**Linear (Sequential) Search**

**Definition:** Search an array or list by checking items one at a time.

Linear search is usually very simple to implement, and is practical when the list has only a few elements, or when performing a single search in an unordered list. When many values have to be searched in the same list, it often pays to pre-process the latter in order to use a faster method. For example, one may sort the list and use binary search.

**Look at every element**

This is a very straightforward loop comparing every element in the array with the key. As soon as an equal value is found, it returns. If the loop finishes without finding a match, the search failed and -1 is returned.

**Performance**

For small arrays, linear search is a good solution because it's so straightforward. In an array of a million elements linear search on average will take 500,000 comparisons to find the key.

**Example:** we'll just use an array of integers, like the following:

![Array](image)

*Figure 1: The array we're searching*

Let's search for the number 3. We start at the beginning and check the first element in the array. Is it 3?

![Array](image)

*Figure 2: Is the first value 3?*

No, not it. Is it the next element?
Figure 3: Is the second value 3?

Not there either. The next element?

Figure 4: Is the third value 3?

Not there either. Next?

Figure 5: Is the fourth value 3? Yes!

We found it!!! Now you understand the idea of linear searching; we go through each element, in order, until we find the correct value.

```c
int linearSearch(int a[], int first, int last, int key) {
    // function:
    //   Searches a[first]..a[last] for key.
    // returns: index of the matching element if it finds key,
    //         otherwise -1.
    // parameters:
    //   a       in  array of (possibly unsorted) values.
    //   first, last in  lower and upper subscript bounds
    //   key     in  value to search for.
    // returns:
    //   index of key, or -1 if key is not in the array.
    for (int i=first; i<=last; i++) {
```
```cpp
if (key == a[i]) {
    return i;
}
}
return -1;  // failed to find key

#include <iostream>
using namespace std;

int LinearSearch(const int *Array, const int Size, const int ValToSearch)
{
    bool NotFound = true;
    int i = 0;
    while(i < Size && NotFound)
    {
        if(ValToSearch != Array[i])
            i++;
        else
            NotFound = false;
    }
    if( NotFound == false )
        return i;
    else
        return -1;
}

int main()
{
    int Number[] = { 67, 278, 463, 2, 4683, 812, 236, 38 };  
    int Quantity = sizeof(Number) / sizeof(int);  
    int NumberToSearch = 0;
    cout << "Enter the number to search: "; cin >> NumberToSearch;
    int i = LinearSearch(Number, Quantity, NumberToSearch);
    if(i == -1)
        cout << NumberToSearch << " was not found in the collection\n\n";
    else
    {
        cout << NumberToSearch << " is at the " << i+1;
        if( i == 0 )
            cout<< "st position of the collection\n\n";
        else if( i == 1 )
            cout<< "nd position of the collection\n\n";
        else if( i == 2 )
            cout<< "rd position of the collection\n\n";
        else
            cout<< "th position of the collection\n\n";
    }
}
return 0;
}

Here is an example of running the program:
Enter the number to search: 278
278 is at the 2nd position of the collection
Press any key to continue

Here is another example of running the program:
Enter the number to search: 288
288 was not found in the collection
Press any key to continue