

Object Oriented Programming (CS1143)

Week 6

Department of Computer Science

Capital University of Science and Technology (CUST)

Outline

- Read from a file using ifstream
- Write to a File using ofstream
- Using fstream

Files for Storing Data

- Data stored in variables, arrays, and objects are temporary; they are lost when the program terminates.
- To store the data created in a program permanently, you must save them in a file on a permanent storage medium, such as a disk.
- C++ provides the `ifstream`, `ofstream`, and `fstream` classes for processing and manipulating files.
- These classes are defined in the `<fstream>` header file.
- The `ifstream` class is for reading data from a file, the `ofstream` class is for writing data to a file, and the `fstream` class can be used for reading and writing data in a file.

Streams

- An input object in C++ is called an input stream and an output object is called an output stream.
- cin (console input) is a predefined object for reading input from the keyboard
- cout (console output) is a predefined object for outputting characters to the console.
- Now we will learn how to read/write data from/to files

Absolute and Relative File Names

- An absolute file name contains a file name with its complete path and drive letter.
- For example, `c:\example\scores.txt` is the absolute file name for the file `scores.txt` on the Windows operating system.
- A relative file name is relative to its current working directory.
- For example, `"scores.txt"` is a relative file name.
- We can use `"./Directory Name/File Name"` as a relative path and store the file in a directory in the same folder where our program is stored.

Writing to Files

ofstream

- The ofstream class can be used to write primitive data-type values, arrays, strings etc. to a text file.

Program 1

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main()
6  {
7      ofstream output;
8
9      // Create a file
10     output.open("./Files/scores.txt");
11
12     // Write two lines
13     output << "John" << " " << "T" << " " << "Smith" << " " << 90 << endl;
14     output << "Eric" << " " << "K" << " " << "Jones" << " " << 85 << endl;
15
16     //close the file
17     output.close();
18
19     cout << "Done" << endl;
20     return 0;
21 }
```


Description

- line 2 includes the required header file.
- Line 7 creates an object, output, from the ofstream class
- Line 10 opens a file named scores.txt for the output object.
- If the file does not exist, a new file is created. If the file already exists, its contents are destroyed without warning.
- Lines 13–14 write strings and numeric values to output
- The close() function (line 17) must be used to close the stream for the object. Without it the data may not be saved properly in the file.

```
output << "John" << " " << "T" << "Smith" << " " << 90 << endl;
```

scores.txt
file

```
John T Smith 90  
Eric K Jones 85
```

```
output << "Eric" << " " << "K" << "Jones" << " " << 85 << endl;
```

An alternate way to open an output stream

```
ofstream output("scores.txt");
```

- This statement is equivalent to

```
ofstream output;  
output.open("scores.txt");
```

Reading Data

ifstream

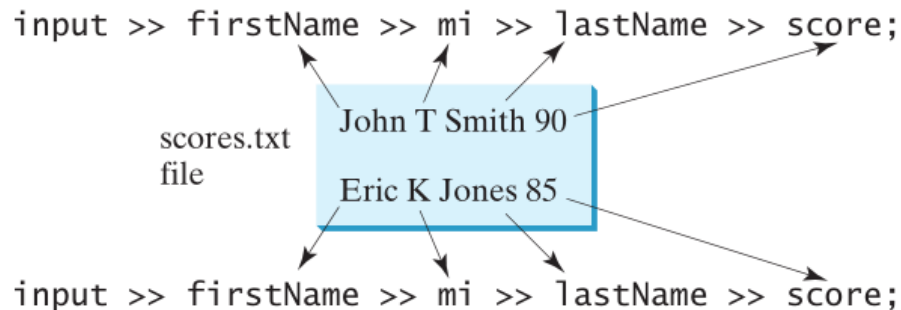
- The ifstream class can be used to read data from a text file.

Program 2

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  int main()
7  {
8      ifstream input("./Files/scores.txt");
9
10     // Variables to Store Read data
11     string firstName;
12     char mi;
13     string lastName;
14     int score;
15
16     input >> firstName >> mi >> lastName >> score;
17     cout << firstName << " " << mi << " " << lastName << " " << score << endl;
18
19     input >> firstName >> mi >> lastName >> score;
20     cout << firstName << " " << mi << " " << lastName << " " << score << endl;
21
22     input.close();
23
24     cout << "Done" << endl;
25     return 0;
26 }
```

Description

- The program creates an instance of ifstream and reads data from the file scores.txt
- Line 8 creates an object, input , from the ifstream class for file scores.txt.
- Lines 16 and 17 read strings and numeric values from the input file
- The close() function (line 22) is used to close the stream for the object.
- It is not necessary to close the input file, but doing so is a good



Important Point

- To read data correctly, you need to know exactly how data are stored.
- For example, the previous program would not work if the file contained score as a double value with a decimal point

Testing File Existence

- If the file does not exist when reading a file, your program will run and produce incorrect results.
- You can invoke the `fail()` function immediately after invoking the `open` function.
- If `fail()` returns `true` , it indicates that the file does not exist.

Program 3

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  int main()
7  {
8      ifstream input("../Files/scores.txt");
9
10     if (input.fail())
11     {
12         cout << "File does not exist. Exiting...." << endl;
13         return 0;
14     }
15
16     cout<<"The file exists"<<endl;
17
18     return 0;
19 }
```

Testing end of file

- If you don't know how many lines are in the file and want to read them all you can invoke the `eof()` function on the input object to detect it.

Program 4

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main()
6  {
7      ifstream input("./Files/p4.txt");
8
9      double sum = 0;
10     double number;
11
12     while (!input.eof()) // Continue if not end of file
13     {
14         input >> number; // Read data
15         cout << number << " "; // Display data
16         sum += number;
17     }
18     input.close();
19
20     cout << "\nSum is " << sum << endl;
21     return 0;
22 }
```

Description

- If there is no extra character or line at the end, the output will be correct.
- If there is an extra character or an extra line at the end it will read the last element twice
- The reason for this is that when the last number 10 is read, the file system does not know it is the last number because there are blank characters after the last number. So, the eof() function returns false
- When the program reads the number again, the eof() function returns true, but the variable number is not changed, because nothing is read from the file.
- The variable number still contains the value 10, which is added again to sum

```
1 1.0
2 2.0 3.0
3 4.0
4 5.0
5 6.0
6 7.0 8.0 9.0 10.0
```

```
D:\Work Umair\4_CUST (1-9-22)\1_Teaching\2_ACS
1 2 3 4 5 6 7 8 9 10
Sum is 55
```



```
1 1.0
2 2.0 3.0
3 4.0
4 5.0
5 6.0
6 7.0 8.0 9.0 10.0
7
```

```
D:\Work Umair\4_CUST (1-9-22)\1_Teaching\2_ACS1143-OOP\
1 2 3 4 5 6 7 8 9 10 10
Sum is 65
-----
```



Solution (P5)

```
1 1.0
2 2.0 3.0
3 4.0
4 5.0
5 6.0
6 7.0 8.0 9.0 10.0
7
```

D:\Work Umair\4_CUST (1-9-22)\1_Teaching\2_AC

```
1 2 3 4 5 6 7 8 9 10
Sum is 55
```



```
1 #include <iostream>
2 #include <fstream>
3 using namespace std;
4
5 int main()
6 {
7     ifstream input("./Files/p4.txt");
8
9     double sum = 0;
10    double number;
11
12    while (input >> number) // Continue if not end of file
13    {
14        cout << number << " "; // Display data
15        sum += number;
16    }
17    input.close();
18
19    cout << "\nSum is " << sum << endl;
20    return 0;
21 }
```

Description

- The statement `input >> number` is actually to invoke an operator function.
- This function returns an object if a number is read; otherwise it returns `NULL` .
- `NULL` is a constant value `0` . C++ automatically casts it to a bool value `false` when it is used as a condition in a loop statement or a selection statement.
- If no number is read from the input stream, `input >> number` returns `NULL` and the loop terminates.

getline() function


- There is a problem in reading data using the stream extraction operator (>>) as data are delimited by whitespace.
- What happens if the whitespace characters are part of a string?
- We can use a function getline() for this purpose.

getline(ifstream& input, int string s, char delimiterChar)

- The function stops reading characters when the delimiter character or end-of-file mark is encountered.
- The third argument delimiterChar has a default value '\n'. That means it will read lines by default.

```
1 New York#New Mexico#Texas#Indiana
```

Program P6

 D:\Work Umair\4_CUST (1-9-22)\1_Teaching

New York
New Mexico
Texas
Indiana
Done

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5
6 int main()
7 {
8     // Open a file
9     ifstream input("./Files/state.txt");
10
11     string city;
12
13     while (!input.eof()) // Continue if not end of file
14     {
15         getline(input, city, '#'); //read till # encountered
16         cout << city << endl;
17     }
18
19     input.close();
20     cout << "Done" << endl;
21     return 0;
22 }
```

Process exited after
Press any key to con

get() and put()

- Two other useful functions are get and put .
- You can invoke the get function on an input object to read a character and invoke the put function on an output object to write a character.
- The program on the next slide prompts the user to enter a file and copies it to a new file.

```

1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  int main()
7  {
8      cout << "Enter a source file name: ";
9      string inputFilename;
10     cin >> inputFilename;
11
12     cout << "Enter a target file name: ";
13     string outputFilename;
14     cin >> outputFilename;
15
16     // Create input and output streams
17     ifstream input(inputFilename.c_str());
18     ofstream output(outputFilename.c_str());
19
20     char ch = input.get();
21     while (!input.eof()) // Continue if not end of file
22     {
23         output.put(ch);
24         ch = input.get(); // Read next character
25     }
26     input.close();
27     output.close();
28     cout << "\nCopy Done" << endl;
29     return 0;
30 }

```

Description

- The program prompts the user to enter a source file name and a target file name
- An input object for `inputFilename` is created in line 17 and an output object for `outputFilename` in line 18.
- File names must be C-strings. `inputFilename.c_str()` returns a C-string from string `inputFilename` .
- Lines 21–25 read characters repeatedly one at a time using the `get` function and write the character to the output file using the `put` function.

fstreams

fstreams

- We used ofstream to write data and ifstream to read data.
- We can use the fstream class to create an input or output stream.
- To open an fstream file, you have to specify a file open mode to tell C++ how the file will be used.

| <i>Mode</i> | <i>Description</i> |
|--------------------------|---|
| <code>ios::in</code> | Opens a file for input. |
| <code>ios::out</code> | Opens a file for output. |
| <code>ios::app</code> | Appends all output to the end of the file. |
| <code>ios::ate</code> | Opens a file for output. If the file already exists, move to the end of the file. Data can be written anywhere in the file. |
| <code>ios::trunc</code> | Discards the file's contents if the file already exists. (This is the default action for <code>ios::out</code> .) |
| <code>ios::binary</code> | Opens a file for binary input and output. |

Program P8

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5
6 int main()
7 {
8     fstream inout;
9
10    inout.open("./Files/city.txt", ios::out); //create a file
11    inout << "Dallas" << " " << "Houston" << " " << "Atlanta" << " "; //write cities
12    inout.close();
13
14    inout.open("./Files/city.txt", ios::out | ios::app); // Append to the file
15    inout << "Savannah" << " " << "Austin" << " " << "Chicago"; // Write cities
16    inout.close();
17
18    string city;
19    inout.open("./Files/city.txt", ios::in); // Open the file
20    while (!inout.eof()) // Continue if not end of file
21    {
22        inout >> city;
23        cout << city << " ";
24    }
25    inout.close();
26
27    return 0;
28 }
```

Description

- The program creates an `fstream` object in line 8
- It opens the file “city.txt” for output using the file mode `ios::out` in line 10. After writing data the program closes the stream
- The program uses the same stream object to reopen the text file with the combined modes `ios::out | ios::app` in line 14. The program then appends new data to the end of the file
- Finally, the program uses the same stream object to reopen the text file with the input mode `ios::in` in line 19. The program then reads all data from the file.