TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB REPORT ON

OBJECT ORIENTED PROGRAMMING [CT 451]

LAB 8

FILE INPUT/OUTPUT IN C++

Submitted by:

Rujal Acharya

PUL076BEI029

Submitted to:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

December, 2020

Problem:

Write a menu driver program in CPP to illustrate the concept of complete input/output operations on data files. Use a class named "student" with members name, roll, marks and address to represent a record. Your program must be able to do following file operations:

- ✓ Write n records to the file.
- ✓ Read current records stored on the file.
- ✓ Update a record on the file.
- ✓ Search a record on the file.
- ✓ Modify a record on the file.
- ✓ Delete a record on the file.
- ✓ Compute the no. of records and total file size.

Program:

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <vector>
#include <cstdlib>

// The name of file is started with "." so that it is made hidden and there is less chance that
// people will modify it by accident.
#define FILE_NAME ".data"

class Student {
	public:
		int getRoll ( );
		friend std::ostream& operator << (std::ostream& out, Student& student);
		friend std::istream& operator >> (std::istream& in, Student& student);

	private:
		// std::string is not used here, instead array of characters is used.
		// It is because the use of std::string made the program prone to segmentation fault
		// while reading (due to variable length of string).
		char name[50];
		char address[50];
		int roll;
		float marks;
};

int Student::getRoll ( ) {
	return this->roll;
}

std::ostream& operator << (std::ostream& out, Student& student) {
```

```
	out << std::setfill (' ') << std::setw (15) << student.name << std::setw (7) << student.roll
<< std::setw (7) << std::fixed << std::setprecision(1) << student.marks << std::setw (15) <<
student.address;
	return out;
}

std::istream& operator >> (std::istream& in, Student& student) {
	in >> student.name >> student.roll >> student.marks >> student.address;
	return in;
}


// Write records into the file
void writeRecords ( ) {
	std::ofstream *file = new std::ofstream (FILE_NAME);
	Student student;
	int n;
	std::cout << "Enter the number of records you want to add: ";
	std::cin >> n;
	for (int i = 0; i < n; i++) {
		std::cout << "Enter name, roll no, marks and address of student no " << i + 1 << ":"
<< std::endl;
		std::cin >> student;
		file->write (reinterpret_cast <char *> (&student), sizeof (student));
	}
	std::cout << "Finished writing..." << std::endl;
	file->close ( );
	delete file;
}

// Read records from the file
void readRecords ( ) {
	Student student;
	std::ifstream *file = new std::ifstream ( FILE_NAME );

	if (!file) {
		std::cout << "Error opening file " << std::endl << "Closing..." << std::endl;
		exit (EXIT_FAILURE);
	}

	std::cout << std::setfill (' ') <<    std::setw (15) << "Name" << std::setw (7) << "Roll" <<
std::setw (7) << std::fixed << std::setprecision(1) << "Marks" << std::setw (15) << "Address"
<< std::endl;
	while (file->read (reinterpret_cast <char *> (&student), sizeof (student))) {
		std::cout << student << std::endl;
	}
	std::cout << "Finished reading..." << std::endl;
	file->close ( );
	delete file;
}


// Update a record into the file
void updateRecord ( ) {
	std::ofstream *file = new std::ofstream (FILE_NAME, std::ios::app);
```

```
	Student student;
	std::cout << "Enter name, roll no, marks and address of student:" << std::endl;
	std::cin >> student;
	file->write (reinterpret_cast <char *> (&student), sizeof (student));
	std::cout << "Finished updating..." << std::endl;
	file->close ( );
	delete file;
}

// Search for a record in the file
void searchRecord ( ) {
	std::ifstream *file = new std::ifstream (FILE_NAME);
	Student student;
	if (!file) {
		std::cout << "Error opening file " << std::endl << "Closing..." << std::endl;
		exit (EXIT_FAILURE);
	}

	int roll;
	bool isFound = false;
	std::cout << "Enter the roll no of student: ";
	std::cin >> roll;
	while (file->read (reinterpret_cast <char *> (&student), sizeof (student))) {
		if (student.getRoll ( ) == roll) {
			std::cout << std::setfill (' ') <<    std::setw (15) << "Name" << std::setw (7) <<
"Roll" << std::setw (7) << std::fixed << std::setprecision(1) << "Marks" << std::setw (15) <<
"Address" << std::endl;
			std::cout << student << std::endl;
			isFound = true;
			break;
		}
	}

	if (!isFound) {
		std::cout << "Sorry, no student is detected with the given roll number" << std::endl;
	}
	std::cout << "Finished searching..." << std::endl;
	file->close ( );
	delete file;
}

// Modify a record in the file
void modifyRecord ( ) {
	std::fstream *file = new std::fstream ( );
	Student student;
	file->open (FILE_NAME, std::ios::in);
	if (!file) {
		std::cout << "Error opening file " << std::endl << "Closing..." << std::endl;
		exit (EXIT_FAILURE);
	}

	std::vector <Student> stds;
	while (file->read (reinterpret_cast <char *> (&student), sizeof (student))) {
		stds.push_back (student);
	}
```

```
	file->close( );

	int roll;
	bool isModified = false;
	std::cout << "Enter the roll no of student whose data you want to modify: " ;
	std::cin >> roll;
	file->open (FILE_NAME, std::ios::out);

	for (int i = 0; i < stds.size ( ); i++) {
		if (stds.at(i).getRoll( ) != roll) {
			file->write (reinterpret_cast <char *> (& stds.at (i)), sizeof (student));
		} else {
			isModified = true;
			std::cout << "Entry found..." << std::endl << "Enter new name, roll no, marks
and address of student: " << std::endl;
			std::cin >> student;
			file->write (reinterpret_cast <char *> (& student), sizeof (student));
			std::cout << "Successfully modified..." << std::endl;
		}
	}

	if (!isModified) {
		std::cout << "The entry could not be found..." << std::endl;
	}
	file->close ( );
	delete file;
}

// Delete a record from the file
void deleteRecord ( ) {
	std::fstream *file = new std::fstream ( );
	Student student;
	file->open (FILE_NAME, std::ios::in);
	if (!file) {
		std::cout << "Error opening file " << std::endl << "Closing..." << std::endl;
		exit (EXIT_FAILURE);
	}

	std::vector <Student> stds;
	while (file->read (reinterpret_cast <char *> (&student), sizeof (student))) {
		stds.push_back (student);
	}

	file->close( );

	int roll;
	bool isDeleted = false;
	std::cout << "Enter the roll no of student whose data you want to delete: " ;
	std::cin >> roll;
	file->open (FILE_NAME, std::ios::out);

	for (int i = 0; i < stds.size ( ); i++) {
		if (stds.at(i).getRoll( ) != roll) {
			file->write (reinterpret_cast <char *> (& stds.at (i)), sizeof (student));
```

```
            } else {
                isDeleted = true;
            }
        }

        if (!isDeleted) {
            std::cout << "The entry could not be deleted..." << std::endl;
        }
        file->close ( );
        delete file;
}

// Count the total no of records and calculate the total size of file
void countRecord ( ) {
        std::ifstream *file = new std::ifstream (FILE_NAME);
        Student student;
        file->seekg (0, std::ios::end);
        int size = file->tellg ( ) / sizeof (student);
        std::cout << "The size of file: " << file->tellg ( ) << std::endl;
        std::cout << "The no of student data: " << size << std::endl;
        file->close ( );
        delete file;
}

// The main program
int main( ) {
        std::string q;
        do {
            int choice;
            std::cout << std::endl << std::setw (40) << std::setfill ('*') << "Student Management
System" << std::setw (15) << std::setfill ('*') << " " << std::endl
                        << "1. Write records to the file" << std::endl
                        << "2. Read current student records" << std::endl
                        << "3. Update a record" << std::endl
                        << "4. Search a record" << std::endl
                        << "5. Modify a record" << std::endl
                        << "6. Delete a record" << std::endl
                        << "7. Compute file size and total number of records" << std::endl <<
std::endl
                        << "Enter your choice: " ;

            std::cin >> choice;
            switch (choice) {
                case 1:
                    writeRecords ( );
                    break;

                case 2:
                    readRecords ( );
                    break;

                case 3:
                    updateRecord ( );
                    break;
```

```
                case 4:
                        searchRecord ( );
                        break;

                case 5:
                        modifyRecord ( );
                        break;

                case 6:
                        deleteRecord ( );
                        break;

                case 7:
                        countRecord ( );
                        break;

                default:
                        std::cout << "Invalid choice..." << std::endl;
                        break;
        }

        std::cout << std::endl << "Enter 'q' if you want to quit, any other button if you want
to continue" << std::endl;
        std::cin >> q;
    } while (q != "q");

    return EXIT_SUCCESS;
}
```