

CSE 1062 Fundamentals of Programming

Lecture #12

Spring 2016

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



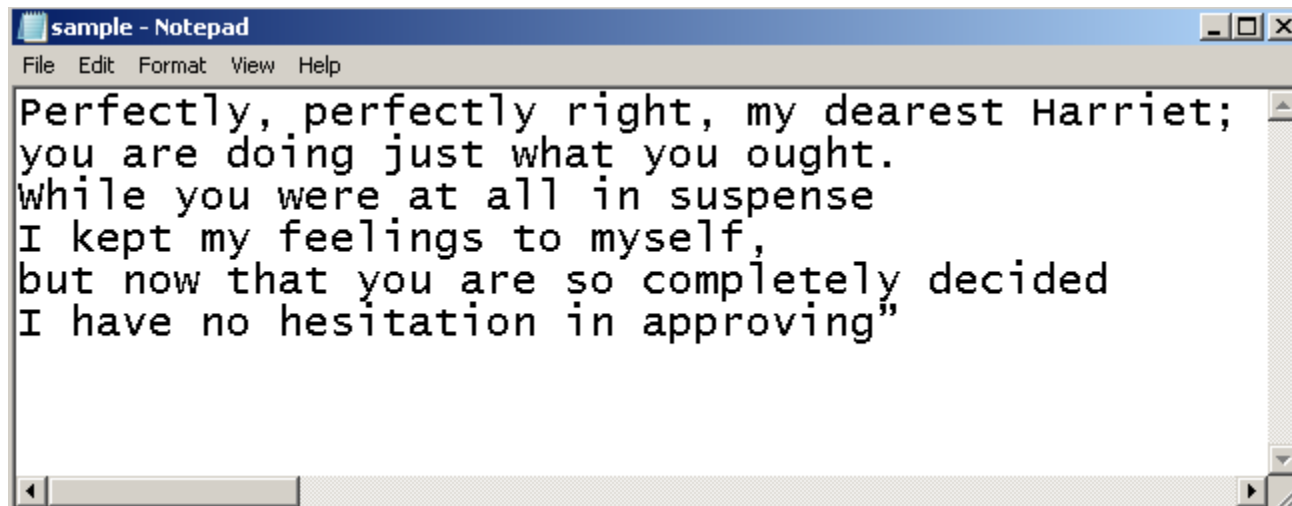


- Files and Streams Practice
 - [Data Processing] Group Average
 - Random File Access
 - Table of Logarithms
 - Word Play
 - [Data Processing] Employee Report

- Store the following data in a file,
5 96 87 78 93 21 4 92 82 85 87 6 72 69 85 75 81 73
- Write a C++ program to calculate and display the average of each group of numbers in the file created above.
- The data is arranged in the file so that each group of numbers is preceded by the number of data items in the group.

- Therefore,
 - the first number in the file, 5, indicates that the next five numbers should be grouped together.
 - The number 4 indicates that the following four numbers are a group,
 - and the 6 indicates that the last six numbers are a group.
 - (Hint: Use a nested loop. The outer loop should terminate when the end of file has been encountered.)

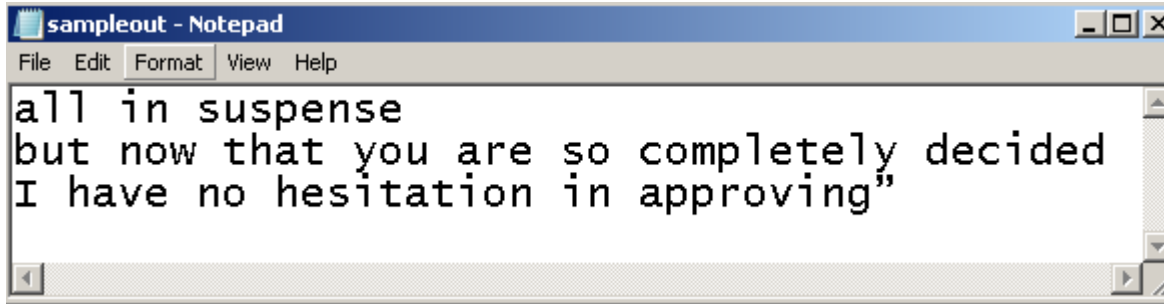
- Use the following text to create a new file in your computer, use it as an input file for this exercise
 - sample.txt



- Using a c++ program open an existing input file (**sample.txt**) and new output file (**sampleout.txt**)
 - The offset passed to seekg() and seekp() must be a long integer
 - Use eof() to check end of file
 - Use fail() to check whether files are opened successfully
 - Use inFile.tellg() to get the offset position
 - Use inFile.get(charVar) function to get character and store in a charVar
 - getline(outFile,stringVar)
 - to read a line and store in a stringVar

- In one c++ program
 - Read all text in the input file and display on the screen
 - Copy the text “all in suspense” and write it to the output file
 - Starting from “but now...” till end of input file, copy text and **append** to the output file `outFile.open(outputfile.c_str(), ios::app);`
 - Move to the end of text in the output file and get the offset and display it.

- **sampleout.txt** final outcome



```
sampleout - Notepad
File Edit Format View Help
all in suspense
but now that you are so completely decided
I have no hesitation in approving"
```

- How much offset did you get?

- Write a C++ program to generate a table of the base-10 logarithms between 1 and 10 in steps of 0.1.
 - The table should be written to a file and it should include a title describing the table and row and column headings.
 - This table should be organized as on the next slide

Table of Logarithms



ASTU

- Use loops and format descriptors

The screenshot shows a Notepad window titled "log_table" with a menu bar (File, Edit, Format, View, Help). The text inside the window is a table of base-10 logarithms. The title of the table is "Table of Base-10 Logarithms Between 1.0 and 10.0". The table has 11 columns representing the fractional part of the logarithm (0 to 0.9) and 11 rows representing the integer part (0 to 10). The first column is labeled with two asterisks (**).

**	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	0.000	0.041	0.079	0.114	0.146	0.176	0.204	0.230	0.255	0.279
2	0.301	0.322	0.342	0.362	0.380	0.398	0.415	0.431	0.447	0.462
3	0.477	0.491	0.505	0.519	0.531	0.544	0.556	0.568	0.580	0.591
4	0.602	0.613	0.623	0.633	0.643	0.653	0.663	0.672	0.681	0.690
5	0.699	0.708	0.716	0.724	0.732	0.740	0.748	0.756	0.763	0.771
6	0.778	0.785	0.792	0.799	0.806	0.813	0.820	0.826	0.833	0.839
7	0.845	0.851	0.857	0.863	0.869	0.875	0.881	0.886	0.892	0.898
8	0.903	0.908	0.914	0.919	0.924	0.929	0.934	0.940	0.944	0.949
9	0.954	0.959	0.964	0.968	0.973	0.978	0.982	0.987	0.991	0.996
10	1.000									

Word Play



- Count all words in word.txt longer than 18 letters. Print these words and the number of such words.

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {
6      ifstream inFile;
7      string filename="words.txt",word;
8      int count=0,totalCount=0;;
9      inFile.open(filename.c_str());
10     while(inFile.good())
11     {
12         inFile>>word;
13         if(word.length()>18)
14         {
15             count+=1;
16             cout<<word<<endl;
17         }
18         totalCount++;
19     }
20     cout<<"\nNo of words = "<<count<<endl;
21     cout<<"Total words= "<<totalCount<<endl;
22     inFile.close();
23     return 0;
24 }
```

- Count all words without the **letter “e”** and print the number of such words.

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstring>
4  using namespace std;
5  int main()
6  {
7      ifstream inFile;
8      string filename="words.txt", word;
9      int count=0, totalCount=0;;
10     inFile.open(filename.c_str());
11     while (inFile.good())
12     {
13         inFile>>word;
14         if (strstr(word.c_str(), "e")==NULL)
15         {
16             count+=1;
17             cout<<word<<endl;
18         }
19         totalCount++;
20     }
21     cout<<"\nNo of words = "<<count<<endl;
22     cout<<"Total words = "<<totalCount<<endl;
23     inFile.close();
24     return 0;
25 }
```

- Is there a word with **triple letters**(three of the same letters in a row)? Learn how to convert strings to char arrays

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  bool has_triple(const char*,int);
5  int main()
6  {
7      ifstream inFile;
8      string filename="words.txt",word;
9      const char* wordarray;
10     int count=0,totalCount=0,len=0;
11     inFile.open(filename.c_str());
12     while(inFile.good())
13     {
14         inFile>>word;
15         len=word.length();
16         wordarray=(word.c_str());
17         if(len<3)
18             continue;
19         if(has_triple(wordarray,len))
20         {
21             count+=1;
22             cout<<word<<endl;
23         }
24         totalCount++;
25     }
```

Word Play



```
26     cout<<"\nNo of words = "<<count<<endl;  
27     inFile.close();  
28     return 0;  
29 }  
30 bool has_triple(const char * wordarray,int len)  
31 {  
32     for(int i=0; i<len-2; i++)  
33     {  
34         if (wordarray[i]==wordarray[i+1]  
35             && wordarray[i]==wordarray[i+2])  
36             return true;  
37         return false;  
38     }  
39 }
```

[Data Processing] Employee Report

- Create a text file containing the following data (without the headings) by accepting from the keyboard.
 - You need to use **getline(outFile, strVar)**

Name	Rate	Hours
Alex Berhanu	16.00	40
Eyerusalem Samuel	15.00	48
Kena Bekele	16.50	35
Rut Abay	18.00	50

- Using the data in the file created above to produce the following pay report for **each** employee

Name Pay Rate Hours Regular Pay Overtime Pay Gross Pay

- Compute regular pay as any hours worked up to and including 40 hours multiplied by the pay rate.
- Compute overtime pay as any hours worked above 40 hours times a pay rate of 1.5 multiplied by the regular rate.
- The gross pay is the sum of regular and overtime pay.
- At the end of the report, the program should display the totals of the regular, overtime, and gross pay columns.