National University of Sciences & Technology
School of Electrical Engineering and Computer Science
Department of Computing

CS-212 Object Oriented Programming
Assignment # 2

File Handling and File Streams in C++			
Maximum Marks: **30**		Instructor: **Muhammad Khurram Shahzad**	
Submission Date: **16th December 2021**		Type: Written/Individual/LMS/Hardcopy	
Name: **Muhammad Umer**	Reg. #: **345834**	Degree: **BEE**	Section: **12C**

Instructions:

- You need to implement the concepts using skills/concepts learned in this course.
- You can get internet help, but assignments are individual
- You can submit on LMS before due date and submit hard copy in the class
- Submission after due date get zero marks

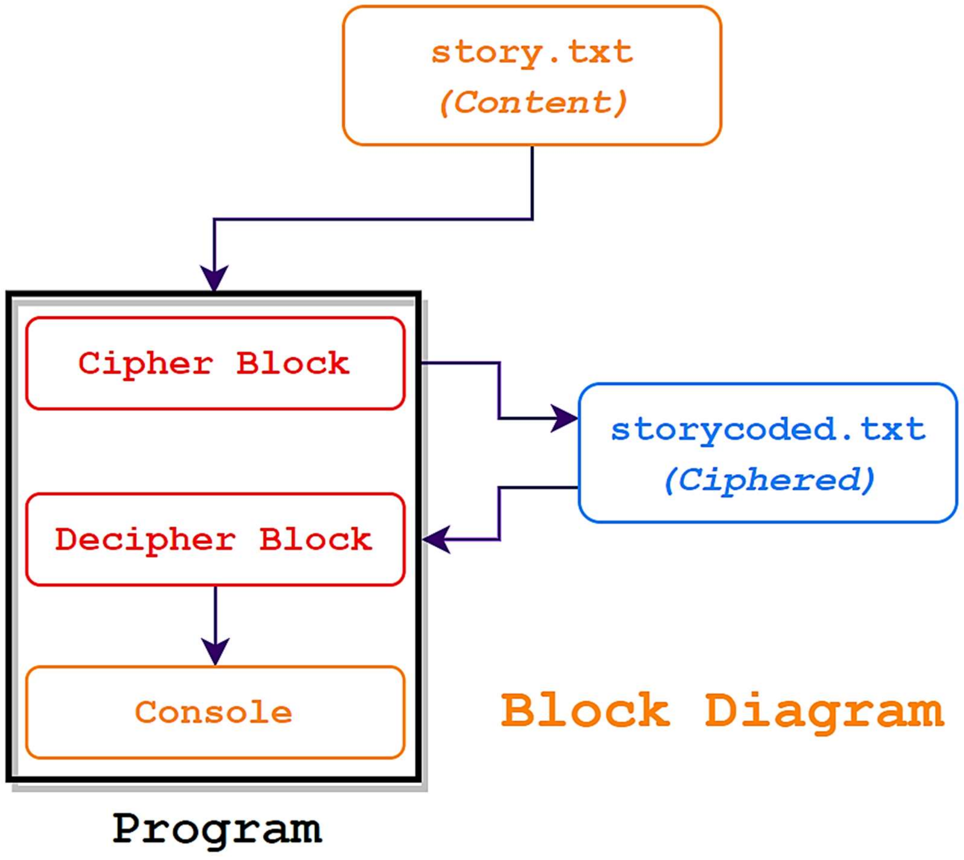
Combine the knowledge from file handing and ASCII table.

- Write your diary/story/secret in a file. *Cipher* the file using ASCII table so if anyone open that file no one could read it instead see only numbers.
- Create a program which is going to read from that file. It should convert back numbers into letters (*decipher*) and write that text to the console.
- User should input a story, program cipher story (*write sometextfile.txt*), and decipher it to the console.

Design & Structure

The design and the folder structure for the solution to this encoding / decoding application is pretty straightforward and lies within the problem statement itself. Collectively, there are four parts to the entire ensemble.

1. **Story:** The original piece of content which needs to be ciphered into ASCII code.
2. **Cipher Module:** Contains the code that reads the original story, ciphers it into ASCII and saves it into a text file.
3. **Decipher Module:** Contains code that reads the ciphered text file, deciphers it back into the original story and dispalys the result on console.
4. **Console:** Displays the output of the Decipher Module.



Now that we have been brifed over the structure of our program, we can move onto the actual implementation of it; coding it all up.

Implementation & Code

- **Story / Content**

The first and the base of our program is the content which needs to be ciphered. We can put all our content / story / secret into a file named `story.txt`. For the present case and for the demonstration of this code, the contents of the `story.txt` file are as follows:

story.txt

```
Several years ago, when I was still in high school, I was assigned the task,
homework rather, to construct a model of the globe using recycle material. I,
being a much docile student, noted it in my journal, that was then serving the
purpose of a to-do list, and returned home. If I recall correctly, the deadline
to it was a month and a half. Now, my plan was to finish it as soon as possible
and go take a long break and rest all the way through, things, however, did not
proceed way I had planned them to. I procrastinated all the way until it was only
one week left, and the end result of my project was deplorable, to say the least.
```

- **Cipher Module**

Since our **`story.txt`** file is up and ready, we can proceed on to cipher it by writing a code to convert each character of our code into its respective ASCII index. The following block of code ciphers the contents of the aforementioned file:

Cipher Module

```
// Simple function to convert a character into its ASCII index
int ascii_cipher(char a)
{
    int r;
    r = int(a);
    return r;
}

int main()
{
    /*
        Part of Program to CIPHER the original story.txt and
                    write it into storycoded.txt
    */
    fstream contentFile, cipherFile;
    string text;
    // Reading the story to cipher
    contentFile.open("story.txt", ios::in);
    cipherFile.open("storycoded.txt", ios::out);

    if (contentFile.is_open()) {

        // Reads all the lines of the story in a sequential manner
        // Serves as a good solution to linebreaks in the content
        while (getline(contentFile, text)) {

            // Saving the coded data into storycoded.txt
            if (cipherFile.is_open()) {
                // Ciphered Text
                for (int i = 0; i < text.length(); i++) {
                    cipherFile << ascii_cipher(text[i]) << " ";
                }
                // Line break corresponding to those
                // in the original text if any present
                cipherFile << endl;
            }
        }
        contentFile.close();
        cipherFile.close();
    }
}
```

This block of code both ciphers and saves the ciphered text into **storycoded.txt** file. This saved file can then be later decoded to demonstrate the working capabilities of our code.

storycoded.txt

```
83 101 118 101 114 97 108 32 121 101 97 114 115 32 97 103 111 44 32 119
104 101 110 32 73 32 119 97 115 32 115 116 105 108 108 32 105 110 32 104
105 103 104 32 115 99 104 111 111 108 44 32 73 32 119 97 115 32 97 115
115 105 103 110 101 100 32 116 104 101 32 116 97 115 107 44 32 104 111
109 101 119 111 114 107 32 114 97 116 104 101 114 44 32 116 111 32 99 111
110 115 116 114 117 99 116 32 97 32 109 111 100 101 108 32 111 102 32 116
104 101 32 103 108 111 98 101 32 117 115 105 110 103 32 114 101 99 121 99
108 101 32 109 97 116 101 114 105 97 108 46 32 73 44 32 98 101 105 110
103 32 97 32 109 117 99 104 32 100 111 99 105 108 101 32 115 116 117 100
101 110 116 44 32 110 111 116 101 100 32 105 116 32 105 110 32 109 121 32
106 111 117 114 110 97 108 44 32 116 104 97 116 32 119 97 115 32 116 104
101 110 32 115 101 114 118 105 110 103 32 116 104 101 32 112 117 114 112
111 115 101 32 111 102 32 97 32 116 111 45 100 111 32 108 105 115 116 44
32 97 110 100 32 114 101 116 117 114 110 101 100 32 104 111 109 101 46 32
73 102 32 73 32 114 101 99 97 108 108 32 99 111 114 114 101 99 116 108
121 44 32 116 104 101 32 100 101 97 100 108 105 110 101 32 116 111 32 105
116 32 119 97 115 32 97 32 109 111 110 116 104 32 97 110 100 32 97 32 104
97 108 102 46 32 78 111 119 44 32 109 121 32 112 108 97 110 32 119 97 115
32 116 111 32 102 105 110 105 115 104 32 105 116 32 97 115 32 115 111 111
110 32 97 115 32 112 111 115 115 105 98 108 101 32 97 110 100 32 103 111
32 116 97 107 101 32 97 32 108 111 110 103 32 98 114 101 97 107 32 97 110
100 32 114 101 115 116 32 97 108 108 32 116 104 101 32 119 97 121 32 116
104 114 111 117 103 104 44 32 116 104 105 110 103 115 44 32 104 111 119
101 118 101 114 44 32 100 105 100 32 110 111 116 32 112 114 111 99 101
101 100 32 119 97 121 32 73 32 104 97 100 32 112 108 97 110 110 101 100
32 116 104 101 109 32 116 111 46 32 73 32 112 114 111 99 114 97 115 116
105 110 97 116 101 100 32 97 108 108 32 116 104 101 32 119 97 121 32 117
110 116 105 108 32 105 116 32 119 97 115 32 111 110 108 121 32 111 110
101 32 119 101 101 107 32 108 101 102 116 44 32 97 110 100 32 116 104 101
32 101 110 100 32 114 101 115 117 108 116 32 111 102 32 109 121 32 112
114 111 106 101 99 116 32 119 97 115 32 100 101 112 108 111 114 97 98 108
101 44 32 116 111 32 115 97 121 32 116 104 101 32 108 101 97 115 116 46
```

- **Decipher Module**

storycoded.txt presently has the ASCII encoding of the original file contents mentioned above, which can now be restored into its original state through the decipher module.

Decipher Module

```
// Simple function to convert an integer into its ASCII equivalent
char ascii_decipher(int a)
{
    char r;
    r = char(a);
    return r;
}

int main()
{
    /*
        Part of program to DECIPHER the ciphered storycoded.txt
                and display the results on the console
    */
    int d_text;
    string dump;
    int* ptr;

    // Reading the story to cipher
    cipherFile.open("storycoded.txt", ios::in);

    if (cipherFile.is_open()) {
        // This piece of code serves purely a purpose
        // to allocate appropiate memory for the dynamic array
        getline(cipherFile, dump);
        int dma = dump.length();

        ptr = new int[dma];
        cipherFile.seekg(0); // Resets the pointer to the start of file

        // Loop to fill the dynamic array with integers from the file
        int counter = 0;
        while (cipherFile >> d_text) { // Loops through the whole file
            // looking for integers, skipping whitespace
            ptr[counter] = d_text;
            counter++;
        }

        // Decoding and displaying output on console
        for (int i = 0; i < counter; i++) {
            cout << ascii_decipher(ptr[i]);
        }

        cipherFile.close();
    }
}
```

After the execution of the following block of code, the original content displayed onto the console, and from common observation, we can infer that our **`decipher.cpp`** works as intended and that it can decode any file ciphered in ASCII code.
The console output is also pasted later in this report.

From the aforementioned modules and text files, we can combine everything to create a single program that does all of this for us.

`main.cpp`

```
// Including libraries for I/O operations and file handling
#include <iostream>
#include <fstream>
#include <string> // To utilize getline for reading the original
// content until a linebreak is encountered

using namespace std;

// Function to convert a character into its ASCII index
int ascii_cipher(char a)
{
    int r;
    r = int(a);
    return r;
}

// Function to convert an integer into its ASCII equivalent
char ascii_decipher(int a)
```

```
{
    char r;
    r = char(a);
    return r;
}

int main()
{
    /*
        Part of Program to CIPHER the original story.txt and
        write it into storycoded.txt
    */

    fstream contentFile, cipherFile;
    string text;
    // Reading the story to cipher
    contentFile.open("story.txt", ios::in);
    cipherFile.open("storycoded.txt", ios::out);

    if (contentFile.is_open()) {

        // Reads all the lines of the story in a sequential manner
        // Serves as a good solution to linebreaks in the content
        while (getline(contentFile, text)) {

            // Saving the coded data into storycoded.txt
            if (cipherFile.is_open()) {
                // Ciphered Text
                for (int i = 0; i < text.length(); i++) {
                    cipherFile << ascii_cipher(text[i]) << " ";
                }
                // Line break corresponding to those
                // in the original text if any present
                cipherFile << endl;
            }
        }
        contentFile.close();
        cipherFile.close();
    }

    /*
        Part of program to DECIPHER the ciphered storycoded.txt
        and display the results on the console
    */

    int d_text;
    string dump;
    int* ptr;

    // Reading the story to cipher
    cipherFile.open("storycoded.txt", ios::in);

    if (cipherFile.is_open()) {
        // This piece of code serves purely a purpose
        // to allocate appropiate memory for the dynamic array
        getline(cipherFile, dump);
        int dma = dump.length();

        ptr = new int[dma];
        cipherFile.seekg(0); // Resets the pointer to the start of file

        // Loop to fill the dynamic array with integers from the file
        int counter = 0;
```

```cpp
        while (cipherFile >> d_text) { // Loops through the whole file
            // looking for integers, skipping whitespace
            ptr[counter] = d_text;
            counter++;
        }

        // Decoding and displaying output on console
        for (int i = 0; i < counter; i++) {
            cout << ascii_decipher(ptr[i]);
        }

        cipherFile.close();
    }

    return 0;
}
```

Console Output

```
PS D:\NUST\Semester 3\Object Oriented Programming\Assignments> cd
"d:\NUST\Semester 3\Object Oriented Programming\Assignments\Assignment 2\"
; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }

Several years ago, when I was still in high school, I was assigned the
task, homework rather, to construct a model of the globe using recycle
material. I, being a much docile student, noted it in my journal, that was
then serving the purpose of a to-do list, and returned home. If I recall
correctly, the deadline to it was a month and a half. Now, my plan was to
finish it as soon as possible and go take a long break and rest all the way
through, things, however, did not proceed way I had planned them to. I
procrastinated all the way until it was only one week left, and the end
result of my project was deplorable, to say the least.
```