  
Computer Science and Creative Technologies

**Tasks**

**Task 1: Basic student grading system prototype using classes and objects. [20 Marks]**

Write a program that manages a simple student grade calculator with the following requirements. Create a Student class that has:

1. Student name (string)
2. Three subject marks (integers)
3. A basic member function to calculate average

The program should:

1. Accept student details (name and marks) from user input
2. Calculate and display:
   1. Total marks
   2. Average marks
   3. Grade (A for ≥90%, B for ≥80%, C for ≥70%, D for ≥60%, F for <60%)
3. Display a message if any mark is below 0 or above 100

Answer: #include <iostream>

#include <string>

class Student {

private:

std::string name;

int marks[3];

public:

Student(const std::string& studentName, int mark1, int mark2, int mark3)

: name(studentName) {

marks[0] = mark1;

marks[1] = mark2;

marks[2] = mark3;

}

int calculateTotal() const {

return marks[0] + marks[1] + marks[2];

}

float calculateAverage() const {

return calculateTotal() / 3.0f; // Return average as a float

}

char determineGrade() const {

float average = calculateAverage();

if (average >= 90) return 'A';

else if (average >= 80) return 'B';

else if (average >= 70) return 'C';

else if (average >= 60) return 'D';

else return 'F';

}

void displayDetails() const {

std::cout << "Student Name: " << name << std::endl;

std::cout << "Total Marks: " << calculateTotal() << std::endl;

std::cout << "Average Marks: " << calculateAverage() << std::endl;

std::cout << "Grade: " << determineGrade() << std::endl;

}

static bool validateMarks(int mark) {

return mark >= 0 && mark <= 100;

}

};

int main() {

std::string name;

int mark1, mark2, mark3;

std::cout << "Enter student name: ";

std::getline(std::cin, name);

std::cout << "Enter marks for three subjects (0-100):\n";

std::cout << "Subject 1: ";

std::cin >> mark1;

std::cout << "Subject 2: ";

std::cin >> mark2;

std::cout << "Subject 3: ";

std::cin >> mark3;

if (!Student::validateMarks(mark1) || !Student::validateMarks(mark2) || !Student::validateMarks(mark3)) {

std::cout << "Error: Marks must be between 0 and 100." << std::endl;

return 1;

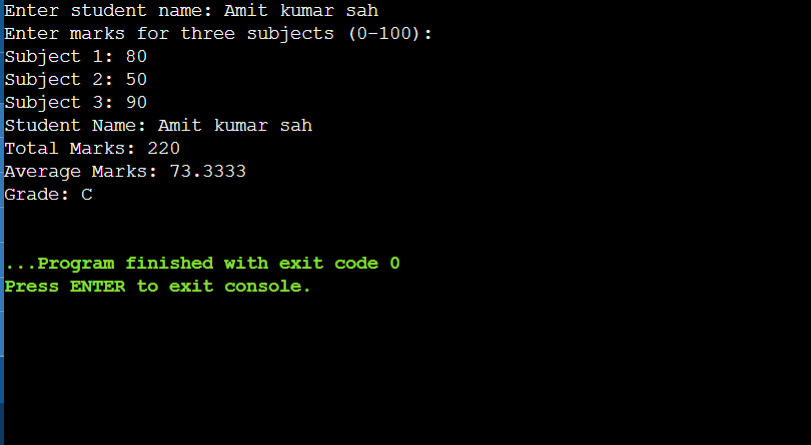
}

Student student(name, mark1, mark2, mark3);

student.displayDetails();

return 0;

}



**Task 2: Programming assignments: All questions are mandatory**

1. Write a program with a class Circle having:
   1. Private member: radius (float)
   2. A constructor to initialize radius
   3. A friend function compareTwoCircles that takes two Circle objects and prints which circle has the larger area

Answer:

**#include <iostream>**

**using namespace std;**

**class Circle {**

**private:**

**float radius;**

**public:**

**Circle(float r) {**

**radius = r;**

**}**

**friend void compareTwoCircles(Circle c1, Circle c2);**

**};**

**void compareTwoCircles(Circle c1, Circle c2) {**

**const float PI = 3.14159;**

**float area1 = PI \* c1.radius \* c1.radius;**

**float area2 = PI \* c2.radius \* c2.radius;**

**if (area1 > area2) {**

**cout << "First circle has larger area (" << area1**

**<< ") compared to second circle (" << area2 << ")" << endl;**

**}**

**else if (area2 > area1) {**

**cout << "Second circle has larger area (" << area2**

**<< ") compared to first circle (" << area1 << ")" << endl;**

**}**

**else {**

**cout << "Both circles have equal area (" << area1 << ")" << endl;**

**}**

**}**

**int main() {**

**Circle circle1(5.0);**

**Circle circle2(3.0);**

**compareTwoCircles(circle1, circle2);**

**Circle circle3(4.0);**

**Circle circle4(4.0);**

**compareTwoCircles(circle3, circle4);**

**return 0;**

**}**

1. Create a program with these overloaded functions named findMax:

One that finds maximum between two integers

One that finds maximum between two floating-point numbers

One that finds maximum among three integers

One that finds maximum between an integer and a float **[30 marks]**

Answer:

#include <iostream>

int findMax(int a, int b) {

return (a > b) ? a : b;

}

float findMax(float a, float b) {

return (a > b) ? a : b;

}

int findMax(int a, int b, int c) {

return findMax(findMax(a, b), c);

}

float findMax(int a, float b) {

return (a > b) ? a : b;

}

int main() {

int int1, int2, int3;

float float1, float2;

std::cout << "Enter two integers: ";

std::cin >> int1 >> int2;

std::cout << "Maximum between " << int1 << " and " << int2 << " is: " << findMax(int1, int2) << std::endl;

std::cout << "Enter two floating-point numbers: ";

std::cin >> float1 >> float2;

std::cout << "Maximum between " << float1 << " and " << float2 << " is: " << findMax(float1, float2) << std::endl;

std::cout << "Enter three integers: ";

std::cin >> int1 >> int2 >> int3;

std::cout << "Maximum among " << int1 << ", " << int2 << ", and " << int3 << " is: " << findMax(int1, int2, int3) << std::endl;

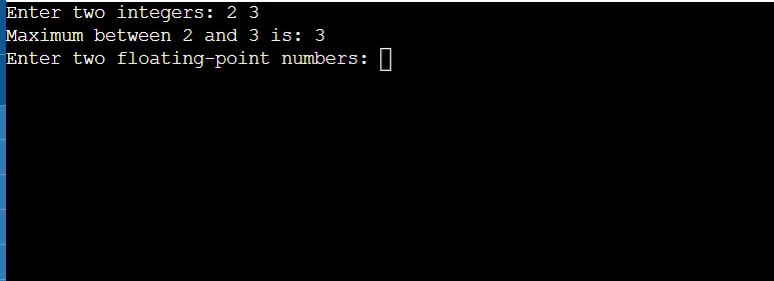
std::cout << "Enter an integer and a floating-point number: ";

std::cin >> int1 >> float1;

std::cout << "Maximum between " << int1 << " and " << float1 << " is: " << findMax(int1, float1) << std::endl;

return 0;

}



1. Write a program that reads the titles of 10 books (use an array of 150 characters) and writes them in a binary file selected by the user. The program should read a title and display a message to indicate if it is contained in the file or not.

Create a program that:

Reads student records (roll, name, marks) from a text file

Throws an exception if marks are not between 0 and 100

Allows adding new records with proper validation

Saves modified records back to file

Answer:

#include <iostream>

#include <fstream>

#include <cstring>

#include <vector>

#include <stdexcept>

using namespace std;

const int MAX\_BOOKS = 10;

const int TITLE\_SIZE = 150;

void writeBooksToBinaryFile(const string& filename) {

char titles[MAX\_BOOKS][TITLE\_SIZE];

cout << "Enter 10 book titles:\n";

for (int i = 0; i < MAX\_BOOKS; ++i) {

cout << "Book " << i + 1 << ": ";

cin.getline(titles[i], TITLE\_SIZE);

}

ofstream fout(filename, ios::binary);

if (!fout) {

cerr << "Error opening file for writing.\n";

return;

}

fout.write(reinterpret\_cast<char\*>(titles), sizeof(titles));

fout.close();

cout << "Books written to " << filename << endl;

}

void searchBookInBinaryFile(const string& filename) {

char titles[MAX\_BOOKS][TITLE\_SIZE];

ifstream fin(filename, ios::binary);

if (!fin) {

cerr << "Error opening file for reading.\n";

return;

}

fin.read(reinterpret\_cast<char\*>(titles), sizeof(titles));

fin.close();

char searchTitle[TITLE\_SIZE];

cout << "\nEnter a book title to search: ";

cin.getline(searchTitle, TITLE\_SIZE);

bool found = false;

for (int i = 0; i < MAX\_BOOKS; ++i) {

if (strcmp(titles[i], searchTitle) == 0) {

found = true;

break;

}

}

if (found)

cout << "The book is in the file.\n";

else

cout << "The book is NOT in the file.\n";

}

struct Student {

int roll;

string name;

int marks;

};

void readStudentsFromFile(const string& filename, vector<Student>& students) {

ifstream fin(filename);

if (!fin) {

cerr << "Error opening student file.\n";

return;

}

Student s;

while (fin >> s.roll >> ws && getline(fin, s.name, ',') && fin >> s.marks) {

if (s.marks < 0 || s.marks > 100)

throw out\_of\_range("Invalid marks found in file!");

students.push\_back(s);

}

fin.close();

}

void addStudent(vector<Student>& students) {

Student s;

cout << "\nEnter new student details:\n";

cout << "Roll: "; cin >> s.roll;

cin.ignore();

cout << "Name: "; getline(cin, s.name);

cout << "Marks (0-100): "; cin >> s.marks;

if (s.marks < 0 || s.marks > 100)

throw invalid\_argument("Marks must be between 0 and 100!");

students.push\_back(s);

}

void saveStudentsToFile(const string& filename, const vector<Student>& students) {

ofstream fout(filename);

if (!fout) {

cerr << "Error opening file to save students.\n";

return;

}

for (const auto& s : students) {

fout << s.roll << " " << s.name << "," << s.marks << endl;

}

fout.close();

cout << "Student records saved to " << filename << endl;

}

int main() {

string bookFile, studentFile;

cout << "Enter name of binary file for books: ";

getline(cin, bookFile);

writeBooksToBinaryFile(bookFile);

searchBookInBinaryFile(bookFile);

cout << "\nEnter name of text file for student records: ";

getline(cin, studentFile);

vector<Student> students;

try {

readStudentsFromFile(studentFile, students);

} catch (const exception& e) {

cerr << "Error: " << e.what() << endl;

}

char choice;

cout << "\nDo you want to add a new student record? (y/n): ";

cin >> choice;

if (tolower(choice) == 'y') {

try {

addStudent(students);

} catch (const exception& e) {

cerr << "Error: " << e.what() << endl;

}

}

saveStudentsToFile(studentFile, students);

return 0;

}

