fundamental of computer programming

LAB\_10

HANAN MAJEED

CMS:519166

**Task A:**

Write a modular program that asks the user to enter how many numbers he/she will add and then get all the numbers. All the numbers should be stored in an array (Note: a fixed-sized array should be used). After getting the numbers, a menu should be shown to the user from where the user can request the following information to be produced based on a selection:

* 1. Enter the list of numbers again
  2. Display maximum and minimum numbers in the list (don’t use built-in function)
  3. Display the sum of the squares of each number in the list
  4. Display the list in ascending and descending orders without changing the original data. You should provide a single sorting function that should sort in ascending or descending order based on the input (don’t use the built-in sort() function)
  5. Exit the program

**Input:**

#include <iostream>

using namespace std;

const int MAX\_NUMBERS = 100; // Maximum array size

// Function Prototypes

void inputNumbers(int numbers[], int& count);

void displayArray(const int numbers[], int count);

int findExtreme(const int numbers[], int count, bool findMax);

int calculateSumOfSquares(const int numbers[], int count);

void sortArray(const int numbers[], int count, int sorted[], int order);

void swap(int& a, int& b);

void handleMenuSelection(int choice, int numbers[], int count);

void displayMenu();

int main() {

int numbers[MAX\_NUMBERS];

int numberCount = 0;

// Get the number of elements to input

cout << "Enter the number of elements you want to add (max " << MAX\_NUMBERS << "): ";

cin >> numberCount;

if (numberCount <= 0 || numberCount > MAX\_NUMBERS) {

cout << "Invalid number of elements. Exiting program." << endl;

return 1;

}

inputNumbers(numbers, numberCount);

while (true) {

displayMenu();

int choice;

cin >> choice;

if (choice == 5) { // Exit the program

cout << "Exiting the program... " << endl;

break;

}

handleMenuSelection(choice, numbers, numberCount);

}

return 0;

}

// Input numbers into the array

void inputNumbers(int numbers[], int& count) {

cout << "Enter " << count << " numbers:" << endl;

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

cin >> numbers[i];

}

}

// Display the elements of an array

void displayArray(const int numbers[], int count) {

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

cout << numbers[i] << " ";

}

cout << endl;

}

// Find the maximum or minimum value in the array

int findExtreme(const int numbers[], int count, bool findMax) {

int extreme = numbers[0];

for (int i = 1; i < count; i++) {

if ((findMax && numbers[i] > extreme) || (!findMax && numbers[i] < extreme)) {

extreme = numbers[i];

}

}

return extreme;

}

// Calculate the sum of squares of all elements

int calculateSumOfSquares(const int numbers[], int count) {

int sum = 0;

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

sum += numbers[i] \* numbers[i];

}

return sum;

}

// Sort an array in ascending or descending order

void sortArray(const int numbers[], int count, int sorted[], int order) {

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

sorted[i] = numbers[i]; // Copy original array

}

// Bubble sorting

for (int i = 0; i < count - 1; i++) {

for (int j = 0; j < count - i - 1; j++) {

if ((order == 1 && sorted[j] > sorted[j + 1]) ||

(order == 2 && sorted[j] < sorted[j + 1])) {

swap(sorted[j], sorted[j + 1]);

}

}

}

}

// Swap two elements

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

// Handle the user's menu selection

void handleMenuSelection(int choice, int numbers[], int count) {

switch (choice) {

case 1:

inputNumbers(numbers, count);

break;

case 2: {

int maxValue = findExtreme(numbers, count, true);

int minValue = findExtreme(numbers, count, false);

cout << "Maximum: " << maxValue << ", Minimum: " << minValue << endl;

break;

}

case 3: {

int sumOfSquares = calculateSumOfSquares(numbers, count);

cout << "Sum of squares: " << sumOfSquares << endl;

break;

}

case 4: {

int sorted[MAX\_NUMBERS];

sortArray(numbers, count, sorted, 1); // Ascending

cout << "Ascending Order: ";

displayArray(sorted, count);

sortArray(numbers, count, sorted, 2); // Descending

cout << "Descending Order: ";

displayArray(sorted, count);

break;

}

default:

cout << "Invalid choice. Please try again." << endl;

}

}

// Display menu options to the user

void displayMenu() {

cout << "\nMenu Options:" << endl;

cout << "1. Enter the list of numbers again" << endl;

cout << "2. Display maximum and minimum numbers" << endl;

cout << "3. Display the sum of squares of the numbers" << endl;

cout << "4. Display the list in ascending and descending orders" << endl;

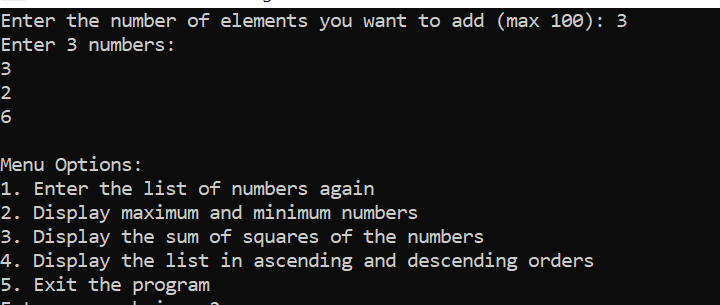
cout << "5. Exit the program" << endl;

cout << "Enter your choice: ";

}

**Output:**

1. **Enter the list of numbers again**

****

1. **Display maximum and minimum numbers in the list (don’t use built-in function)**

**A black screen with white text

Description automatically generated**

**c.Display the sum of the squares of each number in the list**

**A black screen with white text

Description automatically generated**

1. **Display the list in ascending and descending orders without changing the original data. You should provide a single sorting function that should sort in ascending or descending order based on the input (don’t use the built-in sort() function)**

**A black screen with white text

Description automatically generated**

1. **Exit the program**

**A black screen with white text

Description automatically generated**

**Task B:**

**Write a modular program that asks the user how many points he/she wants to enter. After that, the program should get each point individually from the user. All the points should be saved in a fixed-sized 2D array such that the first point is placed at index 0, the second point at index 1, and so on. The 2nd dimension should be used to save the X and Y coordinates at index 0 and 1 respectively. Calculate the total distance from the first point to the last point. An example of how to calculate the total distance of 5 points is shown below. Your program should be general which can calculate the total distance between the number of points entered by the user.**

**A diagram of a line with numbers and letters

Description automatically generated**

**The function to calculate the distance between two points is provided below. Note: the function to calculate the distance between two points should take two parameters only, i.e. one for each point.**

**A diagram of a mathematical equation

Description automatically generated**

**Input:**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**const int MAX\_POINTS = 100; // Maximum allowed points**

**// Function Prototypes**

**void inputPoints(double points[][2], int& count);**

**double calculateDistance(const double point1[], const double point2[]);**

**double calculateTotalDistance(const double points[][2], int count);**

**int main() {**

**double points[MAX\_POINTS][2];**

**int pointCount = 0;**

**cout << "Enter the number of points (minimum 2, maximum " << MAX\_POINTS << "): ";**

**cin >> pointCount;**

**if (pointCount < 2 || pointCount > MAX\_POINTS) {**

**cout << "Invalid input. Exiting program." << endl;**

**return 1;**

**}**

**inputPoints(points, pointCount);**

**double totalDistance = calculateTotalDistance(points, pointCount);**

**cout << "Total distance from the first to the last point: " << totalDistance << endl;**

**return 0;**

**}**

**// Collect points' coordinates**

**void inputPoints(double points[][2], int& count) {**

**cout << "Enter the X and Y coordinates for " << count << " points:" << endl;**

**for (int i = 0; i < count; i++) {**

**cout << "Point " << i + 1 << ": ";**

**cin >> points[i][0] >> points[i][1];**

**}**

**}**

**// Calculate distance between two points**

**double calculateDistance(const double point1[], const double point2[]) {**

**return sqrt(pow(point2[0] - point1[0], 2) + pow(point2[1] - point1[1], 2));**

**}**

**// Compute total distance between consecutive points**

**double calculateTotalDistance(const double points[][2], int count) {**

**double totalDistance = 0.0;**

**for (int i = 0; i < count - 1; i++) {**

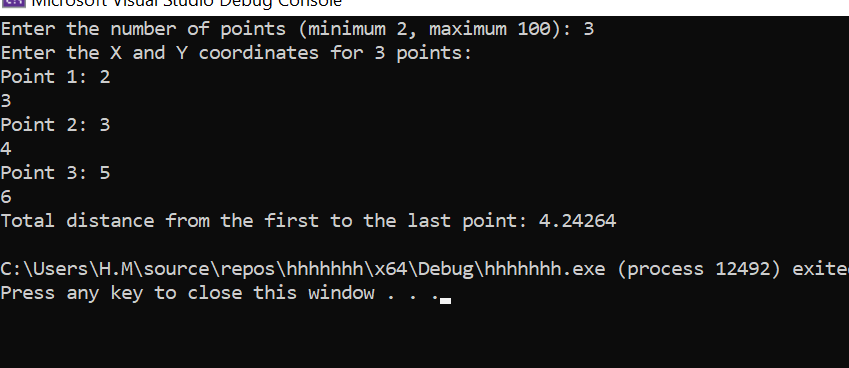
**totalDistance += calculateDistance(points[i], points[i + 1]);**

**}**

**return totalDistance;**

**}**

**Output:**

****