

## **Algorithms and Data Structures Study the Most Popular Algorithms and Data Structures (Arrays, Lists, Drives, Graphs, Search and Sorting Algorithms) and Find Practical Solutions to Them**

*Qahhorova Nargiza Hayit qizi*  
*qahhorovanargiza02@gmail.com*

***Abstract.** Algorithm means the sum of the sequence of calculation or problem solving processes. Algorithms do not depend on any programming language, they are instructions that lead to the same result even if the code is written in any language.*

*But not all instructions are algorithms. To be an algorithm, an instruction has several characteristics.*

***Keywords:** Algorithm, array, database, index.*

**Enter.**

### **Algorithm Complexity**

Algorithm complexity is a value measured based on the amount of Space and Time. Time is the time spent by the program during its operation; Space is the amount of space needed for input, variables, and output. These two factors determine the effectiveness of the algorithm. Good algorithms have a small amount of Space and Time, which increases the speed and performance of the code written in that algorithm.

Now the question may arise, why should we care about the efficiency of an algorithm to solve a simple problem? Or is it necessary to think about performance when using frameworks in the programming process?

In many cases, large projects have to perform operations on millions of rows of data. Then, the performance problem of the code, which is not noticeable in small problems, is clearly visible in the processing of large amounts of data. A performance problem affects the speed of the code, causing it to take up more memory space during operation.

As the main goal is to get a job at a tech giant corporation, it is necessary to prioritize performance, analyze the written code, and determine the complexity.

Analysis of algorithms is also called Asymptotic analysis. In this case, the running time of the algorithm is calculated depending on the size of the entered input.

As an example of asymptotic analysis, we compare Linear Search and Binary Search algorithms to find a given number X from a sorted array.

Linear search checks the array elements one by one starting from the first element of the array until X is found. Assuming the length of the array is N, to find X, a minimum of 1 maximum, N comparison is performed.

In computer science, a data structure is a format for organizing, managing, and storing data that allows efficient access and modification. More specifically, a data structure is a set of data values, relationships between them, and functions or operations that can be applied to the data.

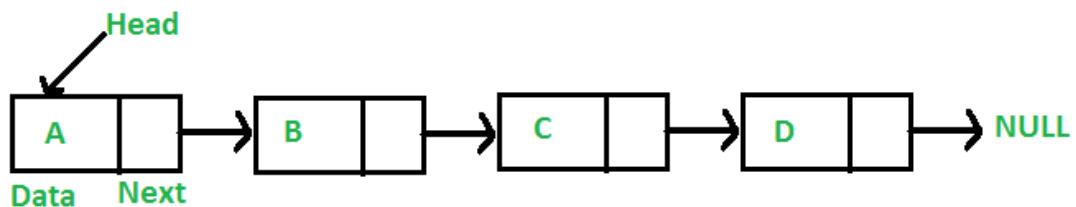
Data structure (data structures) is a standardized form of data storage and management that allows for efficient reading and changing of data. We can show an array as an example of the simplest data structure. (Source)

Data structures are divided into primitive and non-primitive types.

- Primitive type includes Integer, Boolean, Text, Character.
- The non-primitive type itself is divided into two:
  - ✓ Linear: Array, Linked List, Stack, Queues
  - ✓ Non-linear: Trees, Graphs.

Assuming that you, as a programmer, know about primitive types of data structures, let's get acquainted with non-primitive.

**Array** An array is a collection of data, of the same type or of different types, depending on the programming language; size can be predefined or undefined. Since the elements of the array are sequentially located in the memory, its reading is done quickly. Note: We don't care about other languages since we've covered the examples in Javascript ES6. Array operations const arr = [4,6,1,8,3,0,3,7,3,14,9] Read element (Get). in  $O(1)$  time If we know the index of the array, then we can read the element immediately. For example, to get the number 6, it is enough to use arr[1]. Add element (Insert). At time  $O(n)$ . Add to the beginning of the array - unshift To add to the beginning of the array, all elements are shifted to one next index and insert the new element at index 0. Add to end of array - push Goes to the last element of the array and inserts a new element by incrementing the index by one. Delete element (Remove). At time  $O(n)$ . Delete from the beginning of the array - shift Frees the array at index 0 and moves the elements from the next index to the previous index. Delete from end of array - pop Deletes the last index of the array Change the element (Update). in  $O(1)$  time In this case, since the index is known, it simply replaces the element in the index with the new one. Traverse. At time  $O(n)$ . Array elements are considered in the loop. **Linked list** A linked list is a linear structure consisting of nodes, each node stores an element and a pointer indicating the address of the next (and sometimes previous) node.



### Linked list structure

The initial Node is called the head. Nodes after Head are linked to the pointer of the previous node. So, knowing the first node, i.e. head, we find the desired element by moving to the next nodes with the help of pointers. If we find out the index number in the array and immediately read the element with arr[index], in order to get to that element in the linked list, it is necessary to go through all the elements before it.

In a real-life example, the array is a hotel corridor. Knowing the room number, you will immediately find the room. Linked List is a guest room inside the house. In order to enter the room, one must first enter the house, go to the porch, to the corridor.

A database is a centralized data repository. Designed for reading, storing, processing and searching data. Distinguished by its speed

An array is an ordered collection of finite values of the same type. Examples of arrays are vectors and matrices known from the mathematics course.

Arrays are usually divided into one-dimensional and multi-dimensional types.

An array is called one-dimensional, if its element can be referred to by one index.

Indexes of array elements in C\C++ programming languages always start from zero (not one). Let us be given an array named m of char type. And let it consist of 3 elements.

```
m[0] à -9 ;
```

```
m[1] à 15;
```

```
m[2] à 3;
```

So, to refer to an element, the array name and the element index are written in [] brackets.

Here, the value of the first element is -9, and the second element is -15 in index number 1. The index of the last element will be n-1 (the number of n-array elements). The index in parentheses [] must be an integer or an expression leading to an integer. For example:

```
int n=6, m=4;
```

```
L[n-m]=33; // L[2]=33;
```

```
Cout<<m[2]; // on screen : 3;
```

Addressing array elements is slightly different from addressing simple variables.

➤ The initial value for all elements of the array with the specified size is 0

```
int k[5] = {0};
```

For example:

Example 1. Initialize 0 to all elements of the specified size array.

```
#include
```

```
int main()
```

```
{
```

```
    int k[5]={0}; // assign a value of 0 to all elements of the array.
```

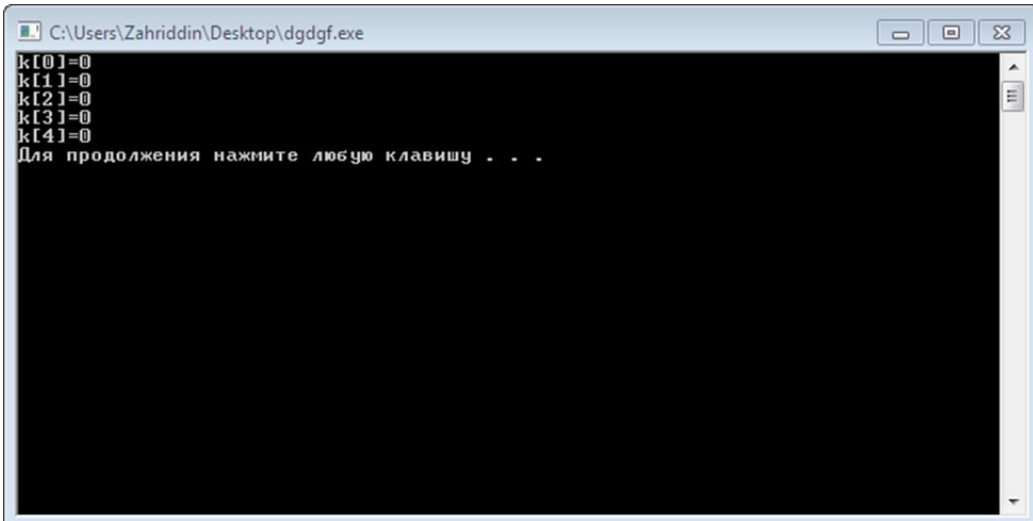
```
    for (int i=0; i<5; i++ )
```

```
        cout<<"k["<<i<<"]="<<k[i]<<endl;
```

```
    return 0;
```

```
}
```

The following will appear on the screen:



```
C:\Users\Zahriddin\Desktop\dgdgf.exe
k[0]=0
k[1]=0
k[2]=0
k[3]=0
k[4]=0
Для продолжения нажмите любую клавишу . . .
```

Example 2. Fully initialize an array of the specified size.

```

#include
int main()
{
    int k[5] = { 2, -9, 112, 3, 8 };
    for (int i=4; i>=0; i-- ) // print indices in reverse order.
        cout<<"k["<<i<<"]="<<k[i]<<endl;
    return 0; }

```

The following will appear on the screen:

```

F:\kurs i2.exe
k[4]=10
k[3]=3
k[2]=112
k[1]=-9
k[0]=2
Для продолжения нажмите любую клавишу . . .

```

### References:

1. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms, Third Edition, 3rd, The MIT Press, 2009. [[Special: Book Sources/978-0262033848|ISBN 978-0262033848]].
2. Black, Paul E. "data structure". Dictionary of Algorithms and Data Structures [online]. National Institute of Standards and Technology, 15 December 2004.
3. "Data structure". Encyclopaedia Britannica. April 17, 2017.
4. Wegner, Peter; Reilly, Edwin D.. Encyclopedia of Computer Science. Chichester, UK: John Wiley and Sons, 2003-08-29 — pp. 507–512. [[Special: Book Sources/978-0470864128|ISBN 978-0470864128]].