

Sorting and Searching

- **Problem:**

Read in a parameter value n , then read in a set of n numbers. Print the numbers in their original order. Sort the numbers into ascending order. Finally, print the numbers in sorted order.

e.g.,

Original Order
14
125
11
4
21

Sorted Order
4
11
14
21
125

The Linear or Selection Sort

- **Algorithm for the Linear or Selection Sort:**
for each position in the array (except the last)
for each candidate for that position
compare the candidate to the element currently in that position
if the candidate is smaller
swap them

- **Trace of the Linear Sort for the Above Example:**

- **A Function to Implement a Linear Sort:**

```
/* Function linearsort()
 * Input:
 *   numb - array to be sorted
 *   n - number of elements to sort
 * Process:
 *   linear sort
 * Output:
 *   numb sorted into ascending order
 */
void linearsort(int numb[], int n)
{
    int temp;

    for (int pos = 0; pos < (n-1); pos + +)
        for (int cand = (pos + 1); cand < n; cand + +)
            if (numb[pos] > numb[cand])
                {
                    temp = numb[pos];
                    numb[pos] = numb[cand];
                    numb[cand] = temp;
                }
    return;
}
```

● **Main Program:**

```
/* program to illustrate linear sort */
#include <iostream>
using namespace std;
const int MAXSIZE = 100;

/* Function Prototypes */
void linearsort(int [], int);
void printarray(int [], int);

int main()
{
    int numb[MAXSIZE];
    int n;

    cout << "Enter the number of elements in the array: "
         << endl;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter a number into the array: ";
        cin >> numb[i];
    }
    cout << endl << "Original Data" << endl;
    printarray(numb,n);

    linearsort(numb,n);

    cout << endl << "Sorted Data" << endl;
    printarray(numb,n);
    return 0;
}

/* Function to print an array */
void printarray(int numb[], int n)
{
    for (int i = 0; i < n; i++)
        cout << numb[i] << endl;
    return;
}
```

The Bubble Sort

- **Algorithm for the Bubble Sort:**

*do the following as long as there has been a swap on the last pass
for each element of the array
compare the element to its neighbor
if they are out of order swap them*

- **A Function to Implement a Bubble Sort:**

```
/* Function bubblesort()
 * Input:
 *   numb - array to be sorted
 *   n - number of elements to sort
 * Process:
 *   bubble sort
 * Output:
 *   numb sorted into ascending order
 */
```

```
void bubblesort(int numb[], int n)
{
    bool swapped;
    int temp;

    do {
        swapped = false;    // initialize swapped to FALSE
        for (int pos = 0; pos < (n-1); pos + +)
            if (numb[pos] > numb[pos + 1])
            {
                temp = numb[pos];
                numb[pos] = numb[pos + 1];
                numb[pos + 1] = temp;
                swapped = true; // indicate swap has occurred
            }
    } while (swapped);
    return;
}
```

Sorting an Array of Strings

```
/* Program to illustrate sorting an array of strings */
#include <iostream>
#include <string>
using namespace std;

const int SIZE = 5;

int main()
{
    string str[SIZE];
    string tempstr;

    str[0] = "Jones";
    str[1] = "Harrow";
    str[2] = "Tenenbaum";
    str[3] = "Arnow";
    str[4] = "Raphan";

    //print the original array
    cout << "Original Array" << endl;
    for (int i = 0; i < SIZE; i++)
        cout << str[i] << endl;

    // sort the array
    for (int pos = 0; pos < SIZE-1; pos++)
        for (int cand = pos+1; cand < SIZE; cand++)
            if (str[cand] < str[pos])
            {
                tempstr = str[cand];
                str[cand] = str[pos];
                str[pos] = tempstr;
            }

    //print the sorted array
    cout << endl << "Sorted Array" << endl;
    for (int i = 0; i < SIZE; i++)
        cout << str[i] << endl;

    return 0;
}
```

Sorting an Array of Strings in a Function

```
/*Program to illustrate sorting an array of strings in a function*/
#include <iostream>
#include <string>
using namespace std;

const int SIZE = 5;

//function prototype
void sortthearray(string str[], int n);

int main()
{
    string str[SIZE];
    string tempstr;

    str[0] = "Jones";
    str[1] = "Harrow";
    str[2] = "Tenenbaum";
    str[3] = "Arnow";
    str[4] = "Raphan";

    //print the original array
    cout << "Original Array" << endl;
    for (int i = 0; i < SIZE; i++)
        cout << str[i] << endl;

    // sort the array
    sortthearray(str, SIZE);

    //print the sorted array
    cout << endl << "Sorted Array" << endl;
    for (int i = 0; i < SIZE; i++)
        cout << str[i] << endl;

    return 0;
}
```

```

/* Function sortthearray()
 * Input:
 *   str - the array to be sorted
 *   n - number of elements to sort
 * Process:
 *   linear sort
 * Output:
 *   str sorted into alphabetical order
 */
void sortthearray(string str[], int n)
{
    string tempstr;

    // sort the array
    for (int pos = 0; pos < n-1; pos + +)
        for (int cand = pos + 1; cand < n; cand + +)
            if (str[cand] < str[pos])
                {
                    tempstr = str[cand];
                    str[cand] = str[pos];
                    str[pos] = tempstr;
                }
    return;
}

```

Searching

- **Problem:**

We have an array of values:

23

-65

89

99

76

-56

95

Search for:

99

77

and if found, return the position of the value within the array
(return -1 if not found).

Linear Search

- **Pseudocode for Linear Search:**

*for each position in the array
compare the element in that position to the search value
if they are equal
return the position in the array
if no element is equal to the search value
return a failure signal (-1)*

- **Function to Implement a Linear Search:**

```
/* Function linearsearch()
 * Input:
 *   numb - array to be searched
 *   n - number of elements in the array
 *   searchvalue - the value to search for
 * Process:
 *   linear search
 * Output:
 *   returns the position of searchvalue in the array
 *   returns -1 if searchvalue not found
 */
int linearsearch(int numb[], int n, int searchvalue)
{
    for (int position = 0; position < n; position + + )
        if (numb[position] == searchvalue)
            return (position);           // search value found
    return (-1);                         // search value not found
}
```

Binary Search of Sorted Array

- **Pseudocode for Binary Search:**

low = 0;

high = n-1;

while (low <= high)

look at the element halfway between low and high

if the element equals the search value

return its position

else if the element is larger than the search value

high = the tested position - 1

else

low = tested position + 1

e.g.,

-65

-56

23

76

89

95

99

● Function to Implement a Binary Search:

```
/* Function binarysearch()
 * Input:
 *   numb - array to be searched
 *   n - number of elements in the array
 *   searchvalue - the value to search for
 * Process:
 *   binary search
 * Output:
 *   returns the position of searchvalue in the array
 *   returns -1 if searchvalue not found
 */
int binarysearch(int numb[], int n, int searchvalue)
{
    int low,high,test;

    low = 0;
    high = n - 1;
    while (low <= high) {
        test = (low + high) / 2;
        if (numb[test] == searchvalue)
            return (test);           // search value found
        else if (numb[test] > searchvalue)
            high = test - 1;
        else
            low = test + 1;
    }
    return (-1);                    // search value not found
}
```