Bse243163

Ayesha jami

**Practice task 1:**

#include <iostream>

using namespace std;

int gcd(int a, int b)

while (a != b) {

if (a > b) {

a = a - b;

} else {

b = b - a;

}

}

return a;

}

int main() {

int num1 = 56;

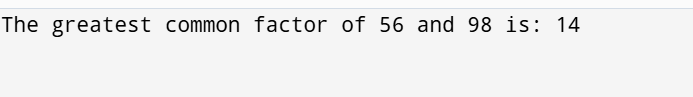
int num2 = 98;

int result = gcd(num1, num2);

cout << "The greatest common factor of " << num1 << " and " << num2 << " is: " << result << endl;

return 0;

}



**Practice task 2:**

#include <iostream>

using namespace std;

int power(int base, int exponent) {

int result = 1;

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

result \*= base;

}

return result;

}

void calculate\_product(int base, int exponent1, int exponent2, int &product) {

if (base <= 0) {

cout << "Base must be a positive number." << endl;

return;

}

int power1 = power(base, exponent1);

int power2 = power(base, exponent2);

product = power1 \* power2;

}

int main() {

int base, exponent1, exponent2, product = 0;

cout << "Enter the base: ";

cin >> base;

cout << "Enter the first exponent: ";

cin >> exponent1;

cout << "Enter the second exponent: ";

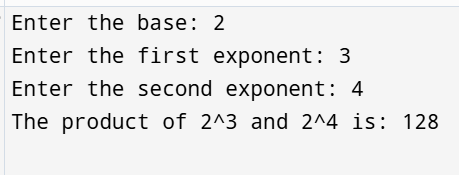
cin >> exponent2;

calculate\_product(base, exponent1, exponent2, product);

cout << "The product of " << base << "^" << exponent1 << " and " << base << "^" << exponent2 << " is: " << product << endl;

return 0;

}



**Practice task 3:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct Course {

int course\_ID;

string course\_title;

int credit\_hrs;

};

void addACourse(vector<Course>& courses) {

Course newCourse;

cout << "Enter Course ID: ";

cin >> newCourse.course\_ID;

cout << "Enter Course Title: ";

cin.ignore();

getline(cin, newCourse.course\_title);

cout << "Enter Credit Hours: ";

cin >> newCourse.credit\_hrs;

courses.push\_back(newCourse);

}

void updateACourse(vector<Course>& courses) {

int id;

cout << "Enter Course ID to update: ";

cin >> id;

for (auto& course : courses) {

if (course.course\_ID == id) {

cout << "Enter new Course Title: ";

cin.ignore();

getline(cin, course.course\_title);

cout << "Enter new Credit Hours: ";

cin >> course.credit\_hrs;

return;

}

}

cout << "Course not found." << endl;

}

void deleteACourse(vector<Course>& courses) {

int id;

cout << "Enter Course ID to delete: ";

cin >> id;

for (auto it = courses.begin(); it != courses.end(); ++it) {

if (it->course\_ID == id) {

courses.erase(it);

return;

}

}

cout << "Course not found." << endl;

}

void searchAndDisplayACourse(const vector<Course>& courses) {

int id;

cout << "Enter Course ID to search: ";

cin >> id;

for (const auto& course : courses) {

if (course.course\_ID == id) {

cout << "Course ID: " << course.course\_ID << endl;

cout << "Course Title: " << course.course\_title << endl;

cout << "Credit Hours: " << course.credit\_hrs << endl;

return;

}

}

cout << "Course not found." << endl;

}

void displayAllCourses(const vector<Course>& courses) {

if (courses.empty()) {

cout << "No courses available." << endl;

} else {

for (const auto& course : courses) {

cout << "Course ID: " << course.course\_ID << ", ";

cout << "Course Title: " << course.course\_title << ", ";

cout << "Credit Hours: " << course.credit\_hrs << endl;

}

}

}

int main() {

vector<Course> courses;

int choice;

do {

cout << "\nMenu:\n";

cout << "1. Add Course\n";

cout << "2. Update Course\n";

cout << "3. Delete Course\n";

cout << "4. Search Course\n";

cout << "5. Display All Courses\n";

cout << "6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: addACourse(courses); break;

case 2: updateACourse(courses); break;

case 3: deleteACourse(courses); break;

case 4: searchAndDisplayACourse(courses); break;

case 5: displayAllCourses(courses); break;

case 6: cout << "Exiting..." << endl; break;

default: cout << "Invalid choice. Please try again." << endl; break;

}

} while (choice != 6);

return 0;

}

