**LAB 6 CONSTRUCTOR OVERLOADING**

#include <iostream>

#include <string>

#include <stdexcept>

using namespace std;

class Student {

private:

string name;

int rollNo;

float\* marks;

int numSubjects;

float average;

string grade;

public:

Student(string name = "Unknown", int rollNo = 0, int numSubjects = 3) {

this->name = name;

this->rollNo = rollNo;

this->numSubjects = numSubjects;

marks = new float[numSubjects];

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

marks[i] = 0.0;

}

average = 0.0;

grade = "F";

cout << "\nDefault constructor called for student: " << name << endl;

}

Student(string name, int rollNo) {

this->name = name;

this->rollNo = rollNo;

this->numSubjects = 3;

marks = new float[numSubjects];

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

marks[i] = 0.0;

}

average = 0.0;

grade = "F";

cout << "\nParameterized constructor called for student: " << name << " (rollNo: " << rollNo << ")" << endl;

}

Student(string name, int rollNo, int numSubjects, float\* marksArr) {

this->name = name;

this->rollNo = rollNo;

this->numSubjects = numSubjects;

marks = new float[numSubjects];

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

marks[i] = marksArr[i];

}

calculateAverage();

calculateGrade();

cout << "\nFull parameterized constructor called for student: " << name << endl;

}

Student(const Student& student) {

name = student.name;

rollNo = student.rollNo;

numSubjects = student.numSubjects;

marks = new float[numSubjects];

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

marks[i] = student.marks[i];

}

average = student.average;

grade = student.grade;

cout << "\nCopy constructor called for student: " << name << endl;

}

~Student() {

delete[] marks;

}

void getDetails() {

cout << "Enter student name: ";

cin.ignore();

getline(cin, name);

cout << "Enter roll number: ";

cin >> rollNo;

cout << "Enter marks for " << numSubjects << " subjects:\n";

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

while (true) {

try {

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

cin >> marks[i];

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

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

break;

}

catch (out\_of\_range& e) {

cout << e.what() << " Try again.\n";

}

}

}

calculateAverage();

calculateGrade();

}

void calculateAverage() {

float total = 0;

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

total += marks[i];

}

average = total / numSubjects;

}

void calculateGrade() {

if (average >= 90) {

grade = "A+";

}

else if (average >= 80) {

grade = "A";

}

else if (average >= 75) {

grade = "B+";

}

else if (average >= 70) {

grade = "B";

}

else if (average >= 60) {

grade = "C";

}

else {

grade = "F";

}

}

bool operator==(const Student& other) const {

return (name == other.name && rollNo == other.rollNo && average == other.average);

}

void displayDetails() const {

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

cout << "Roll Number: " << rollNo << endl;

cout << "Marks in " << numSubjects << " subjects:";

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

cout << " " << marks[i];

}

cout << endl;

cout << "Average Marks: " << average << endl;

cout << "Grade: " << grade << endl;

}

static void compareStudents(const Student& s1, const Student& s2) {

if (s1 == s2) {

cout << "\nBoth students have the same grades and average marks.\n";

}

else {

cout << "\nThe students have different grades or average marks.\n";

}

}

};

int main() {

Student defaultStu;

defaultStu.displayDetails();

Student paramStu(string("Bob"), 102);

paramStu.displayDetails();

float marks[] = { 85.5, 92.0, 78.5 };

Student fullParamStu("Alice", 101, 3, marks);

fullParamStu.displayDetails();

Student enteredStu;

enteredStu.getDetails();

enteredStu.displayDetails();

Student copiedStu(enteredStu);

cout << "\nDetails of copied student:\n";

copiedStu.displayDetails();

Student::compareStudents(enteredStu, copiedStu);

return 0;

}