Arrays and Sorting

Lab 07 - CSF111

Introduction to Loops

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.

TYPES OF LOOPS IN C:

for loop: first Initializes, then condition check, then executes the body and at last, the update is done.

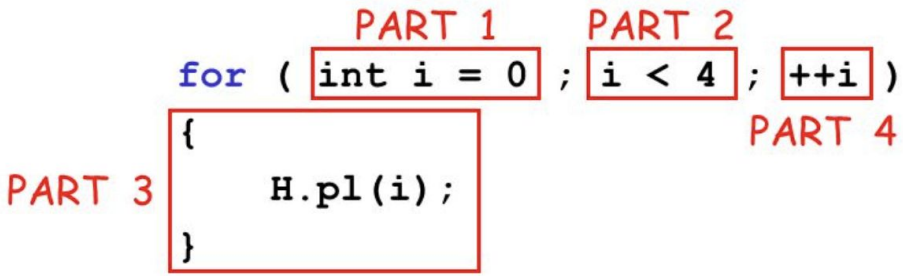
while loop: first Initializes, then condition checks, and then executes the body, and updating can be inside the body.

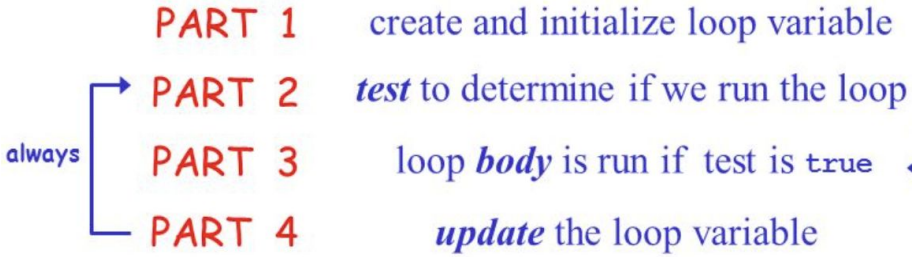
do-while loop: do-while first executes the body and then the condition check is done.

for loop

for loop in C programming is a repetition control structure that allows programmers to write a loop that will be executed a specific number of times.

Syntax:





Example 1:- Print all even numbers less than n

Step-1:- Identify number of times we need to run the loop ⇒ i<n

Step-2:- Identify the body of loop ⇒ if(i%2==0) printf("%d ",i);

Step-3:- Merge everything and make the loop ⇒

```
int n=20;
for(int i=0;i<n;i++){
    if(i%2==0) printf("%d ",i);
}
```

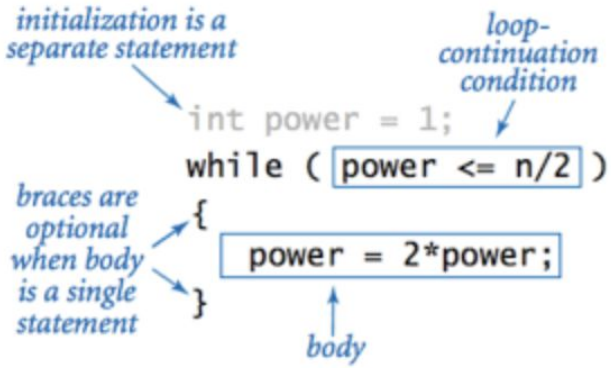
Output:-

```
tanishdesai37@Tanishs-MacBook-Air Demo % ./a.out
0 2 4 6 8 10 12 14 16 18 %
```

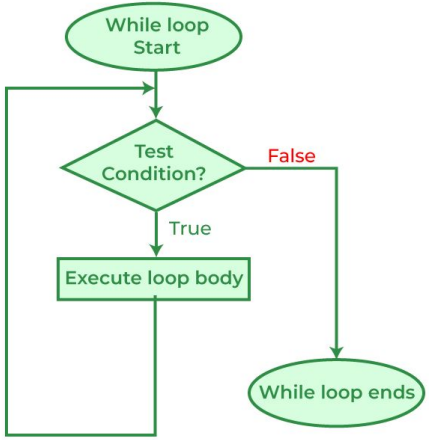
while loop

It is a special kind of loop which execute a set of statements as long as a condition is true.

Syntax:



Flow:



Example 2:- Make a random number guessing game

Step-1:- Identify the condition ⇒ (n!=a)

Step-2:- Body of loop ⇒

```
if(n>a) printf("Incorrect guess,Try with a bigger number\n");
else printf("Incorrect guess,Try with a smaller number\n");
scanf("%d",&a);
```

Step-3:- Merge everything and make the loop ⇒

```
int n=11;//Correct guess
int a;//User Input
scanf("%d",&a);
while(n!=a){
    if(n>a) printf("Incorrect guess,Try with a bigger number\n");
    else printf("Incorrect guess,Try with a smaller number\n");
    scanf("%d",&a);
}
printf("Correct Guess!\n");
```

Output:-

```
tanishdesai37@Tanishs-MacBook-Air Demo % ./a.out
3
Incorrect guess,Try with a bigger number
5
Incorrect guess,Try with a bigger number
15
Incorrect guess,Try with a smaller number
11
Correct Guess!
```

Example 3:- C code to print the given pattern for any n

```
********
*******
******
*****
****
***
**
*
```

n=8

Step-1:- Identify the condition ⇒ Need 2 loops so 2 conditions

1. For 1st loop ⇒ It runs n times ⇒ (i<n)
2. For 2rd loop ⇒ It runs (n-i) times ⇒ (j<n-i)

Step-2:-Code ⇒

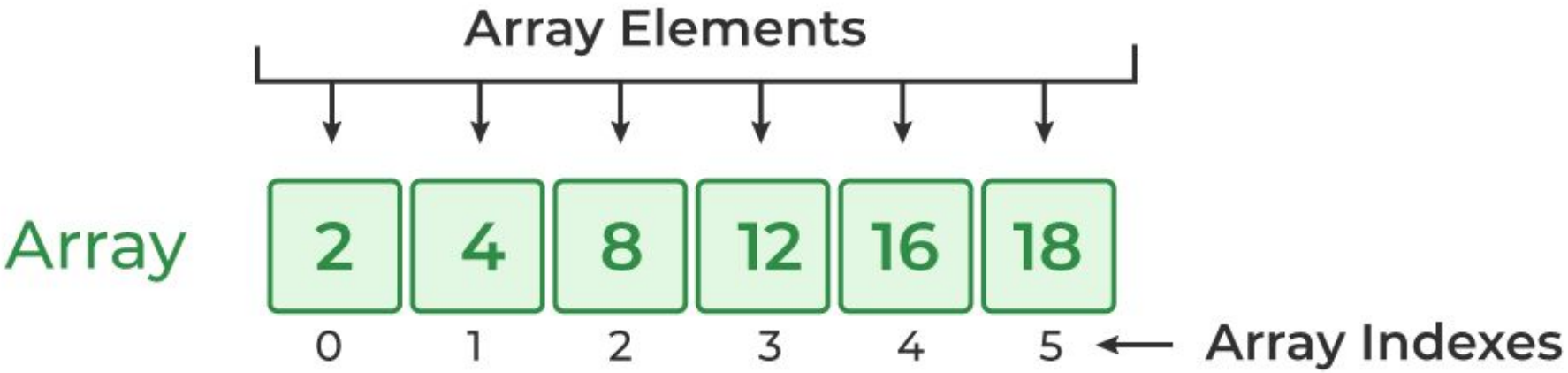
```c
int n = 8;
for(int i=0;i<n;i++){
    for(int j=0;j<(n-i);j++){
        printf("*");
    }
    printf("\n");
}
```

What is an array?

An Array is one of the most used data structures in programming. It is a simple and fast way of storing multiple values under a single name.

An array in C is a fixed-size collection of similar data items stored in contiguous memory locations. It can be used to store the collection of primitive data types such as int, char, float, etc., and also derived and user-defined data types such as pointers, structures, etc.

Array in C



Properties of an Array

Fixed Size: The array in C is a fixed-size collection of elements. The size of the array must be known at the compile time and it cannot be changed once it is declared.

Homogeneous Collection: We can only store one type of element in an array. There is no restriction on the number of elements but the type of all of these elements must be the same.

Indexing in Array: The array index always starts with 0 in C language. It means that the index of the first element of the array will be 0 and the last element will be N – 1.

Random Access: The array in C provides random access to its element i.e we can get to a random element at any index of the array just by using its index number.

No Index Out of Bounds Checking: There is no index out-of-bounds checking in C.

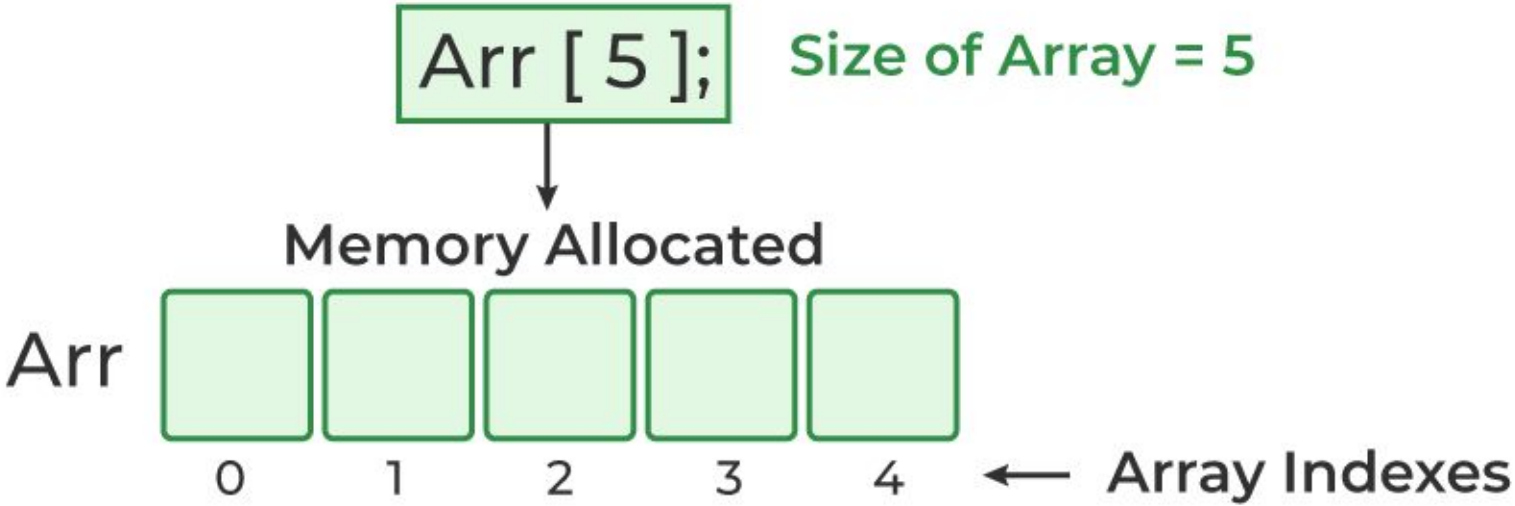
Array Declaration

In C, we have to declare the array like any other variable before using it. We can declare an array by specifying its name, the type of its elements, and the size of its dimensions. When we declare an array in C, the compiler allocates the memory block of the specified size to the array name.

Syntax:

```
data_type array_name [size];
```

Array Declaration



Array Declaration

```
// declaring array of integers
int arr_int[5];
// declaring array of characters
char arr_char[5];
```

Array Initialization

Initialization in C is the process to assign some initial value to the variable. When the array is declared or allocated memory, the elements of the array contain some garbage value. So, we need to initialize the array to some meaningful value. There are multiple ways in which we can initialize an array in C.

1. Array Initialization with Declaration

In this method, we initialize the array along with its declaration. We use an initializer list to initialize multiple elements of the array. An initializer list is the list of values enclosed within braces **{ }** separated by a comma.

Syntax:

```
data_type array_name [size] = {value1, value2, ... valueN};
```

Array Initialization

```
Arr [ 5 ] = { 2, 4, 8, 12, 16 };
```

Memory Allocated and Initialized

Arr	2	4	8	12	16
	0	1	2	3	4

← Array Indexes

2. Array Initialization with Declaration without Size

If we initialize an array using an initializer list, we can skip declaring the size of the array as the compiler can automatically deduce the size of the array in these cases. The size of the array in these cases is equal to the number of elements present in the initializer list as the compiler can automatically deduce the size of the array.

Syntax:

```
data_type array_name[] = {1,2,3,4,5};
```

The size of the above arrays is 5 which is automatically deduced by the compiler.

3. Array Initialization after Declaration (Using Loops)

We initialize the array after the declaration by assigning the initial value to each element individually. We can use for loop, while loop, or do-while loop to assign the value to each element of the array.

Syntax:

```
for (int i = 0; i < N; i++) {
    array_name[i] = valuei;
}
```

Access Array elements

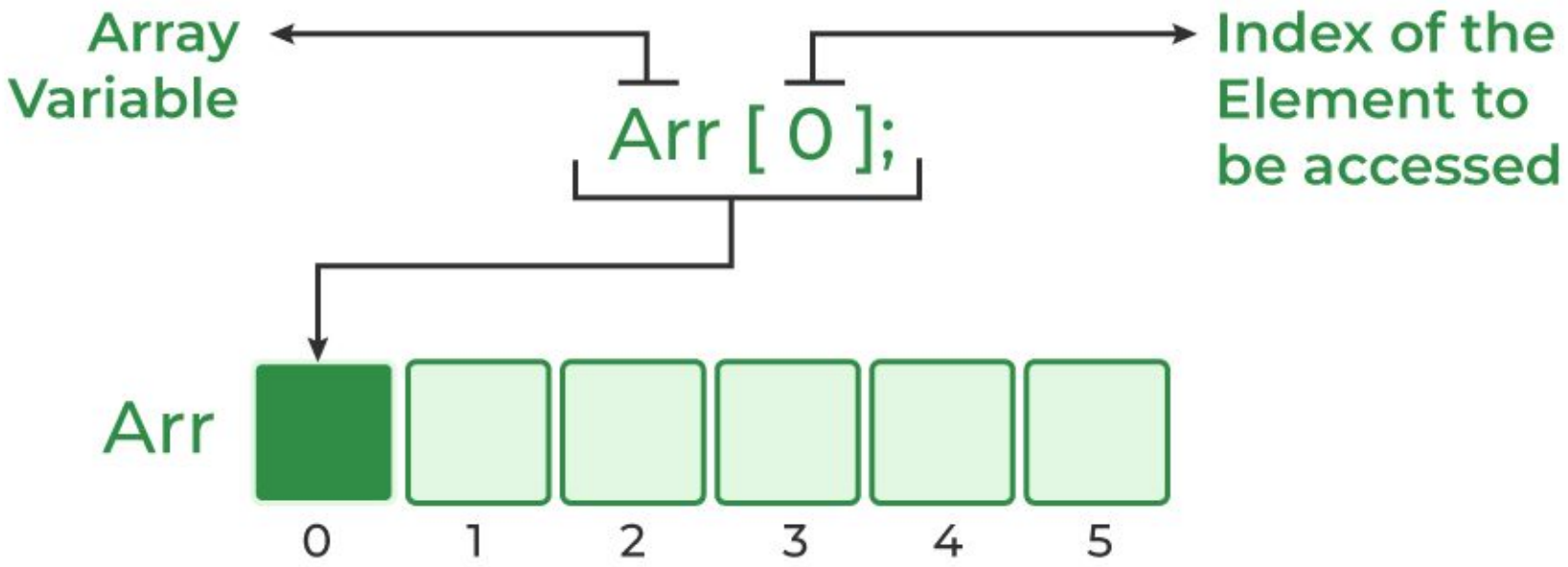
We can access any element of an array in C using the array subscript operator **[ ]** and the index value ***i*** of the element.

Syntax:

```
array_name[index];
```

One thing to note is that the indexing in the array always starts with 0, i.e., the **first element** is at index **0** and the **last element** is at **N – 1** where **N** is the number of elements in the array.

Access Array Element



Update Array element

We can update the value of an element at the given index i in a similar way to accessing an element by using the array subscript operator **[ ]** and assignment operator **=**.

Syntax:

```
Array_name[index] = new value;
```

Example 4:- Create an array with 5 numbers and print the number which is at index 3

Step-1:- Initialize the array ⇒

```
int nums[5] = {5,3,12,4,1};
```

Step-2:- Accessing and printing 3rd element ⇒

```
printf("%d",nums[2]);
```

Output ⇒

```
tanishdesai37@Tanishs-MacBook-Air Demo % ./a.out
12%
```

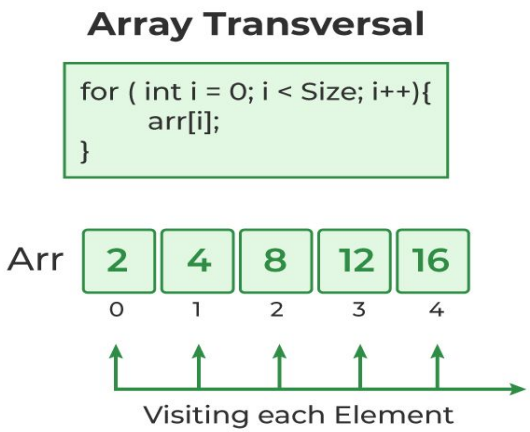
Note:- To access 3rd element we have passed index as 2 because index of array starts from 0 in C.

Array traversal

Traversal is the process in which we visit every element of the data structure. For C array traversal, we use loops to iterate through each element of the array.

Syntax using for loop:

```
for (int i = 0; i < N; i++) {
    array_name[i];
}
```



Passing array as a function argument

- There are 3 ways to pass array in a function first is without size,second is with size and third is using pointer(will learn later)

Passing Array in Functions

```
void func1(int arr[]){
    //Function Body
}
```

```
void func1(int arr[5]){
    //Function Body
}
```

Code Along Ahead

Be ready with your VS Code we would be solving 3 problems using loops and arrays

Problem-1:- Implementing linear search in an array using for loop
Problem-2:- Using

Example 5:- Find the first occurrence of an element in an array

Step-1:- Identify number of times we need to run the loop ⇒ (i<size of array)

Step-2:- Identify the body of loop ⇒

```
if(nums[i]==element)
{
    printf("Found element %d at index %d\n",element,i);
    break;
}
```

Step-3:- Complete Code ⇒

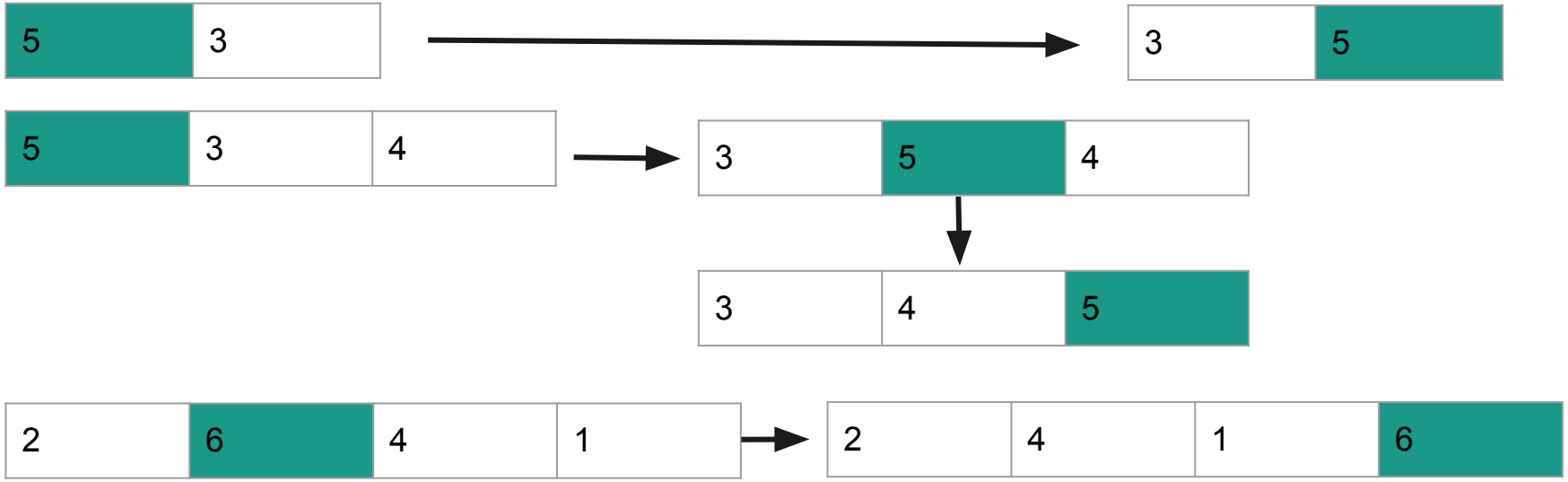
```
int element = 10;
int nums[8] = {5,3,12,4,1,10,11,15};
for(int i=0;i<8;i++){
    if(nums[i]==element){printf("Found element %d at index %d\n",element,i);break;}
}
```

Output:-

```
tanishdesai37@Tanishs-MacBook-Air Demo % ./a.out
Found Element 10 at index 5
```

Example 6:- Swap all the adjacent elements if first element is more then second element in array

Can you identify the pattern?



Code

Step-1:- Identify number of times we need to run the loop ⇒ (i<size of array-1)

Step-2:- Identify the body of loop ⇒

```
if(arr[i]>arr[i+1]){
    int temp = arr[i];
    arr[i] = arr[i+1];
    arr[i+1] = temp;
}
```

Step-3:- Complete Code ⇒

```
for(int i=0;i<n-1;i++){
    if(arr[i]>arr[i+1]){
        int temp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = temp;
    }
}
```

→ Loop which runs n-1 times

→ Swap elements of array if greater

Output ⇒

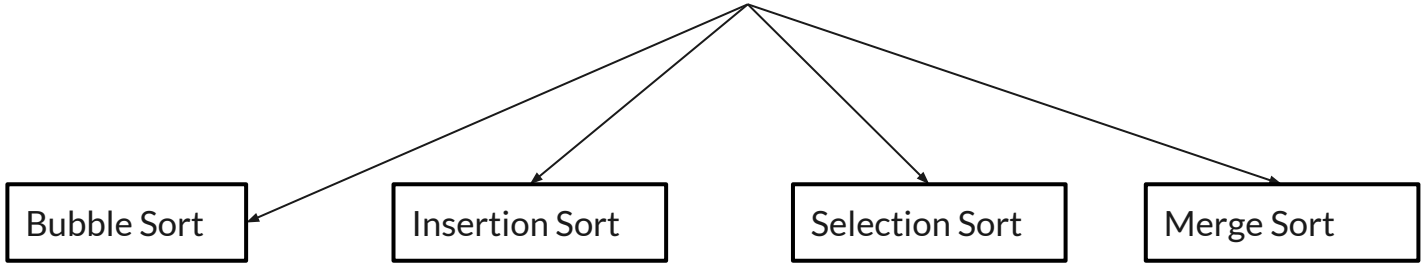
```
tanishdesai37@Tanishs-MacBook-Air Downloads % ./a.out

4 6 6 3 7 1 9 5 0 10
```

Sorting

- *Sorting an array means arranging the elements of the array in a certain order. Generally sorting in an array is done to arrange the elements in increasing or decreasing order.*
- *Examples:-*
  - *Unordered Array = {1,5,3,2,6}*
  - *Sorting array in ascending order = {1,2,3,5,6}*
  - *Sorting array in descending order = {6,5,3,2,1}*

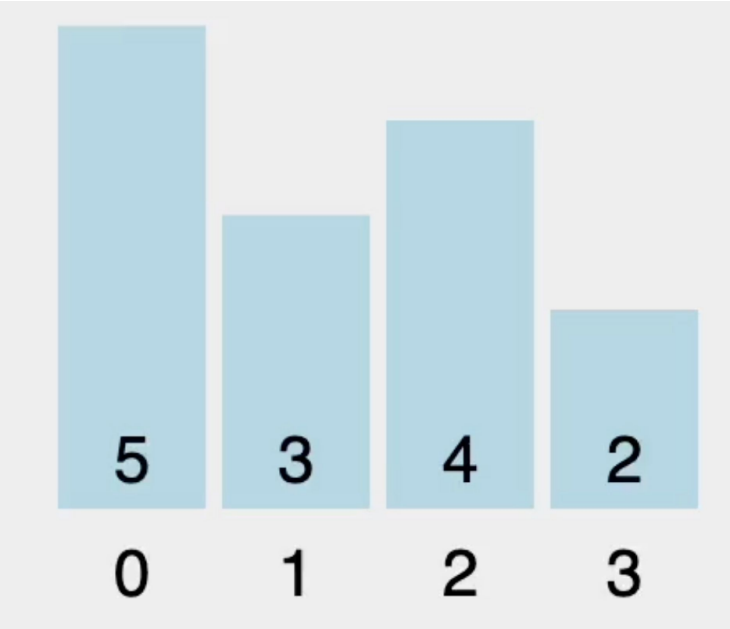
Some common sorting algorithms



Bubble Sort

- **Bubble Sort** is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.
- It starts from the first element of an array and compares it with the second element. If the first element is greater than the second, we swap them. It continues this process until the end of the array, with the largest elements “bubbling” to the top.The last element is locked and the process starts again with first element of array till second last and and so on.

Bubble Sort Animated

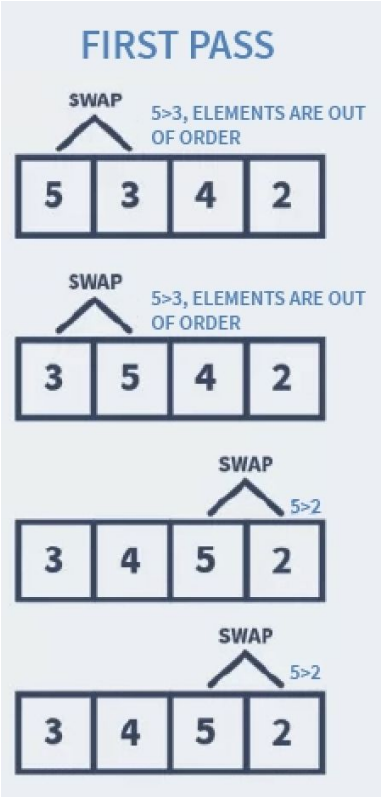


Pseudocode

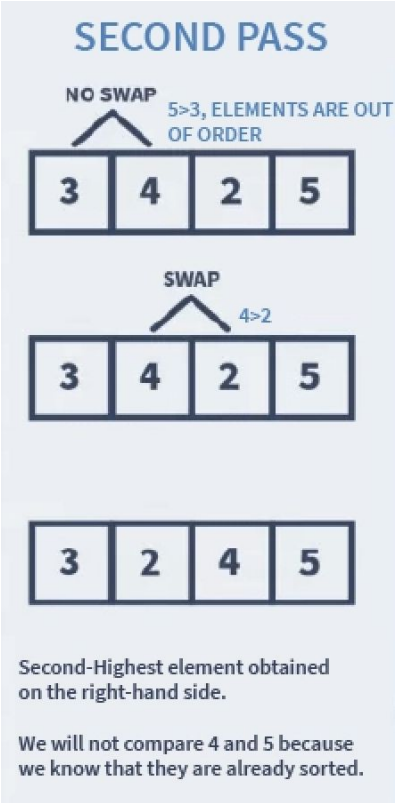
```
BUBBLE SORT(A)
    for i = 0 to Size of Array - 1
        for j = 0 to Size of Array - i - 1
            if A[j] < A[j - 1]
                exchange A[j] with A[j - 1]
```

Dry Run

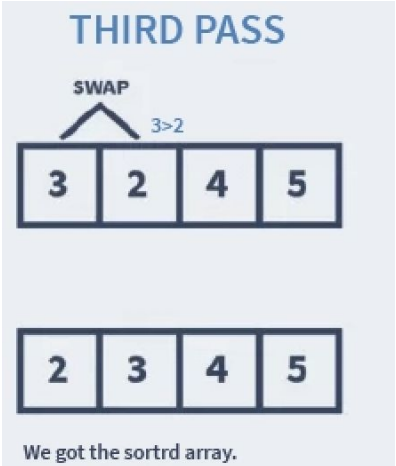
i=0



i=1



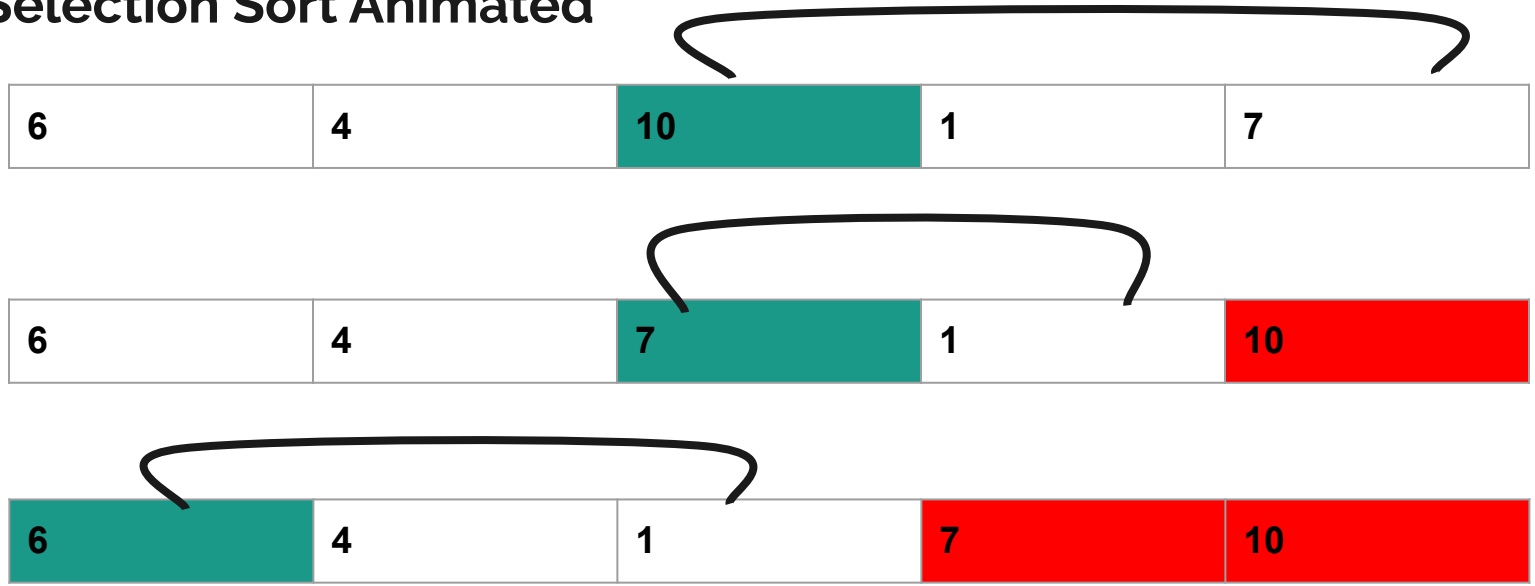
i=2



Selection Sort

- **Selection sort** is a simple and efficient sorting algorithm that works by repeatedly selecting the largest element from the unsorted portion of the list and moving it to the sorted portion of the list.
- It starts from the first element of the and finds the largest element till last index from the array and swap it with last element and locks last element, then it again starts from the first element again and finds the largest element till second last index and swap it with the second last element and so on.
- It divides the array into 2 parts sorted and unsorted and each time we iterate through the unsorted half find the maximum element and swap it with last element of unsorted array and reduce its size by 1.

Selection Sort Animated



No swap needed as 4 is already at the last position of unsorted array

1	4	6	7	10

1	4	6	7	10

1	4	6	7	10