**Example 1: Program with a Linear Search Algorithm**

// This program demonstrates the searchList function, which

// performs a linear search on an integer array.

#include <iostream>

using namespace std;

// Function prototype

int searchList(const int [], int, int);

const int SIZE = 5;

int main()

{

int tests[SIZE] = {87, 75, 98, 100, 82};

int results;

// Search the array for 100.

results = searchList(tests, SIZE, 100);

// If searchList returned -1, then 100 was not found.

if (results == -1)

cout << "You did not earn 100 points on any test\n";

else

{

// Otherwise results contains the subscript of

// the first 100 in the array.

cout << "You earned 100 points on test ";

cout << (results + 1) << endl;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The searchList function performs a linear search on an \*

// integer array. The array list, which has a maximum of numElems \*

// elements, is searched for the number stored in value. If the \*

// number is found, its array subscript is returned. Otherwise, \*

// -1 is returned indicating the value was not in the array. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int searchList(const int list[], int numElems, int value)

{

int index = 0; // Used as a subscript to search array

int position = -1; // To record position of search value

bool found = false; // Flag to indicate if the value was found

while (index < numElems && !found)

{

if (list[index] == value) // If the value is found

{

found = true; // Set the flag

position = index; // Record the value's subscript

}

index++; // Go to the next element

}

return position; // Return the position, or -1

}

**Example 2: Program with a Binary Search Algorithm**

// This program demonstrates the binarySearch function, which

// performs a binary search on an integer array.

#include <iostream>

using namespace std;

// Function prototype

int binarySearch(const int [], int, int);

const int SIZE = 20;

int main()

{

// Array with employee IDs sorted in ascending order.

int idNums[SIZE] = {101, 142, 147, 189, 199, 207, 222,

234, 289, 296, 310, 319, 388, 394,

417, 429, 447, 521, 536, 600};

int results; // To hold the search results

int empID; // To hold an employee ID

// Get an employee ID to search for.

cout << "Enter the employee ID you wish to search for: ";

cin >> empID;

// Search for the ID.

results = binarySearch(idNums, SIZE, empID);

// If results contains -1 the ID was not found.

if (results == -1)

cout << "That number does not exist in the array.\n";

else

{

// Otherwise results contains the subscript of

// the specified employee ID in the array.

cout << "That ID is found at element " << results;

cout << " in the array.\n";

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The binarySearch function performs a binary search on an \*

// integer array. array, which has a maximum of size elements, \*

// is searched for the number stored in value. If the number is \*

// found, its array subscript is returned. Otherwise, -1 is \*

// returned indicating the value was not in the array. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int binarySearch(const int array[], int size, int value)

{

int first = 0, // First array element

last = size - 1, // Last array element

middle, // Mid point of search

position = -1; // Position of search value

bool found = false; // Flag

while (!found && first <= last)

{

middle = (first + last) / 2; // Calculate mid point

if (array[middle] == value) // If value is found at mid

{

found = true;

position = middle;

}

else if (array[middle] > value) // If value is in lower half

last = middle - 1;

else

first = middle + 1; // If value is in upper half

}

return position;

}

**Example 3: Program with a Bubble Sort Algorithm**

// This program uses the bubble sort algorithm to sort an

// array in ascending order.

#include <iostream>

using namespace std;

// Function prototypes

void sortArray(int [], int);

void showArray(const int [], int);

int main()

{

// Array of unsorted values

int values[6] = {7, 2, 3, 8, 9, 1};

// Display the values.

cout << "The unsorted values are:\n";

showArray(values, 6);

// Sort the values.

sortArray(values, 6);

// Display them again.

cout << "The sorted values are:\n";

showArray(values, 6);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of function sortArray \*

// This function performs an ascending order bubble sort on \*

// array. size is the number of elements in the array. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void sortArray(int array[], int size)

{

bool swap;

int temp;

do

{

swap = false;

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

{

if (array[count] > array[count + 1])

{

temp = array[count];

array[count] = array[count + 1];

array[count + 1] = temp;

swap = true;

}

}

} while (swap);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of function showArray. \*

// This function displays the contents of array. size is the \*

// number of elements. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void showArray(const int array[], int size)

{

for (int count = 0; count < size; count++)

cout << array[count] << " ";

cout << endl;

}

**Example 4: Program with a Selection Sort Algorithm**

// This program uses the selection sort algorithm to sort an

// array in ascending order.

#include <iostream>

using namespace std;

// Function prototypes

void selectionSort(int [], int);

void showArray(int [], int);

int main()

{

// Define an array with unsorted values

const int SIZE = 6;

int values[SIZE] = {5, 7, 2, 8, 9, 1};

// Display the values.

cout << "The unsorted values are\n";

showArray(values, SIZE);

// Sort the values.

selectionSort(values, SIZE);

// Display the values again.

cout << "The sorted values are\n";

showArray(values, SIZE);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of function selectionSort. \*

// This function performs an ascending order selection sort on \*

// array. size is the number of elements in the array. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void selectionSort(int array[], int size)

{

int startScan, minIndex, minValue;

for (startScan = 0; startScan < (size - 1); startScan++)

{

minIndex = startScan;

minValue = array[startScan];

for(int index = startScan + 1; index < size; index++)

{

if (array[index] < minValue)

{

minValue = array[index];

minIndex = index;

}

}

array[minIndex] = array[startScan];

array[startScan] = minValue;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of function showArray. \*

// This function displays the contents of array. size is the \*

// number of elements. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void showArray(int array[], int size)

{

for (int count = 0; count < size; count++)

cout << array[count] << " ";

cout << endl;

}