 Computer Science and Creative Technologies

**Coursework or Assessment Specification**

## Module Details

|  |  |
| --- | --- |
| **Module Code** | UFCFGL-30-1 |
| **Module Title** | Programming in C++ |
| **Module Leader** | Benedict Gaster |
| **Module Tutors** | Aashish Acharya |
| **Year** | 2024-2025 |
| **Component/Element Number** | Coursework |
| **Weighting** | 10% |
| **Element Description** |  |

## Dates

|  |  |
| --- | --- |
| **Date issued to students** | 10/02/2024 |
| **Submission Date** | 10/04/2024 |
| **Submission Place** | Backboard |
| **Submission Time** | 00:00 |
| **Submission Notes** | Submit Gitlab URL |

## Feedback

|  |  |
| --- | --- |
| **Feedback provision will be** | Via Blackboard |

**Uwe id = 23045166**

**Coursework-2: OOP in C++**

This coursework covers the topics we have studied in the week5 to week9. So you are required to go through the learning materials of these weeks to complete this coursework.

You are required to develop a programmatic solution, using the object oriented concepts of C++ programming language, to meet the specified requirements as detailed below.

**Please note** - Any awarded mark is for the demonstration and explanation of the work, rather than for the submitted work itself.

The marking scheme for this session is as follows:

* Task 1: 30 marks
* Task 2: 50 marks
* Task 3: 20 marks
* Task 4: 0 marks

**Submission Details in the VLE:**

Submit your solution of worksheet-2[named as “yourid\_task\_2.docx”] along with a text file containing URL of your GIT REPO zipped in file named “YourID.zip” in the link provided on the VLE of UWE.

Necessary steps to complete this worksheet.

* Make sure you have gone through all the contents of lecture slides, tutorial files, Lab exercises and the handout notes of week-5 to week-9 folders.

**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>

using namespace std;

class Student {

private:

string name;

int marks[3];

public:

// Function to accept student details

void inputDetails() {

cout << "Enter student name: ";

getline(cin, name);

for (int i = 0; i < 3; ++i) {

cout << "Enter marks for subject " << i + 1 << " (0-100): ";

cin >> marks[i];

// Input validation

if (marks[i] < 0 || marks[i] > 100) {

cout << "Error: Marks should be between 0 and 100.\n";

--i; // Repeat the input for the same subject

}

}

}

// Function to calculate total marks

int calculateTotal() {

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

}

// Function to calculate average

float calculateAverage() {

return calculateTotal() / 3.0;

}

// Function to determine grade

char calculateGrade() {

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';

}

// Function to display results

void displayResults() {

cout << "\n--- Student Report ---\n";

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

cout << "Total Marks: " << calculateTotal() << "/300" << endl;

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

cout << "Grade: " << calculateGrade() << endl;

}

};

int main() {

Student s;

s.inputDetails();

s.displayResults();

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>

#include <cmath>

using namespace std;

class Circle {

private:

float radius;

public:

// Constructor to initialize radius

Circle(float r) {

radius = r;

}

// Friend function to compare two Circle objects

friend void compareTwoCircles(Circle c1, Circle c2);

};

// Friend function definition

void compareTwoCircles(Circle c1, Circle c2) {

float area1 = M\_PI \* c1.radius \* c1.radius;

float area2 = M\_PI \* c2.radius \* c2.radius;

cout << "\nComparing Circle Areas:\n";

cout << "Circle 1 Area: " << area1 << endl;

cout << "Circle 2 Area: " << area2 << endl;

if (area1 > area2)

cout << "Circle 1 has a larger area.\n";

else if (area2 > area1)

cout << "Circle 2 has a larger area.\n";

else

cout << "Both circles have equal area.\n";

}

1. Create a program with these overloaded functions named findMax:
   1. One that finds maximum between two integers
   2. One that finds maximum between two floating-point numbers
   3. One that finds maximum among three integers
   4. One that finds maximum between an integer and a float **[30 marks]**

Answer:

#include <iostream>

using namespace std;

// Find maximum between two integers

int findMax(int a, int b) {

return (a > b) ? a : b;

}

// Find maximum between two floats

float findMax(float a, float b) {

return (a > b) ? a : b;

}

// Find maximum among three integers

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

return (a > b && a > c) ? a : (b > c ? b : c);

}

// Find maximum between an int and a float

float findMax(int a, float b) {

return (a > b) ? a : b;

}

**Task 3: Basics of File Handling**

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:

1. Reads student records (roll, name, marks) from a text file
2. Throws an exception if marks are not between 0 and 100
3. Allows adding new records with proper validation
4. Saves modified records back to file

**[20 marks]**

Answer:

#include <iostream>

#include <fstream>

#include <vector>

#include <stdexcept>

#include <cstring>

using namespace std;

const int TITLE\_COUNT = 10;

const int TITLE\_SIZE = 150;

struct Student {

int roll;

string name;

int marks;

};

// --- BOOK FUNCTIONS ---

void manageBooks() {

char titles[TITLE\_COUNT][TITLE\_SIZE];

char fileName[100];

cout << "\n--- Enter 10 Book Titles ---\n";

cin.ignore();

for (int i = 0; i < TITLE\_COUNT; ++i) {

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

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

}

cout << "Enter binary file name to save titles: ";

cin.getline(fileName, 100);

ofstream outFile(fileName, ios::binary);

if (!outFile) {

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

return;

}

for (int i = 0; i < TITLE\_COUNT; ++i) {

outFile.write(titles[i], TITLE\_SIZE);

}

outFile.close();

cout << "Titles saved successfully.\n";

char searchTitle[TITLE\_SIZE];

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

cin.getline(searchTitle, TITLE\_SIZE);

ifstream inFile(fileName, ios::binary);

if (!inFile) {

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

return;

}

char temp[TITLE\_SIZE];

bool found = false;

while (inFile.read(temp, TITLE\_SIZE)) {

if (strncmp(temp, searchTitle, TITLE\_SIZE) == 0) {

found = true;

break;

}

}

inFile.close();

if (found) {

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

} else {

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

}

}

// --- STUDENT FUNCTIONS ---

vector<Student> readStudents(const string& filename) {

vector<Student> students;

ifstream file(filename);

if (!file) {

cerr << "File not found. A new one will be created.\n";

return students;

}

Student s;

while (file >> s.roll >> s.name >> s.marks) {

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

throw out\_of\_range("Marks must be between 0 and 100.");

}

students.push\_back(s);

}

file.close();

return students;

}

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

ofstream file(filename);

for (const auto& s : students) {

file << s.roll << " " << s.name << " " << s.marks << "\n";

}

file.close();

}

void manageStudents() {

string filename = "students.txt";

vector<Student> students;

try {

students = readStudents(filename);

} catch (const out\_of\_range& e) {

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

return;

}

cout << "\n--- Current Student Records ---\n";

for (const auto& s : students) {

cout << s.roll << " " << s.name << " " << s.marks << endl;

}

char choice;

cout << "\nAdd a new student? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

Student newStudent;

cout << "Enter roll number: ";

cin >> newStudent.roll;

cout << "Enter name: ";

cin >> newStudent.name;

cout << "Enter marks: ";

cin >> newStudent.marks;

try {

if (newStudent.marks < 0 || newStudent.marks > 100) {

throw out\_of\_range("Marks must be between 0 and 100.");

}

students.push\_back(newStudent);

} catch (const out\_of\_range& e) {

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

return;

}

}

writeStudents(filename, students);

cout << "Records saved successfully.\n";

}

// --- MAIN MENU ---

int main() {

int option;

do {

cout << "\n===== MENU =====\n";

cout << "1. Manage Book Titles\n";

cout << "2. Manage Student Records\n";

cout << "3. Exit\n";

cout << "Choose an option: ";

cin >> option;

switch (option) {

case 1:

manageBooks();

break;

case 2:

manageStudents();

break;

case 3:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid option. Try again.\n";

}

} while (option != 3);

return 0;

}

**Task 4**

* Check and commit all your solutions.
* This task carries no marks but it is mandatory. Ensure that your solution is visible to us.