**Assignment 3 – Grade Data Loader Program**

Samuel Cook

NSCC COGS Campus

PROG2100 – Programming in C++

Kalpana Samaraweera Mudalige

October 22, 2024

Table of Contents

[Section 1. Program Objective 2](#_Toc180509560)

[*Purpose of Program* 2](#_Toc180509561)

[Section 2. Program Code 2](#_Toc180509562)

[*Grade Data Loader Source Code* 2](#_Toc180509563)

[Section 3. Screenshots 6](#_Toc180509564)

[*Screenshots of Functioning Program* 6](#_Toc180509565)

[Section 4. Code Explanation 8](#_Toc180509566)

[*Breakdown of Source Code* 8](#_Toc180509567)

[References 10](#_Toc180509568)

# Section 1. Program Objective

## *Purpose of Program*

The objective of this assignment is to create a simple program to read text from a file and display it for users to read. The program should be able to handle unique scenarios, such as empty files, more data than manageable, or files not found, and display relevant error messages (utilizing enums) to the user if such events occur.

# Section 2. Program Code

## *Grade Data Loader Source Code*

Below is the source code from my gradeLoader.cpp file.

***/\****

***Author: Samuel Cook***

***Course: PROG2100***

***Date: October 21, 2024***

***File: gradeLoader.cpp***

***Purpose: Attempt to read grades from a file, throw an appropriate error message if the reading does not occur.***

***This program assumes each unique grade is on its own line in a text file. Each line is considered a unique grade,***

***regardless of the actual contents of the line.***

***\*/***

***#include <iostream>***

***#include <fstream>***

***using namespace std;***

***// Enum to hold the error codes.***

***enum Errors {***

***NO\_ERROR = 1,***

***NOT\_FOUND,***

***EMPTY,***

***OVER\_MAX***

***};***

***// Prototype for display function.***

***void displayGrades(Errors error, string gradeArray[10]);***

***// Main function.***

***int main() {***

***string grades[10];             // used to hold grades that were read from a file.***

***string gradeEntry;             // used to temporarily hold a line (grade) from a file until it is stored in the grades array.***

***int error = EMPTY;             // used to assign an error depending on the scenario. set to Empty by default until proof of text is found.***

***ifstream readGrades("grades.txt"); // attempt to read from the grades file***

***// if the file cannot be found for reading***

***if (!readGrades) {***

***error = NOT\_FOUND; // set the error to be displayed to not found.***

***}***

***// if the file is found and opened***

***else  {***

***for (int i=0; getline(readGrades, gradeEntry); i++) { // go through every line in the file***

***// if the line is empty***

***if (gradeEntry == "") {***

***i--; // decrease the counter variable by one***

***continue; // return to the top of the loop***

***// by doing this, any time an empty line is found, the program skips over it***

***// and does not include it as a grade to be displayed, even if it is at the***

***// start or in the middle of the list of grades.***

***// This also ensures that the program does not think the maximum amount of***

***// grades has been reached before this has actually occurred.***

***// If this is the case for each loop, the value of error is still set to***

***// empty, which reflects what has occurred.***

***}***

***// if the counter has reached 10 (meaning 10 items have already been added to the grade array)***

***else if (i == 10) {***

***error = OVER\_MAX; // set the error message to the one for over maximum entries.***

***break; // exit the loop immediately***

***}***

***// if the line is not empty, and there is still room in the array***

***else {***

***grades[i] = gradeEntry; // add the line to the array.***

***error = NO\_ERROR; // set the error message to no error.***

***// this will only happen if a line with anything on it is found.***

***// empty lines will not trigger this, but a space will.***

***}***

***}***

***}***

***// call the function and pass it the error and the grades array.***

***displayGrades(static\_cast<Errors>(error), grades);***

***// always close the file that was read from.***

***readGrades.close();***

***return 0;***

***}***

***// Function for displaying either the grades or an appropriate error message.***

***void displayGrades(Errors error, string gradeArray[10]) {***

***// this will run a block of code depending on what error code reached the function call.***

***switch (error) {***

***case NO\_ERROR: // if there was at least one line with at least one character***

***cout << endl << "=== GRADES ===";***

***for (int i=0; i<=9; i++) { // loop through for the fixed number of indexes***

***if (gradeArray[i] == "") { // if an index is not filled***

***continue; // do not print it to the screen, skip it***

***}***

***else {***

***cout << endl << gradeArray[i]; // print the contents of index i to the screen on each loop.***

***}***

***}***

***cout << endl;***

***break; // break from the switch.***

***case NOT\_FOUND: // the file cannot be opened/is not found***

***cout << "ERROR: File not found." << endl; // output an error message to the screen.***

***break; // break from the switch.***

***case EMPTY: // if each line in the file contains no characters***

***cout << "ERROR: The file is empty." << endl; // output an error message to the screen.***

***break; // break from the switch.***

***case OVER\_MAX:***

***cout << "ERROR: Maximum data size exceeded." << endl;***

***break; // break from the switch.***

***default: // if somehow the error is set to another value***

***cout << "ERROR: An unknown error occurred. Please try again." << endl; // output notice of unknown error to the screen***

***break; // break from the switch.***

***}***

***}***

# Section 3. Screenshots

## *Screenshots of Functioning Program*

Below are screenshots showing program functionality. The text file I used for these tests will be included with my Assignment 3 submission.

This is the program executing when there are no issues.

A computer screen with white text

Description automatically generated

It also works with fewer than the maximum number of grades allowed.

A screen shot of a computer code

Description automatically generated

This is the program executing when the grades.txt file contains no characters.

A black background with white text

Description automatically generated

This is the program executing when there are more than the maximum number of grades in the grades.txt file.

A black background with white text

Description automatically generated

This is the result of the program when the grades.txt file is not in the targeted directory.

A black background with white text

Description automatically generated

The text file I used to test this program looks like this:

A screenshot of a computer

Description automatically generated

Blank lines are ignored by this program, though any line that contains any character (including a blank space) will be considered a grade entry.

# Section 4. Code Explanation

## *Breakdown of Source Code*

This is a very simple program, consisting of mostly if statements and one switch in a function. I will break down the pieces individually.

***Errors enumeration***

This is a very basic enum that contains four options: one for no error being found, one for no file being found, one for no text being found in the file, and one for more grades being in the file than what the program is built to handle. In the main function, one of these errors is used as an argument for a display function to display the result of the attempted file reading to the user. The values are 1-4 for the respective enum variables (NO\_ERROR, NOT\_FOUND, EMPTY, and OVER\_MAX).

***Main function***

The main function is very simple. First, variables are created. I make a string array called grades to store the read grades, a string variable called gradeEntry that will be used later to temporarily hold the most recently read line from the file grades.txt, and an integer variable called error that is initialized to equal EMPTY (the Error enumeration variable). This happens so that there can never be an instance where the value of error is not initialized.

Using ifstream, the program attempts to open and read from the grades.txt file. If this file does not exist where the program is running, the value of error will be set to NOT\_FOUND. If this happens, the program will immediately call the function to display the grades, and pass it this error code as an argument (this function will be discussed in the next section). If the file can be opened and read, a loop begins. For every line in the text file, the loop will iterate. In each loop, the program checks two things. First, it checks to see if the current line is blank. If this happens, the counter variable (i) is decremented, and the program continues to the next loop on the next line in the file. Second, if the current counter variable equals 10 (indicating that the current number of non-blank line entries is greater than the maximum amount of allowed grade entries), the program will set the value of error to OVER\_MAX and break out of the loop immediately. Breaking from this loop will jump to the function call for displaying the grades, passing the error code as an argument. If neither of these conditions occur, the current line is added on the current counter variable’s index of the grades array. At the same time, the value of error is set to equal NO\_ERROR. This assignment occurs each time an entry is added to the array, if this was moved out of the loop, it would overwrite any other error codes that were assigned (in the current way my program is written). This is not the most ideal, but the program works and exits cleanly for each outcome.

After the loop is finished, the displayGrades function is called and passed the grades array as well as the value of error, static cast as a variable from the Errors enum. If no reassignment of error occurred in the loop, the function is passed the default value of error, EMPTY, as this is the only possibility for why error wouldn’t have been reassigned. After this function is called, the grades.txt file is closed and the program ends.

***displayGrades function***

This function accepts an Error enum variable and an array of no greater than 10 indexes as arguments. It is essentially just one switch statement which uses the error code as its evaluated expression.

The NOT\_FOUND, EMPTY, and OVER\_MAX codes print simple error messages depending on which type of error the program encountered. The default value is, as far as I am aware, impossible to trigger in the current state of the program, but I included it as a precaution.

The only overly unique case in the switch is the case of NO\_ERROR. In this event, the contents of the array are printed to the screen in a nicely formatted arrangement. Before printing the contents of each index, the program checks to see if the current index is empty. This is caused by the number of grades in the file being less than the maximum rather than exactly the maximum. The program prints the empty indexes of the array made in the function rather than the one from main. This issue could have been resolved by passing pointers instead of the whole array, but I wanted to experiment and solve this problem in a different way. If an index is not empty, it is printed out normally.

After one of these possible cases occurs, the function ends, and the program returns to main and ends.

# References

W3Schools. (n.d.). *C++ arrays*. W3Schools Online Web Tutorials. https://www.w3schools.com/cpp/cpp\_arrays.asp

W3Schools. (n.d.). *C++ enumeration (enum)*. W3Schools Online Web Tutorials. https://www.w3schools.com/cpp/cpp\_enum.asp