**Grade Book Program**

#include "stdafx.h"

#include <iostream>

#include <array>

#include <string>

#include <iomanip>

class GradeBook {

public:

//constant number of students who took the test

static const size\_t students{10};

//constructor initializes courseName and grades array

GradeBook(const std::string& name, const std::array<int, students>&gradesArray)

:courseName(name), grades{gradesArray}{

}

//function to set the course name

void setCourseName(const std::string& name) {

courseName = name;

}

//function to retrieve the course name

const std::string& getCourseName() const {

return courseName;

}

//display a welcome message to the GradeBook user

void displayMessage() const {

//call getCourseName to get the name of this GradeBook's course

std::cout << "Welcome to the grade book for \n" << getCourseName()

<< "!" << std::endl;

}

//perform various operations on the data (none to modify the data)

void processGrades() const {

outputGrades(); //output grades array

//call function getAverage to calculate the average grade

std::cout << std::setprecision(2) << std::fixed;

std::cout << "\nClass average is " << getAverage() << std::endl;

//call functions getMinimum and getMaximum

std::cout << "Lowest grade is " << getMinimum()

<< "\nHighest grade is " << getMaximum() << std::endl;

outputBarChart(); //display grade distribution chart

}

//find minimum grade

int getMinimum() const {

int lowestGrade{100}; //assume lowest grade is 100

//loop through grades array

for (int grade : grades) {

if (grade < lowestGrade) {

lowestGrade = grade;

}

}

return lowestGrade; //return lowest grade

}

//find maximum grade

int getMaximum() const {

int highestGrade{ 100 }; //assume lowest grade is 100

//loop through grades array

for (int grade : grades) {

if (grade < highestGrade) {

highestGrade = grade;

}

}

return highestGrade; //return lowest grade

}

// determine average grade for test

double getAverage() const {

int total{0}; //initialize total

// sum grades in array

for (int grade : grades) {

total += grade;

}

//return average of grades

return static\_cast<double>(total) / grades.size();

}

//output bar chart displaying grade distribution

void outputBarChart() const {

std::cout << "\nGrade distribution: " << std::endl;

//stores frequency of grades in each range of 10 grades

const size\_t frequencySize{11};

std::array<unsigned int, frequencySize> frequency{};

//for each grade, increment the appropriate frequency

for (int grade : grades) {

++frequency[ grade / 10 ];

}

//for each grade frequency, print bar in chart

for (size\_t count{ 0 }; count < frequencySize; ++count) {

//output bar labels ("0-9:", ... , "90-99", "100:")

if (0 == count) {

std::cout << " 0-9 : ";

}

else if (10 == count) {

std::cout << " 100 : ";

}

else {

std::cout << count \* 10 << "-" << (count \* 10) + 9 << ": ";

}

//print bar of asterisks

for (unsigned int stars{ 0 }; stars < frequency[count]; ++stars) {

std::cout << "\*";

}

std::cout << std::endl; // starts a new line of output

}

}

//output the contents of the grades array

void outputGrades() const {

std::cout << "\nThe grades are: \n";

//output each student's grades

for (size\_t student{ 0 }; student < grades.size(); ++student) {

std::cout << "Student " << std::setw(2) << student + 1 << ": "

<< std::setw(3) << grades[student] << std::endl;

}

}

private:

std::string courseName; //course name for this grade book

std::array<int, students> grades; //array of student grades

};

int main()

{

//array of student grades

const std::array<int, GradeBook::students> grades{

87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };

std::string courseName{"CS101 Introduction to C++ Programming"};

GradeBook myGradeBook(courseName, grades);

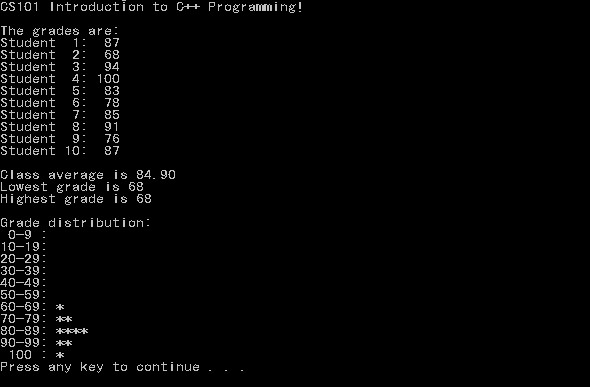
myGradeBook.displayMessage();

myGradeBook.processGrades();

system("pause");

}

**Result:**



**Important notes:**

* Use “ **:** ” after declaration to initialize variables
* Read the code in IDE to understand the algorithms if you want