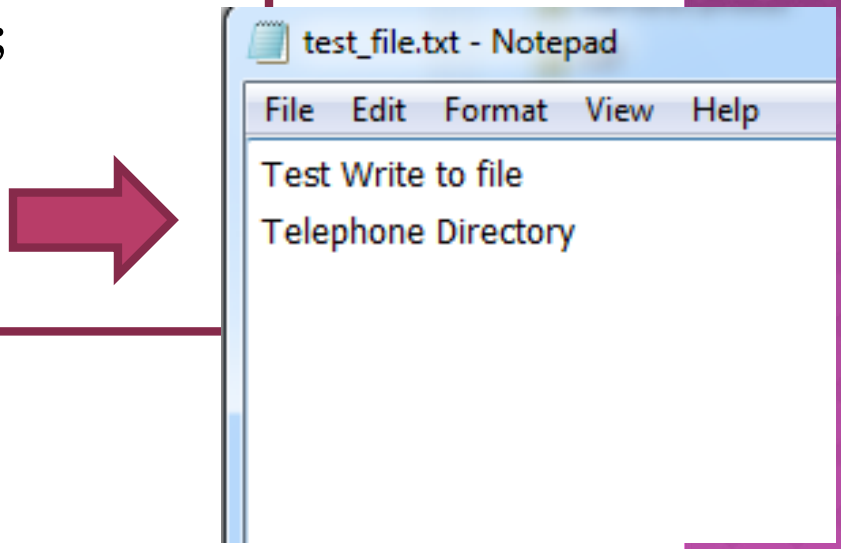
EXAMPLE : USING GET( ) FUNCTION

```
#include <iostream>
#include <fstream>
using namespace std;
int main( )  {
   char ch;
   ifstream in(“test.dat”, ios::in);
   if(!in) {
          cout << "Cannot open file.";  return 1;
   }
   while(in) { // in will be false when eof is reached
          in.get(ch);
          if(in) cout << ch;
   }
return 0;   }
```

EXAMPLES

Writing to a text file using fstream class

```
#include <fstream>
using namespace std;
int main()
{
        fstream file_op("c:\\test_file.txt", ios::out);
        file_op<<"Test Write to file";
        file_op << "Telephone Directory";
        file_op.close();
        return 0;
}
```



get() and getline() function prototypes

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

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

EXAMPLES

using *fstream getline* method

```
#include <fstream.h>
int main()
{
   char str[2000];
   fstream file_op("c:\\test_file.txt", ios::in);
   while(!file_op.eof()) {
                file_op.getline(str,2000);
                cout <<str;
    }
   file_op.close();
   return 0;
}
```

BINARY FILE HANDLING

READ( ) AND WRITE( ) METHODS

- To read and write blocks of binary data in C++.

Syntax

istream &**read**(char **buf*, *streamsize num*);

ostream &**write**(const char **buf*, *streamsize num*);

- The ***read***() function reads num characters from the invoking stream and puts them in the buffer pointed to by buf.
- The ***write***() function writes num characters to the invoking stream from the buffer pointed to by buf.
- ***streamsize*** is a type defined by the C++ library as some form of integer. It is capable of holding the largest number of characters that can be transferred in any one I/O operation.

EXAMPLE : USING READ() AND WRITE() METHODS

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
struct status {
    char name[80];
    double balance;
    unsigned long account_num;
};
int main(){
    status acc;
    strcpy(acc.name, "Ralph
    Trantor");
    acc.balance = 1123.23;
    acc.account_num = 34235678;
```

```
// write data
ofstream outbal("balance", ios::out |
                           ios::binary);
if(!outbal) {
    cout << "Cannot open file.\n";
    return 1; }
outbal.write((char *) &acc, sizeof(
                                 acc));
outbal.close();
// now, read back;
ifstream inbal("balance", ios::in |
                          ios::binary);
if(!inbal) {
    cout << "Cannot open file.\n";
    return 1;
}
```

EXAMPLE : USING READ() AND WRITE() METHODS

```
    inbal.read((char *) &acc, sizeof(status));
    cout << acc.name << endl;
    cout << "Account # " << acc.account_num;
    //cout.precision(2);
    cout << endl << "Balance: $" << acc.balance << endl;
    inbal.close();
    return 0;
}
```

Output

```
Ralph Trantor
Account # 34235678
Balance: $1123.23
Press any key to continue . . .
```

MORE FUNCTIONS FOR FILE HANDLING

bool **eof**( );

- It returns true when the end of the file has been reached; otherwise it returns false.

istream &**ignore**(streamsize *num=1, int_type delim=EOF);*

- It reads and discards characters until either num characters have been ignored (1 by default) or the character specified by delim is encountered (EOF by default).

int_type **peek**( );

- It returns the next character in the stream or EOF if the end of the file is encountered.

EXAMPLE

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream in("test.txt");
    if(!in) {
            cout << "Cannot open file.\n";
            return 1; }
    /* Ignore up to 10 characters or until first space is found. */
    in.ignore(10, ' ');
    char c;
    while(in) {
            in.get(c);
            if(in) cout << c;
    }
    in.close();
     return 0;
}
```

FUNCTIONS FOR RANDOM ACCESS

- istream & **seekg**(off_type *offset*, *seekdir origin*);
  - The seekg() function moves the associated file's current get pointer offset number of characters from the specified origin, which must be one of these three values:
    - **ios::beg** **Beginning-of-file**
    - **ios::cur** **Current location**
    - **ios::end** **End-of-file**

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

  file.seekg(0, ios::beg) // reset to start of file

  file.seekg(0, ios:: end) // reset to end of file

  file.seekg(-10, ios::cur); //reset 10 bytes before from current pos

- ostream & **seekp**(off_type *offset*, *seekdir origin*);
  - moves the associated file's current put pointer offset number of characters from the specified origin, which must be one of the values just shown.

FUNCTIONS FOR RANDOM ACCESS

Obtaining the Current File Position

- pos_type **tellg**( );
- pos_type **tellp**( );

- can use the values returned by **tellg() and tellp()** as arguments to the following forms of **seekg()** and **seekp()** , respectively.

SUMMARY

- In C++, each file is simply a sequential stream of bytes.
- A file must first be opened properly before it can be accessed for reading or writing. When a file is opened, a stream is associated with the file.
- Successfully opening a file returns the address of a file structure, which contains a file descriptor and a file control block.

ASSIGNMENTS

1. In a loop, prompt the user to enter name (char array) and employee number (unsigned integer). Then using the insertion operator (<<) to write these two data items to an ofstream object.

   Don’t forget that string must be terminated with a space or white space character. When the user indicates that no more name data will be entered, close the ofstream object, open an ifstream object, read and display all the data in the file, and terminate the program.

ASSIGNMENTS

2. Create a structure

```
struct building { char owner[20];
                  int building_no;
                  int room_no;
                };
```

Accept the data from the user and then write to the binary formatted file until no more data from the user. And then display the followings:

(i) display number of records from the user

(ii) display all building information

(iii) display the third building information, if it exist

ASSIGNMENTS

3. Write a program that Reads a list of first and last names from the file phonebook. The first and last names are stored one per line. Prints each entry as last name, first name.

4. Write a program that Reads a list of first and last names from the file phonebook. The first and last names are stored one per line. Writes each entry as last name, first name to the file phonebook2. The entries are stored one per line.