### Program 1: Linear Search with Array of Integers and Negative Values

cpp

#include <iostream>

using namespace std;

int linearSearch(int arr[], int size, int key) {

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

if (arr[i] == key) {

return i;

}

}

return -1;

}

int main() {

int arr[] = {-5, -3, -1, 0, 2, 4};

int size = sizeof(arr) / sizeof(arr[0]);

int key = -3;

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Element found at index: " << result << endl;

} else {

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

}

return 0;

}

### Program 2: Linear Search with Array of Mixed Data Types (Using Variant)

cpp

#include <iostream>

#include <variant>

#include <vector>

using namespace std;

using VariantType = variant<int, double, string>;

int linearSearch(const vector<VariantType>& arr, const VariantType& key) {

for (int i = 0; i < arr.size(); i++) {

if (arr[i] == key) {

return i;

}

}

return -1;

}

int main() {

vector<VariantType> arr = {1, 2.5, "Hello", 3};

VariantType key = "Hello";

int result = linearSearch(arr, key);

if (result != -1) {

cout << "Element found at index: " << result << endl;

} else {

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

}

return 0;

}

### Program 3: Linear Search with Array of Structs and Sorting by Name

cpp

#include <iostream>

#include <algorithm>

using namespace std;

struct Person {

int id;

string name;

};

bool compareByName(const Person& a, const Person& b) {

return a.name < b.name;

}

int linearSearch(Person arr[], int size, const string& key) {

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

if (arr[i].name == key) {

return i;

}

}

return -1;

}

int main() {

Person arr[] = { {1, "Alice"}, {2, "Bob"}, {3, "Charlie"} };

int size = sizeof(arr) / sizeof(arr[0]);

sort(arr, arr + size, compareByName); // Sort by name

string key = "Bob";

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Person found at index: " << result << " with ID: " << arr[result].id << endl;

} else {

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

}

return 0;

}

### Program 4: Linear Search with Array of Pointers to Structs

cpp

#include <iostream>

using namespace std;

struct Student {

int id;

string name;

};

int linearSearch(Student\* arr[], int size, int key) {

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

if (arr[i]->id == key) {

return i;

}

}

return -1;

}

int main() {

Student\* arr[3];

arr[0] = new Student{1, "John"};

arr[1] = new Student{2, "Jane"};

arr[2] = new Student{3, "Doe"};

int key = 2;

int result = linearSearch(arr, 3, key);

if (result != -1) {

cout << "Student found at index: " << result << " with name: " << arr[result]->name << endl;

} else {

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

}

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

delete arr[i]; // Free allocated memory

}

return 0;

}

### Program 5: Linear Search with Array of Strings and Substring Search

cpp

#include <iostream>

#include <string>

using namespace std;

int linearSearch(string arr[], int size, const string& key) {

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

if (arr[i].find(key) != string::npos) {

return i;

}

}

return -1;

}

int main() {

string arr[] = {"apple", "banana", "cherry"};

int size = sizeof(arr) / sizeof (arr[0]);

string key = "nan"; // Substring search

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Substring found in string at index: " << result << endl;

} else {

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

}

return 0;

} ### Program 6: Linear Search with Array of Structs and Age Filtering

cpp

#include <iostream>

using namespace std;

struct Person {

int age;

string name;

};

int linearSearch(Person arr[], int size, int minAge) {

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

if (arr[i].age >= minAge) {

return i;

}

}

return -1;

}

int main() {

Person arr[] = { {25, "Alice"}, {30, "Bob"}, {22, "Charlie"} };

int size = sizeof(arr) / sizeof(arr[0]);

int minAge = 25;

int result = linearSearch(arr, size, minAge);

if (result != -1) {

cout << "Person found at index: " << result << " with name: " << arr[result].name << endl;

} else {

cout << "No person found with age >= " << minAge << "." << endl;

}

return 0;

}

### Program 7: Linear Search with Array of Floats and Average Calculation

cpp

#include <iostream>

using namespace std;

int linearSearch(float arr[], int size, float key) {

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

if (arr[i] == key) {

return i;

}

}

return -1;

}

float calculateAverage(float arr[], int size) {

float sum = 0;

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

sum += arr[i];

}

return sum / size;

}

int main() {

float arr[] = {1.5, 2.5, 3.5, 4.5};

int size = sizeof(arr) / sizeof(arr[0]);

float key = 3.5;

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Element found at index: " << result << endl;

} else {

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

}

cout << "Average of array: " << calculateAverage(arr, size) << endl;

return 0;

}

### Program 8: Linear Search with Array of Integers and Count of Occurrences

cpp

#include <iostream>

using namespace std;

int linearSearch(int arr[], int size, int key, int& count) {

count = 0;

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

if (arr[i] == key) {

count++;

}

}

return count > 0 ? 0 : -1; // Return 0 if found at least once

}

int main() {

int arr[] = {1, 2, 3, 2, 4, 2};

int size = sizeof(arr) / sizeof(arr[0]);

int key = 2;

int count;

int result = linearSearch(arr, size, key, count);

if (result != -1) {

cout << "Element found with count: " << count << endl;

} else {

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

}

return 0;

}

### Program 9: Linear Search with Array of Characters and Vowel Count

cpp

#include <iostream>

using namespace std;

int linearSearch(char arr[], int size, char key) {

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

if (arr[i] == key) {

return i;

}

}

return -1;

}

int countVowels(char arr[], int size) {

int count = 0;

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

if (arr[i] == 'a' || arr[i] == 'e' || arr[i] == 'i' || arr[i] == 'o' || arr[i] == 'u' ||

arr[i] == 'A' || arr[i] == 'E' || arr[i] == 'I' || arr[i] == 'O' || arr[i] == 'U') {

count++;

}

}

return count;

}

int main() {

char arr[] = {'a', 'b', 'c', 'e', 'i'};

int size = sizeof(arr) / sizeof(arr[0]);

char key = 'e';

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Character found at index: " << result << endl;

} else {

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

}

cout << "Total vowels in array: " << countVowels(arr, size) << endl;

return 0;

}

### Program 10: Linear Search with Array of Structs and Sorting by Age

cpp

#include <iostream>

#include <algorithm>

using namespace std;

struct Employee {

int age;

string name;

};

bool compareByAge(const Employee& a, const Employee& b) {

return a.age < b.age;

}

int linearSearch(Employee arr[], int size, int key) {

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

if (arr[i].age == key) {

return i;

}

}

return -1;

}

int main() {

Employee arr[] = { {30, "Alice"}, {25, "Bob"}, {35, "Charlie"} };

int size = sizeof(arr) / sizeof(arr[0]);

sort(arr, arr + size, compareByAge); // Sort by age

int key = 25;

int result = linearSearch(arr, size, key);

if (result != -1) {

cout << "Employee found at index: " << result << " with name: " << arr[result].name << endl;

} else {

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

}

return 0;

}