|  |  |  |
| --- | --- | --- |
| **American University of Sharjah**  **College of Engineering**  Dept of Computer Science & Engg  P. O. Box 26666  Sharjah, UAE | A picture containing logo  Description automatically generated | **Instructors:** Dr. Aliaa Moualla  **Lab Instructor:** Sameer Alawnah  **Office:** EB1-0012C  **Phone**: 971-6-515-4940  **e-mail**: salawnah@aus.edu  **Semester**: Spring 2024 |

**CMP 220L - Programming II**

**Lab #3 – File I/O**

**Note: using ChatGPT will be considered a violation of the AUS integrity code.**

**Objectives:**

* To practice file I/O.
* To practice moving the cursor in a file.

Using Visual Studio 2022, write the below programs, compile and provide screenshots of output.

Note: you are required to submit copy of the code + screenshots of program run for each exercise.

**Exercise #1**

Write a C++ program that performs the following tasks:

1. Create a text file named "numbers.txt" and write 6 integers (one integer per line) into this file. You can choose any set of integers.
2. Read the integers from "numbers.txt" and store them in a vector.
3. Calculate and print the sum of these integers.
4. Calculate and print the average of these integers.
5. Find and print the maximum and minimum values from the array.

**number.txt**



**Sample output**

Sum of integers: 210

Average of integers: 35

Maximum value: 60

Minimum value: 10

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int main() {

ifstream input("numbers.txt");

if(input.fail())

{

cout<<"Error opening file";

}

else

{

cout<<"File opened successfuly";

}

int sum, avg = 0;

int number;

vector<int> numbers;

while(input >> number)

{

numbers.push\_back(number);

}

for(int i=0; i < numbers.size(); i++)

{

sum += numbers[i];

}

avg = sum / numbers.size();

cout <<"Sum: " << sum << endl;

cout <<"Average " << avg << endl;

int max = numbers[0];

int min = numbers[0];

for(int i =0; i < numbers.size(); i++)

{

if(numbers[i] > max)

{

max = numbers[i];

}

}

for(int i=0; i < numbers.size(); i++)

{

if(numbers[i] < min)

{

min = numbers[i];

}

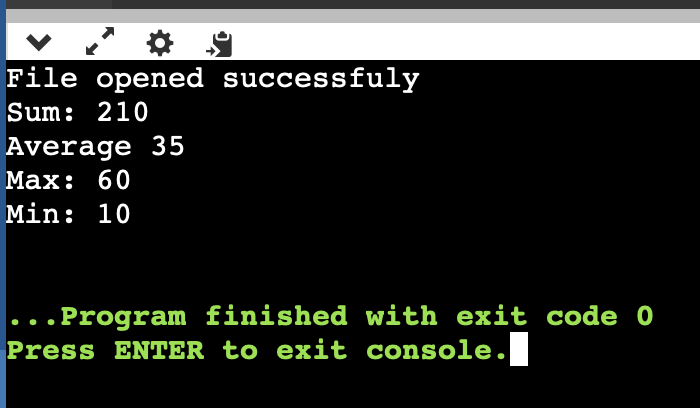
}

cout <<"Max: " << max << endl;

cout <<" Min: " << min << endl;

return 0;

}



**Exercise #2**

**File Cursor Position**

1. Create a C++ program that performs the following tasks:
2. Open a text file named "exercise.txt" in write mode (**ofstream**).
3. Ask the user to input a string of text.
4. Displaying the cursor position before and after writing
5. Write the user's input to the file.
6. Use the **tellg()** function to determine and display the current cursor (or file pointer) position in bytes before and after writing the user's input.
7. Provide a menu-driven program that lets the user choose between writing to the file, displaying the cursor position after writing, and include an option to exit the program.
8. Ensure that the program handles errors gracefully, such as incorrect user input or failed file operations.

**Sample output**

Menu:

1. Write to the file

2. Display cursor position before and after writing

3. Exit

Current cursor position: 0 bytes

Enter your choice: 1

Enter a string: CMP220L

Text written to the file

Menu:

1. Write to the file

2. Display cursor position before and after writing

3. Exit

Current cursor position: 7 bytes

Enter your choice: 2

The cursor position after writing is : 7 bytes

Menu:

1. Write to the file

2. Display cursor position before and after writing

3. Exit

Current cursor position: 7 bytes

Enter your choice: 3

**#include <iostream>**

**#include <fstream>**

**using namespace std;**

**int main() {**

**ofstream outputFile("exercise.txt", ios::out);**

**if (!outputFile.is\_open()) {**

**cerr << "Error opening the file." << endl;**

**return 1;**

**}**

**int cursorPositionBefore, cursorPositionAfter;**

**while (true) {**

**cout << "Menu:\n";**

**cout << "1. Write to the file\n";**

**cout << "2. Display cursor position before and after writing\n";**

**cout << "3. Exit\n";**

**cout << "Current cursor position: " << outputFile.tellp() << " bytes\n";**

**cout << "Enter your choice: ";**

**int choice;**

**cin >> choice;**

**switch (choice) {**

**case 1: {**

**cin.ignore();**

**cout << "Enter a string: ";**

**string userInput;**

**getline(cin, userInput);**

**cursorPositionBefore = outputFile.tellp();**

**outputFile << userInput;**

**outputFile.flush();**

**cursorPositionAfter = outputFile.tellp();**

**cout << "Text written to the file\n";**

**break;**

**}**

**case 2:**

**cout << "The cursor position after writing is: " << cursorPositionAfter << " bytes\n";**

**break;**

**case 3:**

**outputFile.close();**

**cout << "Exiting the program." << endl;**

**return 0;**

**default:**

**cerr << "Invalid choice. Please enter a valid option.\n";**

**}**

**}**

**return 0;**

**}**

**A screenshot of a computer screen

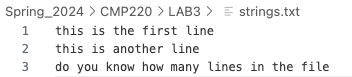
Description automatically generated**

**Exercise #3**

**Reading File Line by Line and Word by Word**

1. Create a text file named "string.txt" with several lines of text, including spaces.
2. Write a C++ program that performs the following tasks:
3. Open the "strings.txt" file in read mode (**ifstream**). st
4. Read the contents of the file line by line using the **getline** function.
5. After processing all lines, count and display the total number of lines and words in the file.
6. Close the file.
7. Ensure that your program handles errors gracefully, such as failing to open the file.

**strings.txt**

****

**Sample output**

Number of lines are 3

Number of words are 18

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int main() {

ifstream infile("string.txt");

if(!infile.is\_open())

{

cout<<"cant open file" << endl; exit(1);

}

int wordsCount, linesCount = 0;

string line;

while(getline(infile, line))

{

linesCount++;

}

cout <<"number of lines are " << linesCount <<endl;

infile.clear();

infile.seekg(0);

string word;

while(infile >> line)

{

wordsCount++;

}

cout<<"number of words are " << wordsCount << endl;

return 0;

}**A screen shot of a computer

Description automatically generated**