Chapter 5

Objective

- Manipulators
- File Processing In C++
- Making Of A File
- C++ Files and Streams
- Writing And Reading A Text File
- Writing And Reading A Binary File
- Stream Operating Modes
- Friend Functions
- Friend Class

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Copyright © 2013 By Suleman Saya

Unit 1

Manipulators

•Manipulators are used as a special I/O formatting functions.

•Manipulators provide the facility to use insertion << and extraction >> operator for I/O.

•The programmer can use built-in manipulators or manipulators can be customized.

•There are two kinds of built-in manipulators:

Manipulators that take the parameter.

Manipulators that has no parameter.

•You must include iomanip.h file to use manipulators that take the parameter.

```
0   #include <iostream>    // Example 5-1
1   #include <iomanip>
3   using namespace std;
5   int main()
6   {
7       cout << hex << 15 << endl;
8       cout << oct << 9 << endl;
10       cout << setfill('0') << setw(10);
11       cout << 15 << "\nConverted to octel " << endl;
12
13       return 0;
14   }
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 1

```
0   #include <iostream>    // Example 5-2
1   #include <iomanip>
2   #include <cmath>
4   using namespace std;
6   int main()
7   {
9       cout << setprecision(4);
10       cout << setfill(' ') << setw(6) << 'X';
11       cout << setfill(' ') << setw(5);
12       cout << "sqrt(x)" << setfill(' ') << setw(10);
13       cout << "x^2\n\n";
14
15       for (double x = 2.0; x <= 10.0; x++) {
16       cout << setw(7) << x << setfill(' ') << setw(3);
17       cout << setw(7) << sqrt(x) << setfill(' ') << setw(3);
18       cout << setw(7) << x * x << '\n';
19       }
20
21       return 0;
22   }
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 2

File Processing In C++

- Variables and arrays are used for temporary storage of data in internal memory
- Files are used for permanent storage of large amounts of data on a secondary storage devices (usually some type of disk or tape)
- Files are organized for sequential access or random access (also called direct access)
- A file can contain formatted data (as would be written to the monitor), or unformatted "raw" data (as it is stored in memory)

Making Of A File

- A group of related bytes is called a field
- A group of related fields is called a record; in C++, we can represent these as a struct or a class
- Usually one field in each record is chosen as a key field that uniquely identifies the record
- A group of related records is a file
- A group of related files is a database

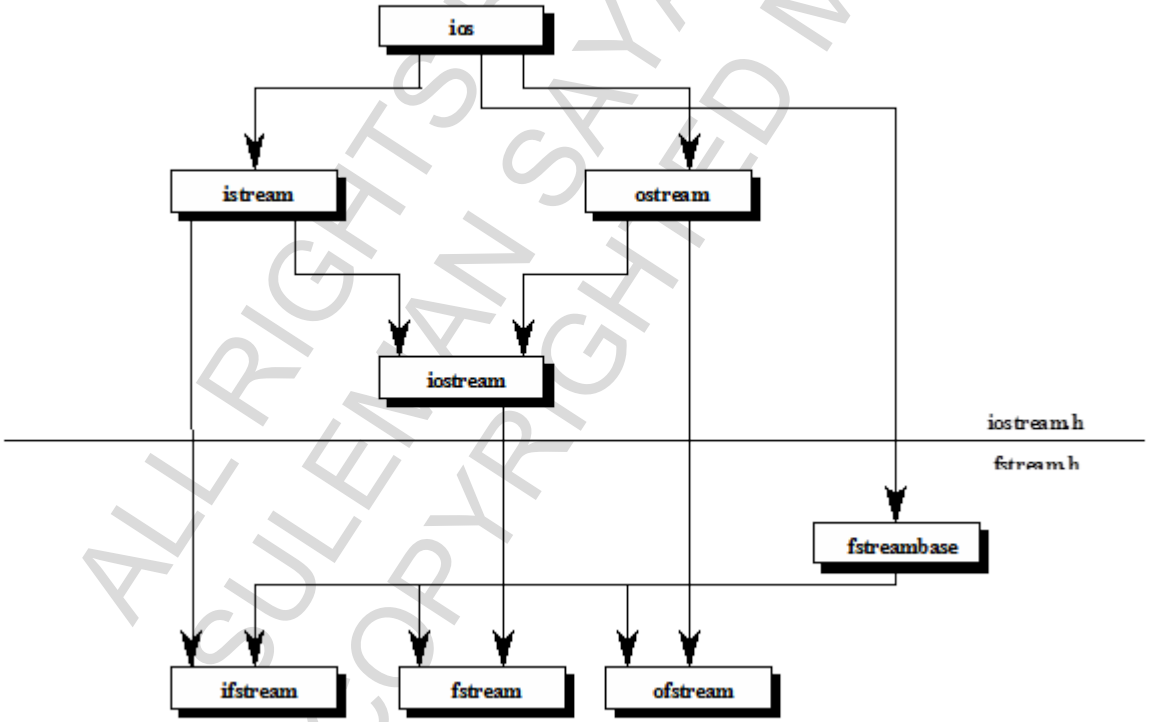
ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 2

C++ Files and Streams

C++ views each file simply as a sequence of bytes

- A file ends with either an end-of-file marker (eof), or at a specific byte number (the file size) in a system-maintained, administrative data structure (like a file directory)
- When a file is opened, a file object of the appropriate type is instantiated, and a stream (also called a logical file name) is associated with the object
- To perform file processing in C++, the header files <iostream> and <fstream> must be #included in a source file.



ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 2

C++ Files and Streams

- **fstream** class contains the definitions for the stream classes **ifstream** (for file input) and **ofstream** (for file output)

- Since **fstream** class is derived from **iostream** class, all iostream class member functions, operators, and manipulators used for formatted console I/O can be applied to file streams, as well

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 3

Writing And Reading A Text File

```
0   #include <iostream>    // Example 5-3
1   #include <fstream>
2
3   using namespace std;
4
5   int main()
6   {
7       // create normal output file
8       ofstream OutFile("readme.txt");
9
10       if(!OutFile) {
11           cout << "Cannot open output file.\n";
12           return 1;
13       }
14       // write to a file
15       OutFile << "Tax_Return_1996_Amount:";
16       OutFile << 100 << "\n";    // write to a file
17
18       OutFile.close();
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 3

```
19
20        ifstream InFile("readme.txt"); // open normal input file
21
22        if(!InFile) {
23            cout << "Cannot open input file.\n";
24            return 1;
25        }
26
27        char Str[80];
28
29        InFile >> Str;   // read from a file
30
31        cout << Str << "\n";
32
33        InFile.close();
34
35        return 0;
36
37    }
38
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 3

Writing And Reading A Binary File

```
0   #include <iostream>    // Example 5-4
1   #include <fstream>
2   #include <string>
3
4   using namespace std;
5
6   int main()
7   {
8       // create normal output file
9       ofstream OutFile("readme.dat", ios::binary);
10
11       if(!OutFile) {
12           cerr << "Cannot open output file.\n";
13           return 1;
14       }
15
16       char sOutBuff[] = "Tax Return 1996 Amount:";
17       double dAmount = 100.00;
18
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 3

```
19        // write to a file
20        OutFile.write(sOutBuff,strlen(sOutBuff));
21        OutFile.write((char *) &dAmount, sizeof(double));
23        OutFile.close();
25        // open Binary input file
26        ifstream InFile("readme.dat",ios::binary);
28        if(!InFile) {
29            cerr << "Cannot open input file.\n";
30            return 1;
31        }
33        char sInBuff[23];
34        double nSum;
36        InFile.read(sInBuff, 23);
37        InFile.read((char *)&nSum, sizeof(double));
38        nSum = nSum + 100;
39        cout << sInBuff << nSum << "\n";
40
41        InFile.close();
43        return 0;
44    }
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 4

Passing Arguments to main

```
0   #include <iostream>    // Example 5-5
1   #include <string>
2   #include <fstream>
3
4   using namespace std;
5
6   int main(int argc, char *argv[])
7   {
8       if(argc!=2) {
9           cout << "Usage: WRITE <filename>\n";
10           return 1;
11       }
12
13       ofstream OutFile(argv[1]); // output, normal file
14
15       if(!OutFile) {
16           cout << "Cannot open output file.\n";
17           return 1;
18       }
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 4

Passing Arguments to main

```
19
20        cout << "Write strings to disk, RETURN to stop\n";
21
22        char StrBuf[80];
23        memset(StrBuf,'\0',80);
24
25        char *pStr = StrBuf;
26
27        do {
28            cout << ": ";
29            cin.getline(pStr,80);       // read string
30            OutFile << pStr << "\n";    // write string to a file
31        } while (*pStr);
32
33        OutFile.close();
34
35        return 0;
36    }
37
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 5

Read a Character and Display it

```
0   #include <iostream>    // Example 5-6
1   #include <fstream>
2
3   using namespace std;
4
5   int main(int argc, char *argv[])
6   {
7
8       if(argc!=2) {
9           cout << "Usage: PR <filename>\n";
10           return 1;
11       }
12
13       ifstream InFile(argv[1]);
14
15       if(!InFile) {
16           cout << "Cannot open file";
17           return 1;
18       }
19
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 5

```
20        char cCharacter;
21
22        while(!InFile.eof()) {
23            InFile.get(cCharacter);
24            cout << cCharacter;
25        }
26
27        return 0;
28    }
29
30
31
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 5

Read User Input and Write to a File

```
0   #include <iostream>    // Example 5-7
1   #include <fstream>
2
3   using namespace std;
4
5   int main(int argc, char *argv[])
6   {
7       if(argc!=2) {
8           cout << "Usage: WRITE <filename>\n";
9           return 1;
10       }
11
12       ofstream OutFile(argv[1]);
13
14       if(!OutFile) {
15           cout << "Cannot open file";
16           return 1;
17       }
18
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 5

```
19        char cCharacter;
20        cout << "Enter a $ to stop\n";
21        do {
22            cout << ": ";
23            cin.get(cCharacter);      // read a character
24            // write a character to a file
25            OutFile.put(cCharacter);
26        } while (cCharacter!='$');
27
28        OutFile.close();
29
30        return 0;
31    }
32
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 6

Read and Write with Single File Object

```
0   // Demonstrate peek() and seekp().
1   #include <iostream>    // Example 5-8
2   #include <fstream>
3
4   using namespace std;
5
6   int main()
7   {
8       fstream FileObj(“data.txt",ios::in | ios::out);
9
10       if(!FileObj) {
11           cout << "Cannot open output file.\n";
12           return 1;
13       }
14       FileObj << 123 << "this is a test" << 23;
15       FileObj << "Hello there!" << 99 << "sdf" << "\n";
16       char cCharacter;
17       char sLine[80], *pString;
18       FileObj.seekp(0L);    // Go to top of the file
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 6

```
19        do {
20            pString = sLine;
21            cCharacter = FileObj.peek();//see the next type of char
22            if(isdigit(cCharacter)) {    // read integer
23                while(isdigit(*pString = FileObj.get()))
24                   pString++;    // increment the pointer
25                FileObj.putback(*pString);//return char to stream
26                *pString = '\0'; // null terminated string
27                cout << "Integer: " << atoi(sLine);
28            } // end if
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 6

```
29            else if(isalpha(cCharacter)) { // read a string
30                    while(isalpha(*pString=FileObj.get()))
31                         pString++;
32                      // return char to stream
33                    FileObj.putback(*pString);
34                    *pString = '\0';//null terminated string
35                    cout << "String: " << sLine;
36            } // end else if
37            else
38                FileObj.get(); // ignore
39
40            cout << '\n';
41
42        } while(!FileObj.eof());
43
44        FileObj.close();
45
46        return 0;
47    }
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 7

Convert Spaces or Tabs to '*'

```
1   #include <iostream>     // Example 5-9
2   #include <fstream>
3
4   using namespace std;
5
6   int main(int argc, char *argv[])
7   {
8       if(argc!=3) {
9           cout << "Usage: CONVERT <input> <output>\n";
10           return 1;
11       }
12
13       ifstream InFile(argv[1]); // open input file
14       ofstream OutFile(argv[2]);  // create output file
15
16       if(!OutFile) {
17           cout << "Cannot open output file.\n";
18           return 1;
19       }
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 7

```
21        if(!InFile) {
22            cout << "Cannot open input file.\n";
23            return 1;
24        }
25
26        char cCharacter;
27
28        InFile.unsetf(ios::skipws);  // do not skip spaces
29
30        while(!InFile.eof()) {
31            InFile >> cCharacter;
32
33            if(cCharacter == ' ' || cCharacter == '\t')
34                cCharacter = '*';
35            OutFile << cCharacter;
36        } // end while loop
37
38        return 0;
39    }
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Stream Operating Modes

Mode Name	Operation
ios::app	Appends data to the file.
ios::ate	When first opend, positions file at end-of-file (ate stands for at end)
ios::in	Opens file for reading
ios::nocreate	Fails to open file if it does not already exist.
ios::noreplace	If file exists, open for output fails unless ios::app or ios::ate is set
ios::out	Opens file for writing.
ios::trunc	Truncates file if it already exists.

- These enumerators can be combined with the bitwise OR operator (|)
- Overloaded ! operator returns non-zero (true) if either the failbit or badbit is set for the stream as a result of the open operation
- Extraction operator >> on cin returns 0 when eof has been encountered, so loop terminates
- Insertion operator << on file object writes formatted output to the object (which will accept any stream format manipulators)
- File object is destroyed (destructor is invoked) either explicitly by close member function, or implicitly when it goes out of scope

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHT

Unit 8

Friend Functions

- Keyword "**friend**" is preceded with function prototype.
- Standard C functions or class member functions can be friend with another class.
- Friend function provides facility to access private data member of a class.
- Friend function can be placed anywhere in the class (private section or public section)
- Scope of a friend function is the outermost scope of the module.
- Scope resolution operator is not needed because it is not a member of a class.

```
1   #include <iostream>    // Example 5-10
2   #include <iomanip>
3   using namespace std;
5   class CSpoons;    // Forward referencing
7   class CCups {
8       public:
9           CCups(int num = 1) { m_nQuantity = num; }
10           int GetTotal() { return m_nQuantity; }
11           friend int Add(const CCups&, const CSpoons&);
12       private:
13           int m_nQuantity;
14   };
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 8

```
16   class CSpoons {
17       public:
18           CSpoons(int num = 1) { m_nQuantity = num; }
19           int GetTotal() { return m_nQuantity; }
20           friend int Add(const CCups&, const CSpoons&);
21       private:
22           int m_nQuantity;
23   };
24
25   // the friend function
26   int Add(const CCups& Obj1, const CSpoons& Obj2)
27   {
28       return(Obj1.m_nQuantity + Obj2.m_nQuantity);
29   }
30
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 8

```
33    int main()
34   {
35       CCups cupObject(6);
36       CSpoons spoonObject(6);
37
38       cout << "\nCups Total   = " << cupObject.GetTotal();
39       cout << "\nSpoons Total = " << spoonObject.GetTotal();
40       cout << "\nTotal inventory = ";
41       cout << Add(cupObject, spoonObject) << "\n";
42   }
43
```

OUTPUT:

Cups Total = 6
Spoons Total = 6
Total inventory = 12

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 9

Friendship Between Member Function and Another Class

- A member function of one class can be a friend with another class

```
0   #include <iostream>    // Example 5-11
1   using namespace std;
2
3   class CTruck;     // Forward referencing
4
5   class CCar {
6       public:
7           CCar(int p, int s) { m_nPassengers = p; m_nSpeed = s;}
8           int sp_greater(CTruck &t);
9       private:
10           int m_nPassengers;
11           int m_nSpeed;
12   };
13
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 9

```
14   class CTruck {
15       public:
16           CTruck(int w, int s) {m_nWeight = w, m_nSpeed = s;}
17           // scope resolution tell the compiler that the
18           // function sp_greater() is a member of the car class.
19           friend int CCar::sp_greater(CTruck &t);
20       private:
21           int m_nWeight;
22           int m_nSpeed;
23   };
24
25   //Return positive if car speed is faster than the truck.
26   //Return 0 if speeds are the same.
27   //Return negative if truck speed is faster than the car.
28   //Only truck object is passed because
29   //sp_greater() is member of car class.
30   int CCar::sp_greater(CTruck &t)
31   {
32       return(m_nSpeed - t.m_nSpeed);
33   }
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 9

```
36   int main()
37   {
38       int t;
39       CCar c1(6,55), c2(2, 120);
40       CTruck t1(10000, 55), t2(20000, 72);
41
42       cout << "Comparing c1 and t1: \n";
43       // evoke as member function of car
44       t = c1.sp_greater(t1);
45
46       if (t < 0) cout << "Truck is faster. \n";
47       else if (t == 0) cout << "Car and truck speed is the same. \n";
48       else cout << "Car is faster. \n";
49
50       cout << "Comparing c2 and t2:\n";
51       t = c2.sp_greater(t2);    // passing objects
52       if (t < 0) cout << "Truck is faster. \n";
53       else if (t == 0) cout << "Car and truck speed is the same. \n";
54       else cout << "Car is faster. \n";
55   }
```

OUTPUT:
Comparing c1 and t1:
Car and truck speed is the same.
Comparing c2 and t2:
Car is faster.

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 10

Friend Class

- Define a class as friend of another class, granting an access to the private members of the second class.

Friend Class

```
0   #include <iostream>    // Example 5-12
1   #include <iomanip>
2
3   using namespace std;
4
5   class CQuadratic;     // Forward referencing
6
7   class CComplex {
8       private:
9           float m_fReal;
10          float m_fImag;
11          friend CQuadratic;
12   };
13
```

ALL RIGHTS RESERVED SULEMAN SAYA COPYRIGHTED MATERIAL

Unit 10

```
14   class CQuadratic {
15       public:
16           CQuadratic(float r, float i);
17           void Print();
18       private:
19           CComplex m_Number;
20   };
21
22   // Constructor Function for CQuadratic Class
23   CQuadratic::CQuadratic(float r, float i)
24   {
25       m_Number.m_fReal = r;
26       m_Number.m_fImag = i;
27   }
28
```

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL

Unit 10

```
29   void CQuadratic::Print()
30   {
31       //floats to print as fixed point
32       cout.setf(ios::fixed, ios::floatfield);
33       cout.setf(ios::showpoint); // always have decimal point
34       cout << " Real Number: ";
35       cout << setprecision(2) << m_Number.m_fReal;
36       cout << " Imaginary Number:";
37       cout << setprecision(2) << m_Number.m_fImag;
38       cout << "\n";
39   }
40
41 int main()
42 {
43   CQuadratic Q(9.0,5.5);
44   Q.Print();
45 }
```

OUTPUT:

Real Number: 9.00
Imaginary Number:5.50

ALL RIGHTS RESERVED
SULEMAN SAYA
COPYRIGHTED MATERIAL