#include <iostream>

#include <vector>

#include <string>

// Structure to represent a task

struct Task {

std::string description;

bool isCompleted;

Task(std::string desc) : description(desc), isCompleted(false) {}

};

// Function declarations

void addTask(std::vector<Task>& tasks);

void viewTasks(const std::vector<Task>& tasks);

void markTaskAsCompleted(std::vector<Task>& tasks);

void removeTask(std::vector<Task>& tasks);

int main() {

std::vector<Task> tasks;

int choice;

do {

// Display menu options

std::cout << "\nTo-Do List Manager\n";

std::cout << "1. Add Task\n";

std::cout << "2. View Tasks\n";

std::cout << "3. Mark Task as Completed\n";

std::cout << "4. Remove Task\n";

std::cout << "5. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

// Menu-driven selection

switch (choice) {

case 1:

addTask(tasks);

break;

case 2:

viewTasks(tasks);

break;

case 3:

markTaskAsCompleted(tasks);

break;

case 4:

removeTask(tasks);

break;

case 5:

std::cout << "Exiting program...\n";

break;

default:

std::cout << "Invalid choice. Please try again.\n";

}

} while (choice != 5);

return 0;

}

// Function to add a new task

void addTask(std::vector<Task>& tasks) {

std::cin.ignore(); // Clear the input buffer

std::string taskDescription;

std::cout << "Enter the task description: ";

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

tasks.emplace\_back(taskDescription);

std::cout << "Task added successfully!\n";

}

// Function to view all tasks

void viewTasks(const std::vector<Task>& tasks) {

if (tasks.empty()) {

std::cout << "No tasks in the list.\n";

return;

}

std::cout << "\nTo-Do List:\n";

for (size\_t i = 0; i < tasks.size(); ++i) {

std::cout << i + 1 << ". " << tasks[i].description;

if (tasks[i].isCompleted) {

std::cout << " [Completed]";

}

std::cout << "\n";

}

}

// Function to mark a task as completed

void markTaskAsCompleted(std::vector<Task>& tasks) {

if (tasks.empty()) {

std::cout << "No tasks to mark as completed.\n";

return;

}

int taskNumber;

std::cout << "Enter the task number to mark as completed: ";

std::cin >> taskNumber;

if (taskNumber > 0 && taskNumber <= static\_cast<int>(tasks.size())) {

tasks[taskNumber - 1].isCompleted = true;

std::cout << "Task marked as completed!\n";

} else {

std::cout << "Invalid task number.\n";

}

}

// Function to remove a task

void removeTask(std::vector<Task>& tasks) {

if (tasks.empty()) {

std::cout << "No tasks to remove.\n";

return;

}

int taskNumber;

std::cout << "Enter the task number to remove: ";

std::cin >> taskNumber;

if (taskNumber > 0 && taskNumber <= static\_cast<int>(tasks.size())) {

tasks.erase(tasks.begin() + taskNumber - 1);

std::cout << "Task removed successfully!\n";

} else {

std::cout << "Invalid task number.\n";

}

}