CSE 1062 **Fundamentals of Programming**

Lecture #11

Spring 2016

Computer Science & Engineering Program
The School of EE & Computing
Adama Science & Technology University

ASTU

Files and Streams

- I/O file stream objects and functions
- Reading and writing character-based files
- Random file access
- File streams as function arguments

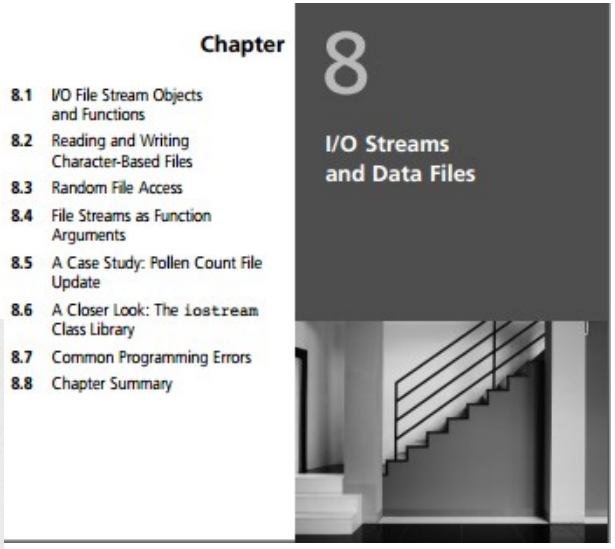
Case Study: Weather Forecast File Updates

- Reading Assignments
  - Chapter 8 of the text book
  - About ASCII Codes
    - http://www.theasciicode.com.ar

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ



I/O File Stream Objects and Functions

ASTU

- To store and retrieve data outside a C++ program, two items are needed:
  - A file
  - A file stream object
- A file is a collection of data stored toget her under a common name, usually on d isk, magnetic tape, USB drive, or CD
- Each file has a unique file name, referred to as file's **external name**

I/O File Stream Objects and Functions

ASTU

- Choose filenames that indicate the type of data in the file
- Two basic types of files exist
  - **Text files**
    - (also known as **character-based** files)
  - **Binary files**

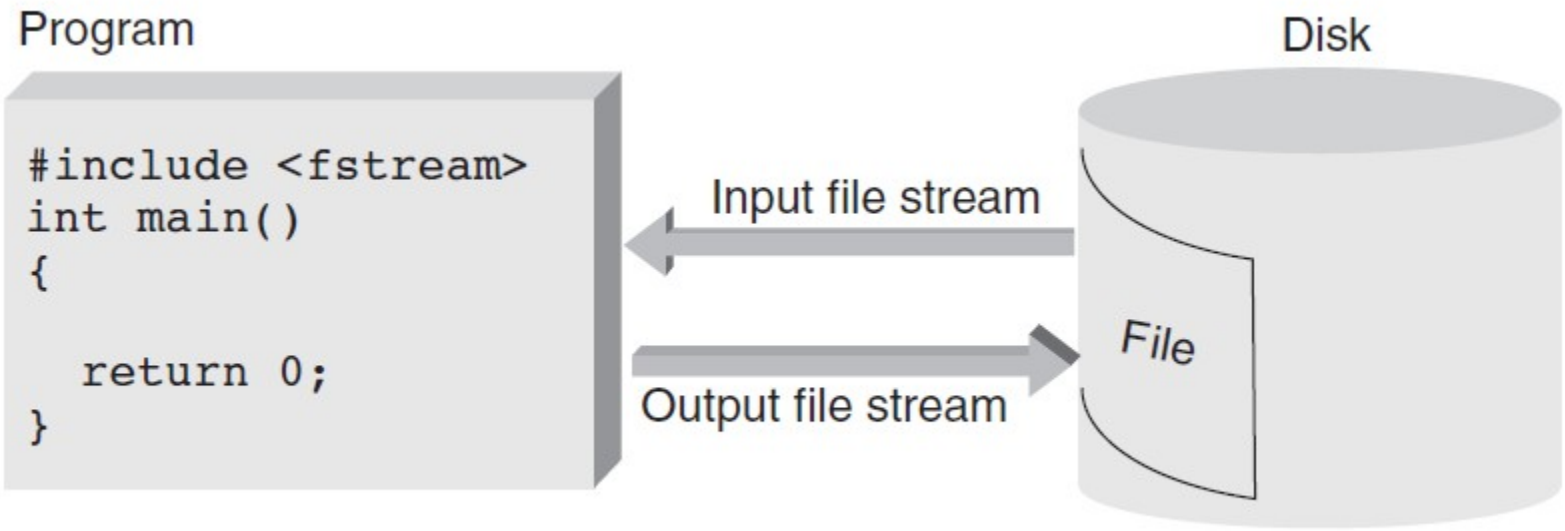
File Stream Objects

- **File stream:** A one-way transmission path used to connect a file stored on a physical device, such as a disk or CD, to a program
- Each file stream has its own mode that determines direction of data on transmission path
- That is, whether path moves data from a file to a program or from a program to a file
- **Input file stream:** File stream that receives or reads data from a file to a program
- **Output file stream:** File stream that sends or writes data to a file

File Stream Objects

ASTU

- For each file your program uses, regardless of file's type, a distinct file stream object must be created



File Stream Functions

- Each file stream object has access to functions defined for its class
- Methods perform following functions:
  - Connecting stream object name to external filename: **opening a file**
  - Determining whether successful connection has been made
  - Closing connection: **closing a file**
  - Getting next data item into program from input stream
  - Putting new data item from program onto output stream

File Stream Functions

- When existing file is connected to input stream, file's d ata is made available for input, starting with first data i tem in file
  - Called **read mode** or **input mode**
- File connected to output stream creates new file and makes file available for output
  - Called **output mode**
- When opening file for input or output, check that con nection has been established before attempting to use file

File Stream Functions

ASTU

Prototype	Description
`fail()`	Returns a Boolean `true` if the file hasn't been opened successfully; otherwise, returns a Boolean `false` value.
`eof()`	Returns a Boolean `true` if a read has been attempted past the end-of-file; otherwise, returns a Boolean `false` value. The value becomes `true` only when the first character after the last valid file character is read.
`good()`	Returns a Boolean `true` value while the file is available for program use. Returns a Boolean `false` value if a read has been attempted past the end-of-file. The value becomes `false` only when the first character after the last valid file character is read.
`bad()`	Returns a Boolean `true` value if an error occurs that results in data loss when reading from or writing to a stream; otherwise, returns a `false`.

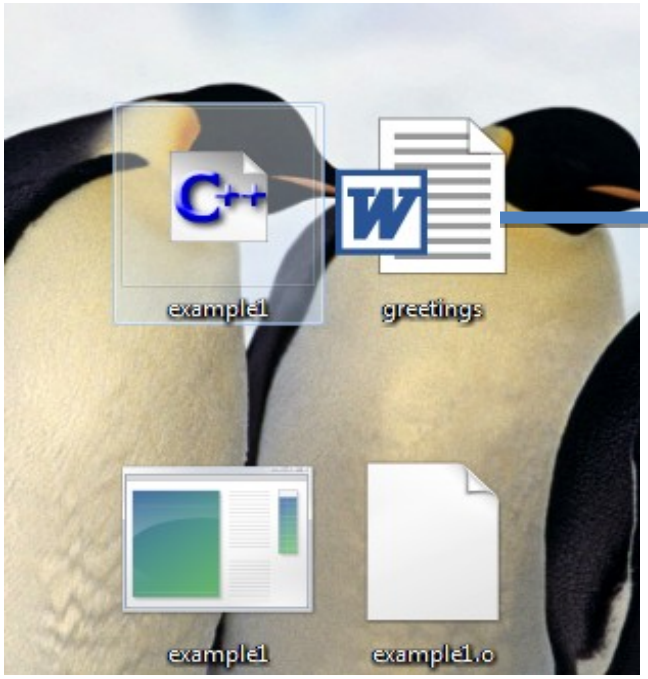
Example 1

ASTU

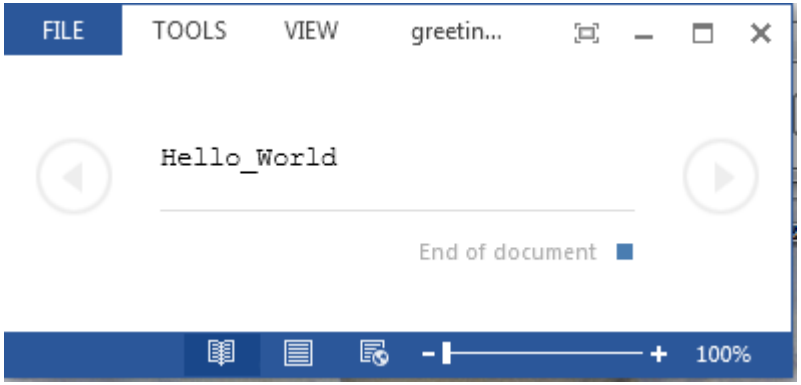
```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdlib> // needed for exit()
4  #include <string>
5  using namespace std;
6  int main()
7  {
8      string filename = "greetings.rtf"; // place the filename up front
9      ofstream outFile;
10     outFile.open(filename.c_str()); // open the file
11     if (outFile.fail()) // check for successful open
12     {
13         cout << "\nThe file named " << filename
14              << " was not successfully opened"
15              << "\n Please check that the file currently exists."
16              << endl;
17         exit(1);
18     }
19     cout << "\nThe file has been successfully opened for writing.\n";
20     outFile<<"Hello_World";
21     return 0;
22 }
```

```
C:\Users\Tinsae\Desktop\example1.exe

The file has been successfully opened for writing.

Process returned 0 (0x0)   execution time : 0.166 s
Press any key to continue.
```

Desktop



File Opened



Example 2

ASTU

```
#include <iostream>
#include <fstream>
#include <cstdlib> // needed for exit()
#include <string>
using namespace std;
int main()
{
    string filename,retrieved;
    ifstream inFile;
    cout << "Enter the name of the file you wish to open \n";
    cin >> filename;
    inFile.open(filename.c_str()); // open the file
    if (inFile.fail()) // check for successful open
    {
        cout << "\nThe file named " << filename
             << " was not successfully opened"
             << "\n Please check that the file currently exists."
             << endl;
        exit(1);
    }
    cout << "\nThe file has been successfully opened for reading.\n";
    inFile>>retrieved;
    cout<<"found the following:\n"<<retrieved;
    return 0;
}
```

Example 2: Testing

ASTU

```
C:\Users\Tinsae\Desktop\example2.exe
Enter the name of the file you wish to open
greetings

The file named greetings was not successfully opened
 Please check that the file currently exists.

Process returned 1 (0x1)   execution time : 3.949 s
Press any key to continue.
```

```
C:\Users\Tinsae\Desktop\example2.exe
Enter the name of the file you wish to open
greetings.rtf

The file has been successfully opened for reading.
found the following:
Hello_World
Process returned 0 (0x0)   execution time : 32.750 s
Press any key to continue.
```

Closing a File

ASTU

- File is closed using `close()` method
- This method breaks connection between file's external name and file stream, which can be used for another file
- Because all computers have limit on maximum number of files that can be open at one time, closing files no longer needed makes good sense
- Any open files existing at end of normal program execution are closed automatically by OS

- Reading or writing character-based files involves almost identical operations for reading input from keyboard and writing data to screen
  - For writing to a file, `cout` object is replaced by `ofstream` object name declared in program
  - Reading data from text file is almost identical to reading data from standard keyboard, except `cin` object is replaced by `ifstream` object declared in program

Example 3

ASTU

```
#include <iostream>
#include <fstream>
#include <cstdlib> // needed for exit()
#include <string>
#include <iomanip>  // needed for formatting
using namespace std;
int main()
{
    string filename = "prices.dat"; // put the filename up front
    ofstream outFile;
    outFile.open(filename.c_str());
    if (outFile.fail())
    {
        cout << "The file was not successfully opened" << endl;
        exit(1);
    }
// Set the output file stream formats
    outFile << setiosflags(ios::fixed)
            << setiosflags(ios::showpoint)
            << setprecision(2);
// Send data to the file
    outFile << "Mats "  << 39.95 << endl
            << "Bulbs " << 3.22  << endl
            << "Fuses " << 1.08  << endl;
    outFile.close();
    cout << "The file " << filename
         << " has been successfully written." << endl;
    return 0;
}
```

```
"C:\Users\Tinsae\Desktop\example 3.exe"
The file prices.dat has been successfully written.

Process returned 0 (0x0)   execution time : 0.185 s
Press any key to continue.
```

Reading from a Text File

Function Name	Description
`get()`	Returns the next character extracted from the input stream as an `int`.
`get(charVar)`	Overloaded version of `get()` that extracts the next character from the input stream and assigns it to the specified character variable, `charVar`.
`getline(fileObject, strObj, termChar)`	Extracts characters from the specified input stream, `fileObject`, until the terminating character, `termChar`, is encountered. Assigns the characters to the specified string class object, `strObj`.
`peek()`	Returns the next character in the input stream without extracting it from the stream.
`ignore(int n)`	Skips over the next *n* characters. If *n* is omitted, the default is to skip over the next single character.

Example 4

ASTU

```
#include <iostream>
#include <fstream>
#include <cstdlib> // needed for exit()
#include <string>
using namespace std;
int main()
{
    string filename = "prices.dat"; // put the filename up front
    string descrip;
    double price;
    ifstream inFile;
    inFile.open(filename.c_str());
    if (inFile.fail()) // check for successful open
    {
        cout << "\nThe file was not successfully opened"
             << "\n Please check that the file currently exists."
             << endl;
        exit(1);
    }
// Read and display the file's contents
    inFile >> descrip >> price;
    while (inFile.good()) // check next character
    {
        cout << descrip << ' ' << price << endl;
        inFile >> descrip >> price;
    }
    inFile.close();
    return 0;
}
```

```
C:\Users\Tinsae\Desktop\example4.exe
Mats 39.95
Bulbs 3.22
Fuses 1.08

Process returned 0 (0x0)   execution time : 0.158 s
Press any key to continue.
```

Standard Device Files

- **Logical file object:** Stream that connects a file o f logically related data to a program
- **Physical file object:** Stream that connects to har dware device such as keyboard, screen, or print er
- Actual physical device assigned to your progra m for data entry is formally called **standard inp ut file**
  - `cin` method calls are routed to this standard input fi le
  - `cout` method calls are written to a device that has b een assigned as standard output file

Random File Access

- **File access:** Refers to process of retrieving dat a from a file
- Two types of file access
  - Sequential file access
  - Random file access
- **File organization:** Refers to the way data is st ored in a file
- The files you have used and will continue to u se have a sequential organization, meaning c haracters in file are stored in a sequential ma nner

Random File Access

- Each open file has been read in a sequen tial manner, meaning characters are acc essed one after another, which is called **sequential access**
  - Although characters are stored sequentially, they don't have to be accessed in same way

Random File Access

- In **random access**, any character in opened file can be read without having to read all characters stored ahead of it first
  - To provide random access, each `ifstream` object creates a file position marker automatically
  - This marker is a long integer representing an offset from the beginning of file

Random File Access

ASTU

- File Position Marker Functions

Name	Description
`seekg(offset, mode)`	For input files, move to the offset position indicated by the mode.
`seekp(offset, mode)`	For output files, move to the offset position indicated by the mode.
`tellg(void)`	For input files, return the current value of the file position marker.
`tellp(void)`	For output files, return the current value of the file position marker.

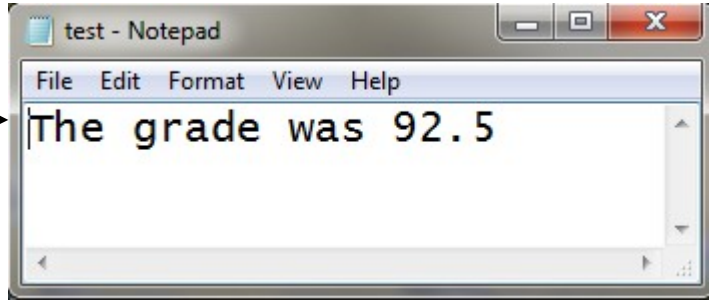
- `seek( )` method allows programmer to move to any position in file
- Character's position is referred to as its **offset** from the start of file

Example 5: Using seekg( ) and tellg( )

ASTU

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <cstdlib>
5  using namespace std;
6  int main(){
7      string filename = "test.dat";
8      char ch;
9      long offset, last;
10     ifstream inFile(filename.c_str());
11     if (inFile.fail()) // check for successful open
12     {
13         cout << "\nThe file was not successfully opened"
14              << "\n Please check that the file currently exists"
15              << endl;
16         exit(1);
17     }
18     inFile.seekg(0L,ios::end); // move to the end of the file
19     last = inFile.tellg(); // save the offset of the last character
```

Suppose test.dat contains this text



Example 5: Using seekg( ) and tellg( )

ASTU

```
19      last = inFile.tellg(); // save the offset of the last character
20      cout<<"last character offset: "<<last<<endl;
21      for (offset = 1L; offset <= last; offset++)
22      {
23          inFile.seekg(-offset,ios::end);
24          ch = inFile.get();
25          cout << ch << " : ";
26      }
27      inFile.close();
28      cout << endl;
29      return 0;
30  }
```

-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	h	e		g	r	a	d	e		w	a	s		9	2	.	5	EOF

```
C:\Users\Tinsae\Desktop\example5.exe
last character offset: 18
5 : . : 2 : 9 :   : s : a : w :   : e : d : a : r : g :   : e : h : T :

Process returned 0 (0x0)   execution time : 0.104 s
Press any key to continue.
```

File Streams as Function Arguments

- A file stream object can be used as a function argument
- The function's formal parameter must be a reference to the appropriate stream, either `ifstream&` or `ofstream&`
  - Examples: `inOut(), getOpen()`

Example 7: File Streams as Function Arguments

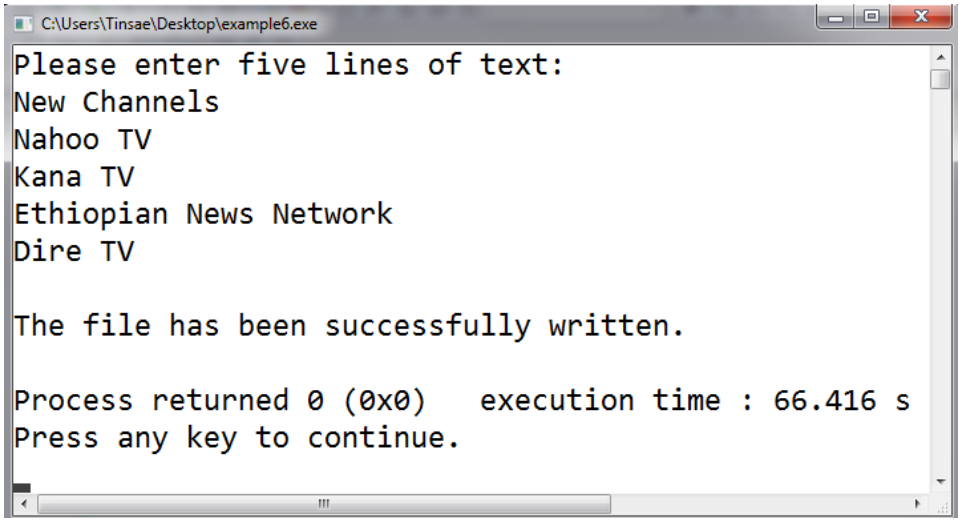
ASTU

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdlib>
4  #include <string>
5  using namespace std;
6  int main()
7  {
8      string fname = "list.dat"; // the file you are working with
9      void inOut(ofstream&); // function prototype
10     ofstream outFile;
11     outFile.open(fname.c_str());
12     if (outFile.fail()) // check for a successful open
13     {
14         cout << "\nThe output file " << fname
15              << " was not successfully opened"
16              << endl;
17         exit(1);
18     }
19     inOut(outFile); // call the function
20     return 0;
21 }
```

Example 7: File Streams as Function Arguments

ASTU

```
22  void inOut(ofstream& fileOut)
23  {
24      const int NUMLINES = 5; // number of lines of text
25      string line;
26      int count;
27      cout << "Please enter five lines of text:" << endl;
28      for (count = 0; count < NUMLINES; count++)
29      {
30          getline(cin,line);
31          fileOut << line << endl;
32      }
33      cout << "\nThe file has been successfully written." << endl;
34      return;
35  }
```



Case Study: Weather Forecast File Updates

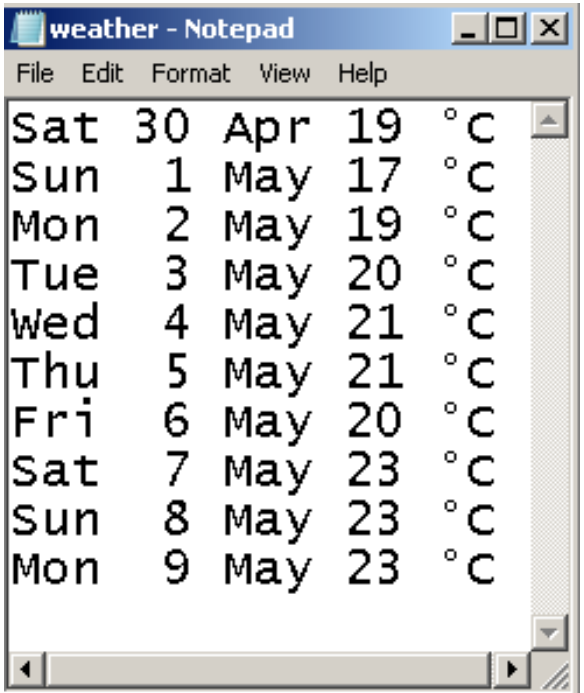
- After a data file has been created, application programs are typically written to read and update the file with current data
- In this case study, a file is used as a database storing the **ten most** recent temperature forecasts of Addis Ababa
  - Analyze the problem
  - Develop a solution
  - Code the solution
  - Test and correct the program

Case Study: Weather Forecast File Updates

ASTU

- Analyze the Problem
  - Data Obtained from http://www.timeanddate.com/weather/ethiopia/addis-ababa/ext
  - A file containing the ten most recent forecasts is created
  - File Name and Extension: weather.in

Only Higher Temperatures



Oldest

Recent

Day	Temperature
Sat, 30 Apr	12 / 19 °C
Sun, 1 May	13 / 17 °C
Mon, 2 May	12 / 19 °C
Tue, 3 May	11 / 20 °C
Wed, 4 May	10 / 21 °C
Thu, 5 May	11 / 21 °C
Fri, 6 May	10 / 20 °C
Sat, 7 May	12 / 23 °C
Sun, 8 May	12 / 23 °C
Mon, 9 May	11 / 23 °C
Tue, 10 May	11 / 22 °C
Wed, 11 May	11 / 23 °C
Thu, 12 May	11 / 21 °C
Fri, 13 May	10 / 21 °C
Sat, 14 May	11 / 22 °C

Case Study: Weather Forecast File Updates

- Analyze the Problem
  - The input data for this problem consists of
    - a file of 10 daily weather forecasts
      - Each forecast contains
        - » a day (Mon-Sun), date (1-31), month(Jan-Dec), tem perature(0-35)
    - a user-input value of the most recent weather fore cast. It contains
        - » a day (Mon-Sun), date (1-31), month(Jan-Dec), tem perature(0-35)
  - There are two required outputs:
    - A file of the 10 most recent daily forecasts values
    - The average of the data in the updated file

Case Study: Weather Forecast File Updates

ASTU

- Develop a Solution

main() function

Display a message indicating what the program does
Call the Input stream function
Call the Output stream function
Call the Update function
Display the new top ten recent daily weather forecasts

Input stream function

Request the name of the input data file
Open an input file stream and validate a successful connection
Output stream function
Request the name of the output data file
Open an output file stream and validate a successful connection

Update function

Request a new daily weather forecast
Read the oldest daily weather forecast from the input data file
For the remaining input file daily weather forecasts:
    Read an input value
    Add the value of the temperature to a total
    Write the input value to the output file stream
End For
Write the new weather forecast to the output file stream
Add the new value of temperature (from the new weather forecast) to the total
Calculate the average as total / (number of daily weather forecasts)
Return the new 10 daily forecasts average
Close all files

Case Study: Weather Forecast File Updates

ASTU

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <iomanip>
using namespace std;
void openInput(ifstream&); // pass a reference to an ifstream
void openOutput(ofstream&); // pass a reference to an ofstream
double weatherUpdate(ifstream&, ofstream&); // pass two references
int main()
{
    ifstream inFile; // inFile is an ifstream object
    ofstream outFile; // outFile is an ofstream object
    double average;
// Display a user message
    cout << "\n\nThis program reads the old weather forecast file, "
         << "creates a current weather "
         << "\n forecast file, and calculates and displays "
         << "the latest 10 days average.";
    openInput(inFile);
    openOutput(outFile);
    average = weatherUpdate(inFile, outFile);
    cout << "\nThe new 10 days average is: " << average << endl;
    return 0;
}
```

Case Study: Weather Forecast File Updates

ASTU

```
26  // This function gets an external filename and opens the file for input
27  void openInput(ifstream& fname)
28  {
29      string filename;
30      cout << "\n\nEnter the input weather forecast filename: ";
31      cin >> filename;
32      fname.open(filename.c_str());
33      if (fname.fail()) // check for a successful open
34      {
35          cout << "\nFailed to open the file named " << filename << "for input"
36               << "\n Please check that this file exists"
37               << endl;
38          exit(1);
39      }
40      return;
41  }
```

Case Study: Weather Forecast File Updates

ASTU

```
42  // This function gets an external filename and opens the file for output
43  void openOutput(ofstream& fname)
44  {
45      string filename;
46      cout << "Enter the output weather forecast filename: ";
47      cin >> filename;
48      filename="weather.out";
49      fname.open(filename.c_str());
50      if (fname.fail()) // check for a successful open
51      {
52          cout << "\nFailed to open the file named " << filename << "for output"
53               << endl;
54          exit(1);
55      }
56      return;
57  }
```

Case Study: Weather Forecast File Updates

ASTU

```
58  // The following function reads the weather file,
59  // writes a new file,
60  // and returns the new 10 days average
61  double weatherUpdate(ifstream& inFile, ofstream& outFile)
62  {
63      const int WEATHNUMS = 10; // maximum number of forecasts
64      int i;
65      string newday,newmonth,oldday,oldmonth,day,month;;
66      int newdate,newtemp,olddate,oldtemp,date,temp;
67
68      double sum = 0;
69      double average;
70  // Get the latest weather forecast
71      cout << "Enter the latest weather forecast reading:\n";
72      cout << "[Mon-Sun] [1-30] [Jan-Dec] [0-35] ̊C \n";
73      cin>>newday>>newdate>>newmonth>>newtemp;
74
75  // Read the oldest weather forecast
76      inFile>>oldday>>olddate>>oldmonth>>oldtemp;
77  // Using Random File Access: moving to the next line
78      inFile.seekg(18L,ios::beg);
79
```

Case Study: Weather Forecast File Updates

ASTU

```
80  // Read, sum, and write out the rest of the weather forecasts
81      for (i = 1; i < WEATHNUMS; i++)
82      {
83          inFile>>day>>date>>month>>temp;
84
85  // Using Random File Access: moving to the i+1 line
86          inFile.seekg((i+1)*18L,ios::beg);
87          sum += temp;
88  // Write to the output file
89          outFile<<setw(2)<<day<<" "
90                 <<setw(2)<<date<<" "
91                 <<setw(2)<<month<<" "
92                 <<setw(2)<<temp<<" "<<"°C"<<endl;
93      }
94  //// Write out the latest forecast
95      outFile<<newday<<" "<<newdate<<" "<<newmonth<<" "<<newtemp<<" "<<"°C"<<endl;
96  //// Compute and display the new average
97      average = (sum + newtemp) / double(WEATHNUMS);
98      inFile.close();
99      outFile.close();
100     cout << "\nThe output file has been written.\n";
101     return average;
102 }
```

Testing the Program

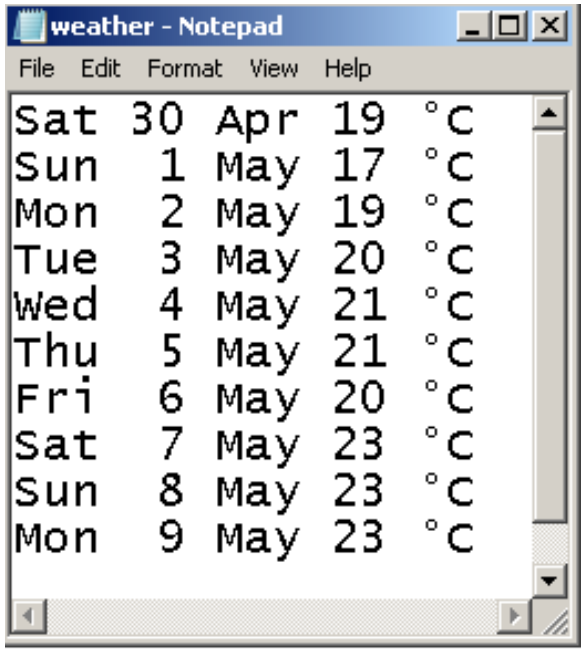
ASTU

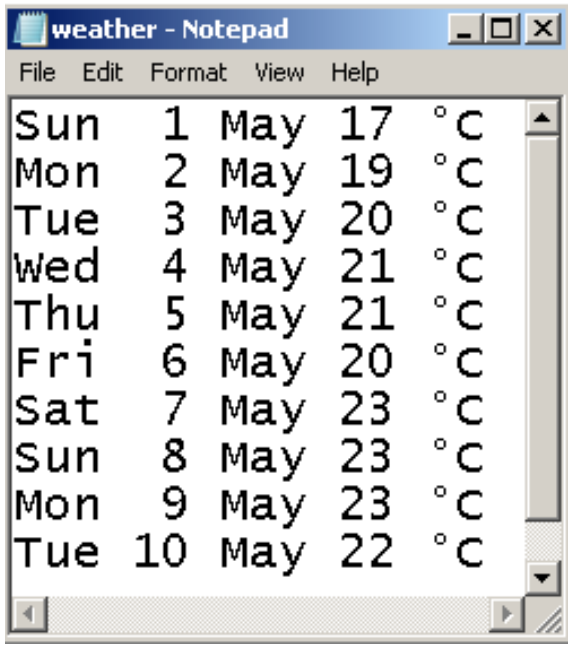
```
"C:\Users\Tinsae\Desktop\cpp scrap\pollen.exe"

This program reads the old weather forecast file, creates a current weather
 forecast file, and calculates and displays the latest 10 days average.

Enter the input weather forecast filename: weather.in
Enter the output weather forecast filename: weather.out
Enter the latest weather forecast reading:
[Mon-Sun] [1-30] [Jan-Dec] [0-35] C
Tue 10 May 22

The output file has been written.

The new 10 days average is: 20.9

Process returned 0 (0x0)   execution time : 70.773 s
Press any key to continue.
```

Testing the Program

ASTU

weather.in	weather.out





0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
S	a	t		3	0		A	p	r		1	9		°	C	**LF**	**LF**
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
S	u	n			1		M	a	y		1	7		°	C	**LF**	**LF**

Looking ASCII Codes Using get Function

ASTU

```
1  #include <iostream>
2  #include <fstream>
3  #include <iomanip>
4  using namespace std;
5  int main()
6  {
7      ifstream inFile; // inFile is an ifstream object
8      ofstream outFile; // outFile is an ofstream object
9
10     string oldday,oldmonth;
11     int olddate,oldtemp;
12     string filename="weather.in";
13     inFile.open(filename.c_str());
14
15     inFile>>oldday>>olddate>>oldmonth>>oldtemp;
16     for(long i=0L;i<=17;i++)
17     {
18         inFile.seekg(i,ios::beg);
19         cout<<setw(4)<<inFile.get(); //prints the ASCII code of characters
20     }
21
22     return 0;
23 }
```

```
"C:\Users\Tinsae\Desktop\cpp scrap\aaaa.exe"
  83   97 116   32   51   48   32   65 112 114   32   49   57   32 176   67   10   10
Process returned 0 (0x0)   execution time : 0.132 s
Press any key to continue.
```