### SWE20004 Technical Software Development Semester 2 2019

# Assignment 3 report

## Name: Jake Scott Student ID: 102581840

## Lab class: Wednesday / 12:30 PM / BA601

**Due Date: Wednesday 30th September 2019 at 11:59pm**

**Date Submitted: Wednesday 30th September 2019 at 11:50pm**

## Assignment Title:

### Program description

* Part one of the assignment consists of a menu-based program with different functions, however part one gets its data from a file named “grades.txt” and stores it in a structured array. Below are the menu functions:

1. Display all student data in the array to the screen.
2. Calculate the floating average for all student marks within the array.
3. Sort student details, with respect to either mark (Highest to Lowest) or name (A to Z)
4. Search for a student record based off student name, user decides search algorithm, either linear or binary search.
5. Find the highest student mark from all records
6. Add a new student to the “grades.txt” file and update the student array with new student information
7. Exit the program.

* Part two of the assignment consists of many of the same elements for Part one however the way the data is collected, and store varies, for part two we used a structured linked list to store all student records.

1. Display all student data in the linked list.
2. Calculate the floating average for all student marks within the linked list.
3. Search for a particular student within the linked list
4. Find the highest student mark from all records
5. Add a new student to the linked list
6. Exit the program.

### Problem 1 - Source code:

/\*

Filename: 102581840\_assignment3Part1.cpp

Purpose: Part 1, Assignment 3

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

using namespace std;

//Structure of student array

struct student{

string name;

int id;

float mark;

};

// Reading in values from file into array

struct student readFile(student studentArray[100], string fileLocation) {

ifstream file(fileLocation.c\_str());

for (int i = 0; i < 100; i++) {

file >> studentArray[i].name >> studentArray[i].id >> studentArray[i].mark;

if (file.eof()) {

break;

}

}

file.close();

}

// Menu Function

int menu() {

int menuSelection;

cout << "" << endl;

cout << "" << endl;

cout << "1. Display student's details" << endl;

cout << "2. Calculate average of all student's marks" << endl;

cout << "3. Sort the student's details" << endl;

cout << "4. Search for a particular student's mark" << endl;

cout << "5. Find maximum grade" << endl;

cout << "6. Add new student to the record" << endl;

cout << "7. Quit Program" << endl;

cin >> menuSelection;

return menuSelection;

}

// Print students to screen

void displayStudents(student studentArray[100], int sizeOfArray) {

for (int i = 0; i < sizeOfArray; i++) {

cout << studentArray[i].name << " | " << studentArray[i].id << " | " << studentArray[i].mark << endl;

}

cout << "" << endl;

}

// Calculate Average Student Mark

int calculateAverage(student studentArray[100], int sizeOfArray) {

float gradeTotal = 0;

float average = 0;

int divisor = 0;

for (int i = 0; i < sizeOfArray; i++) {

gradeTotal = studentArray[i].mark + gradeTotal;

divisor = divisor + 1;

}

average = gradeTotal / divisor;

return average;

}

// Calculate the size of the array, based up how many rows are contained within "grades.txt"

int sizeOfArray(student studentArray[100], string fileLocation) {

int counter = 0;

string line;

ifstream file(fileLocation.c\_str());

while(getline(file, line)) {

counter++;

}

return counter;

}

// Contains two different sorts based on user input (A-Z or Highest-Lowest)

void sortStudents(student studentArray[100], int sizeOfArray) {

int menu;

cout << "" << endl;

cout << "1. Sort by name (A-Z)" << endl;

cout << "2. Sort by marks (Highest-Lowest)" << endl;

cin >> menu;

if (menu == 1) {

for (int i = 0; i < sizeOfArray; i++) {

for (int j = i + 1; j < sizeOfArray; j++) {

if (studentArray[i].name > studentArray[j].name) {

swap(studentArray[i], studentArray[j]);

}

}

}

cout << "Sorted Results by name (A-Z)" << endl;

cout << "----------------------------" << endl;

displayStudents(studentArray, sizeOfArray);

}

else if (menu == 2) {

for (int i = 0; i < sizeOfArray; i++) {

for (int j = i + 1; j < sizeOfArray; j++) {

if (studentArray[i].mark < studentArray[j].mark) {

swap(studentArray[i], studentArray[j]);

}

}

}

cout << "Sorted Results by mark (Highest-Lowest)" << endl;

cout << "---------------------------------------" << endl;

displayStudents(studentArray, sizeOfArray);

}

}

// Search for a student record based of user input (Linear or Binary)

int searchStudent(student studentArray[100], int arraySize) {

int menuOption;

int flag = 0;

int searchIndex = 0;

string studentName;

cout << "Enter Student Name to search for: " << endl;

cin >> studentName;

cout << "Search for student: " << studentName << endl;

cout << "1. Linear Search" << endl;

cout << "2. Binary Search" << endl;

cin >> menuOption;

if (menuOption == 1) {

for (int i = 0; i < arraySize; i++) {

if (studentArray[i].name == studentName) {

searchIndex = i;

flag = 1;

break;

}

}

if (flag == 0) {

searchIndex = -1;

}

return searchIndex;

}

else if (menuOption == 2) {

int first = 0;

int last = arraySize;

int middle = (first + last) / 2;

while (first <= arraySize) {

if (studentArray[middle].name < studentName) {

first = middle + 1;

}

else if (studentArray[middle].name == studentName) {

return middle + 1;

break;

}

else {

last = middle - 1;

}

middle = (first + last) / 2;

}

}

}

// Find the student with the highest mark

int findMaximum(student studentArray[100], int sizeOfArray) {

float maximumMark = 1.0;

for (int i = 0; i < sizeOfArray; i++) {

if (maximumMark < studentArray[i].mark) {

maximumMark = studentArray[i].mark;

}

}

return maximumMark;

}

// Update File Function that adds user input into "grades.txt" file

void updateFile(student studentArray[100], string fileLocation, int sizeOfArray) {

//Local vairables to write to file

string name;

int id;

float mark;

int indexPtr = sizeOfArray + 1;

// Prompt user for values

cout << "Please enter student name: " << endl;

cin >> name;

studentArray[indexPtr].name = name;

cout << "Please enter student id: " << endl;

cin >> id;

studentArray[indexPtr].id = id;

cout << "Please enter student mark: " << endl;

cin >> mark;

studentArray[indexPtr].mark = mark;

// Write changes to file

ofstream file;

file.open(fileLocation.c\_str(), ios::app);

file << "\n" << name << " " << id << " " << mark;

file.close();

}

/\*int readUpdatedFile (student studentArray[100], string fileLocation, int sizeOfArray()) {

ifstream file(fileLocation.c\_str());

sizeOfArray = sizeOfArray + 1;

file >> studentArray[sizeOfArray].name >> studentArray[sizeOfArray].id >> studentArray[sizeOfArray].mark;

file.close();

}\*/

int main()

{

// Variable Initilization

struct student studentArray[100];

string fileLocation = "grades.txt";

int menuSelcetion, gradeAverage;

int searchIndex = 0;

bool z = true;

string studentName = "default";

float maximumMark;

// Start of Program

cout << "Welcome to the Student Management Program!" << endl;

cout << "To continue, please enter the file location of student grades: " << endl;

cin >> fileLocation;

// Reading grades.txt file, function call.

readFile(studentArray, fileLocation);

// Calling the menu function

while (z) {

// Continues to keep track of array size

int arraySize = sizeOfArray(studentArray, fileLocation);

menuSelcetion = menu();

switch(menuSelcetion) {

case 1:

displayStudents(studentArray, arraySize);

break;

case 2:

gradeAverage = calculateAverage(studentArray, arraySize);

cout << "The student average is " << gradeAverage << endl;

break;

case 3:

sortStudents(studentArray, arraySize);

break;

case 4:

searchIndex = searchStudent(studentArray, arraySize);

if (searchIndex == -1) {

cout << "Couldn't find student name" << endl;

}

else {

cout << "Found student at array index: " << searchIndex << endl;

cout << studentArray[searchIndex].name << " | " << studentArray[searchIndex].id << " | " << studentArray[searchIndex].mark << endl;

}

break;

case 5:

maximumMark = findMaximum(studentArray, arraySize);

cout << "Maximum mark is: " << maximumMark << endl;

break;

case 6:

updateFile(studentArray, fileLocation, arraySize);

break;

case 7:

cout << "Thank you for using the student management program!" << endl;

cout << "Exiting Program..." << endl;

z = false;

break;

}

}

return 0;

}

### Problem 1 - Screenshots showing working program (Show all possible outcome):A screenshot of a cell phone Description automatically generatedA screenshot of a cell phone Description automatically generatedA screenshot of a cell phone Description automatically generatedA screenshot of a cell phone Description automatically generated

### 4. Problem 2 - Source code:

/\*

Filename: 102581840\_assignment3Part2.cpp

Purpose: Part 2, Assignment 3

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

#include <fstream>

using namespace std;

// Linked List Structure

struct student{

string name;

int id;

float mark;

student \*next;

};

// Assigning the pointer of head to null

struct student \*head = NULL;

// Function to insertNode into the linked list

void insertNode(string studentName, int studentID, float studentMark) {

struct student \*newNode = new student;

newNode->name = studentName;

newNode->id = studentID;

newNode->mark = studentMark;

newNode->next = head;

head = newNode;

}

// Function to read in data from file

void readFile() {

// Variable Initilization to be passed

string studentName;

int studentID;

float studentMark;

ifstream file("grades.txt");

for (int i = 0; i < 4; i++) {

file >> studentName >> studentID >> studentMark;

insertNode(studentName, studentID, studentMark);

}

file.close();

}

// Function to display data from linked list to screen

void display() {

if (head == NULL) {

cout << "Student List is empty!" << endl;

return;

}

struct student \*temp = head;

while(temp != NULL) {

cout << temp->name << ", " << temp->id << ", " << temp->mark << endl;

temp = temp->next;

}

cout << " " << endl;

}

// Function to calculate the average mark of students within the list

int calculateAverage() {

int markTotal = 0;

int divisor = 0;

int average = 0;

struct student \*temp = head;

while(temp != NULL) {

markTotal += temp->mark;

divisor++;

temp = temp->next;

}

average = markTotal / divisor;

return average;

}

// Function to search for a student from the list

void searchStudent() {

string searchString;

int flag = 0;

cout << " " << endl;

cout << "Please enter a student name to search for: " << endl;

cin >> searchString;

cout << "" << endl;

struct student \*temp = head;

while(temp != NULL) {

if (temp->name == searchString) {

flag = 1;

cout << "Student Found: " << endl;

cout << temp->name << ", " << temp->id << ", " << temp->mark << endl;

break;

}

temp = temp->next;

}

if (flag == 0) {

cout << "Error: Student not found" << endl;

}

}

// Function to find the maximum mark within the list

void findMaximum() {

float maximumMark = 1.0;

struct student \*temp = head;

while(temp != NULL) {

if(maximumMark < temp->mark) {

maximumMark = temp->mark;

}

temp = temp->next;

}

cout << "The maximum mark for all students is: " << maximumMark << endl;

}

// Function to add a new node to the list

void updateFile() {

// Variable Initilzation for Input to file.

string studentName;

int studentID;

float studentMark;

cout << "Please enter student name: " << endl;

cin >> studentName;

cout << "Please enter the student ID: " << endl;

cin >> studentID;

cout << "Please enter the student mark: " << endl;

cin >> studentMark;

// Write new student to node

insertNode(studentName, studentID, studentMark);

}

// Function to display the menu to the screen

int menu() {

int menuSelection;

cout << "1. Display student's details" << endl;

cout << "2. Calculate average of all student's marks" << endl;

cout << "3. Search for a particular student's mark" << endl;

cout << "4. Find maximum grade" << endl;

cout << "5. Add a new student to the record" << endl;

cout << "6. Exit Program" << endl;

cin >> menuSelection;

return menuSelection;

}

int main() {

// Variable Initilization

int menuSelection;

float studentAverage;

bool z = true;

// Processing Data From File in Main

readFile();

// Redirecting to Menu Option

while(z) {

menuSelection = menu();

switch(menuSelection) {

case 1:

display();

break;

case 2:

studentAverage = calculateAverage();

cout << "The average mark of all students is: " << studentAverage << endl;

break;

case 3:

searchStudent();

break;

case 4:

findMaximum();

break;

case 5:

updateFile();

break;

case 6:

cout << "Thank you for using student management program!" << endl;

z = false;

break;

}

}

return 0;

}

### 5. Problem 2 - Screenshots showing working program (Show all possible outcome):A screenshot of a cell phone Description automatically generatedA screenshot of a cell phone Description automatically generated

## Task 8.5

### Program description

* This program reads information from a data file and then saves a summary of this information to another file, the summary includes number of price readings, maximum price, minimum price and average price.

### Source Code

/\*

Filename: Task 8.5.cpp

Purpose: Reading and Writing File Data

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

int loopCount, lineNumber;

ifstream myfile;

myfile.open("price.dat");

myfile >> loopCount;

float data[10];

float maximum = 0;

float minimum = 1000;

float total = 0;

for (int i = 0; i < loopCount; i++) {

myfile >> lineNumber >> data[i];

cout << data[i] << endl;

}

myfile.close();

for (int i = 0; i < loopCount; i++) {

total = data[i] + total;

if (data[i] > maximum) {

maximum = data[i];

}

if (data[i] < minimum) {

minimum = data[i];

}

}

ofstream outputFile ("output.dat");

if (outputFile.is\_open()) {

outputFile << "Number of price readings: " << loopCount << endl;

outputFile << "Maximum Price: " << maximum << endl;

outputFile << "Minimum Price: " << minimum << endl;

outputFile << "Average Price: " << total/loopCount << endl;

}

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generatedA screenshot of a social media post

Description automatically generated

## Task 8.6

/\* Incomplete \*\

## Task 9.2

### Program description

* Using structured array’s to store both employee and company data into a structured array, then accessing and manipulating the data to display the average salary of employees to a particular company.

### Source Code

/\*

Filename: Task 9.2.cpp

Purpose: Using Structured Arrays

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

using namespace std;

struct company\_detail {

string company\_id;

string company\_name;

};

struct Emp {

string emp\_name;

string emp\_id;

double salary;

struct company\_detail cmp\_detail;

};

int get\_data(Emp employees[]) {

for (int i = 0; i < 2; i++) {

cout << "Please enter Employee Name: " << endl;

cin >> employees[i].emp\_name;

cout << "Please enter Employee ID: " << endl;

cin >> employees[i].emp\_id;

cout << "Please enter Employee Salary: " << endl;

cin >> employees[i].salary;

cout << "Please enter Employee's Company ID: " << endl;

cin >> employees[i].cmp\_detail.company\_id;

cout << "Please enter Employee's Company Name: " << endl;

cin >> employees[i].cmp\_detail.company\_name;

}

}

void print\_data(Emp employees[]) {

for (int i = 0; i < 2; i++) {

cout << "--------------------------" << endl;

cout << "Employee " << i << endl;

cout << "Name: " << employees[i].emp\_name << endl;

cout << "ID: " << employees[i].emp\_id << endl;

cout << "Salary: " << employees[i].salary << endl;

cout << " " << endl;

cout << "Company Details: " << endl;

cout << "Company ID: " << employees[i].cmp\_detail.company\_id << endl;

cout << "Company Name: " << employees[i].cmp\_detail.company\_name << endl;

cout << "--------------------------" << endl;

}

}

double get\_average(Emp employees[], string searchString) {

int employeeNumbers = 0;

int salaryTotal = 0;

int salaryAverage = 0;

for (int i = 0; i < 2; i++) {

if (employees[i].cmp\_detail.company\_name == searchString) {

salaryTotal += employees[i].salary;

employeeNumbers++;

}

}

salaryAverage = (salaryTotal/employeeNumbers);

cout << "Avarage salary of " << searchString << " is: " << salaryAverage << endl;

}

int menu() {

int menuSelection;

cout << "1. Display All Records" << endl;

cout << "2. Get average (Per Company)" << endl;

cout << "3. Exit Program" << endl;

cin >> menuSelection;

return menuSelection;

}

int main()

{

// Variable Initilization

Emp employees[5];

int menuSelection;

string searchString;

bool z = true;

// Call get\_data function

get\_data(employees);

while(z) {

menuSelection = menu();

switch (menuSelection) {

case 1:

print\_data(employees);

break;

case 2:

cout << "Please enter the company name to find average: " << endl;

cin >> searchString;

get\_average(employees, searchString);

break;

case 3:

cout << "Exiting Program..." << endl;

z = false;

break;

}

}

// Return Code

return 0;

}

### Screenshots showing working program

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

## Task 9.3

/\* Incomplete \*\

## Task 10.2

### Program description

* Using a linked list to store 5 letters in individual nodes, then displaying the data in each node to the screen.

### Source Code

/\*

Filename: Task 10.2cpp

Purpose: Linked list data storage & output

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

using namespace std;

struct studentName {

char \*letter;

struct studentName \*next;

};

typedef struct studentName STUDENTName;

typedef STUDENTName \*STUDENTNamePtr;

void display(studentName \*contents) {

while (contents != NULL) {

cout << contents->letter << "\t\t" << endl;

contents = contents->next;

}

cout << endl;

return;

}

int main()

{

studentName \*root;

studentName \*conductor;

root = new studentName;

root->next = 0;

root->letter = "s";

conductor = root;

cout << root->letter << endl;

if (conductor != 0) {

while( conductor->next !=0)

conductor = conductor->next;

}

conductor->next = new studentName;

conductor = conductor->next;

conductor->next = 0;

conductor->letter = "c";

cout << conductor->letter << endl;

conductor->next = new studentName;

conductor = conductor->next;

conductor->next = 0;

conductor->letter = "o";

cout << conductor->letter << endl;

conductor->next = new studentName;

conductor = conductor->next;

conductor->next = 0;

conductor->letter = "t";

cout << conductor->letter << endl;

conductor->next = new studentName;

conductor = conductor->next;

conductor->next = 0;

conductor->letter = "t";

cout << conductor->letter << endl;

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generated

## Task 10.3

### Program description

* Linked list that inserts 10 random integers (0-50) in order and insert it into a linked list, using a self-referential structure.

### Source Code

/\*

Filename: Task 10.3.cpp

Purpose: Number Sorting into Linked List

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 30/10/2019

\*/

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

struct node {

int number;

node \*next;

};

struct node \*head = NULL;

void insertNode(int insertNumber) {

struct node \*newNode = new node;

newNode->number= insertNumber;

newNode->next = head;

head = newNode;

}

void display() {

int i = 1;

if (head == NULL) {

cout << "List is empty." << endl;

}

struct node \*temp = head;

while(temp != NULL) {

cout << i << ": " <<temp->number << endl;

temp = temp->next;

i++;

}

cout << "" << endl;

}

void numberSort(node randomNumber[10]) {

for (int i = 0; i < 10; i++) {

for (int j = i + 1; j < 10; j++) {

if(randomNumber[i].number < randomNumber[j].number) {

swap(randomNumber[i].number, randomNumber[j].number);

}

}

}

}

void calculateAverage(node randomNumber[10]) {

int numberTotal = 0;

int divisor = 0;

int average = 0;

struct node \*temp = head;

while (temp != NULL) {

numberTotal += temp->number;

divisor++;

temp = temp->next;

}

average = (numberTotal/divisor);

cout << "The average of all random numbers is: " << average << endl;

}

int main() {

srand(time(0));

struct node randomNumber[10];

for (int i = 0; i < 10; i++) {

randomNumber[i].number = rand() % 50 + 1;

}

numberSort(randomNumber);

for (int i = 0; i < 10; i++) {

insertNode(randomNumber[i].number);

}

display();

calculateAverage(randomNumber);

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated