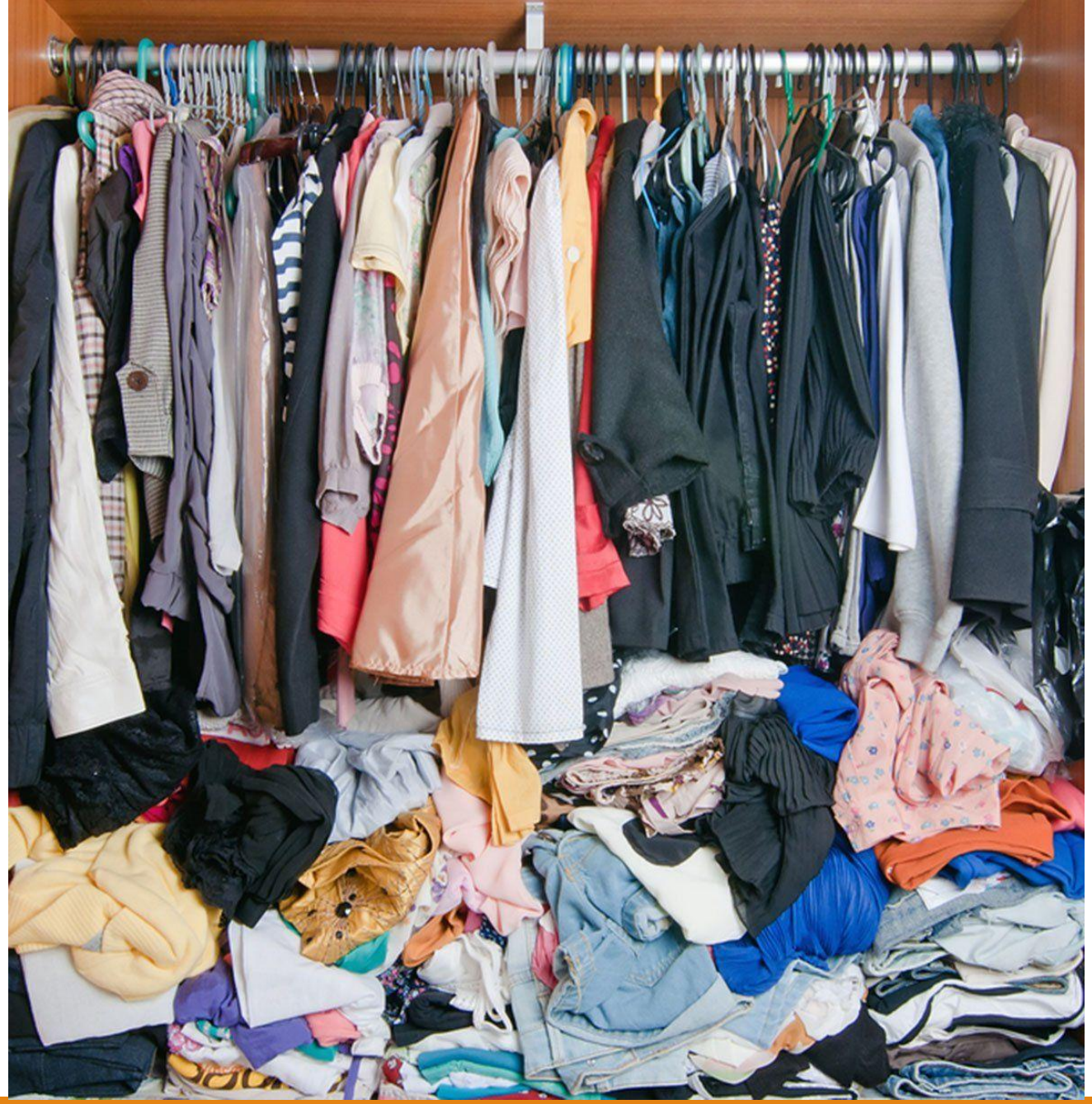
# Arrays

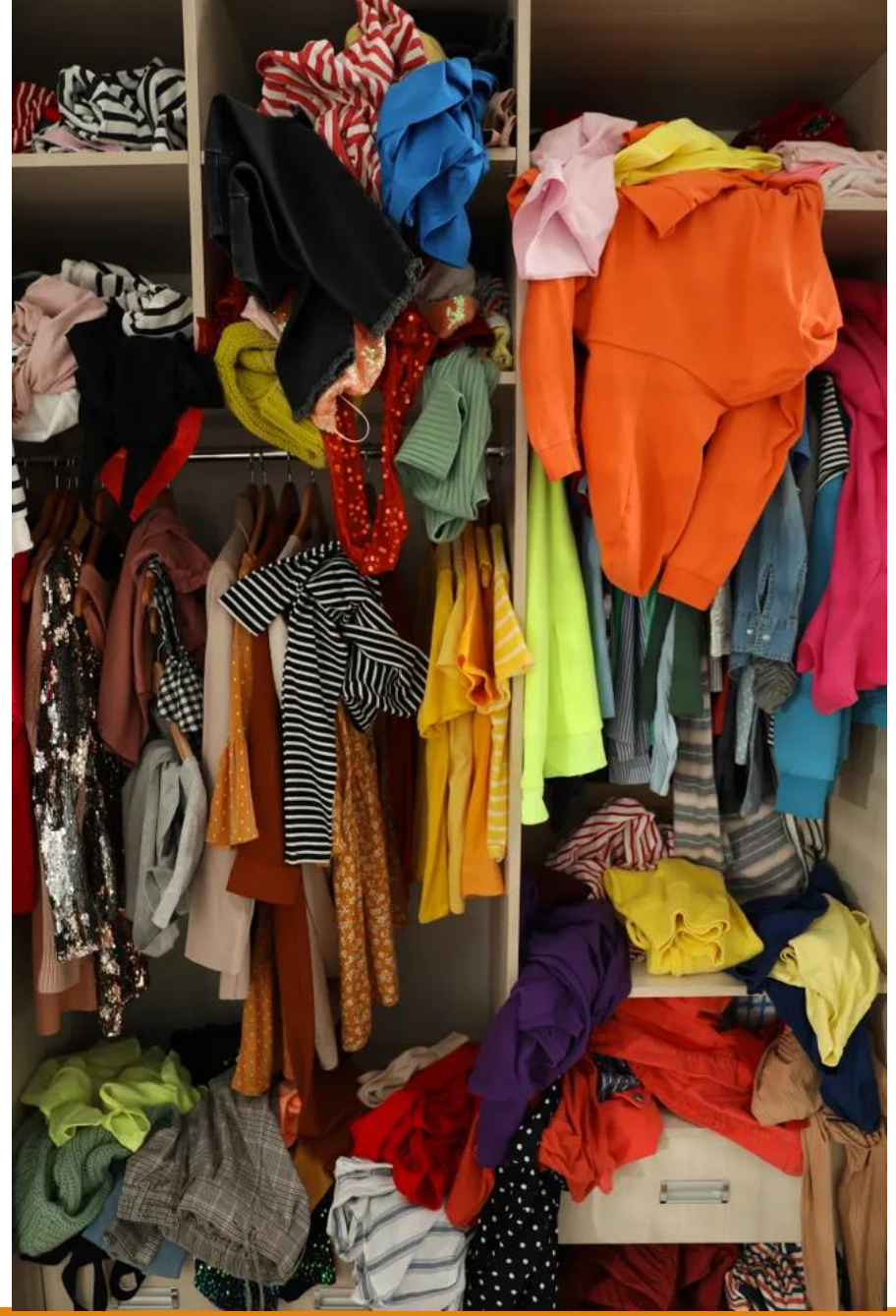
DR. SHWETA SHARMA

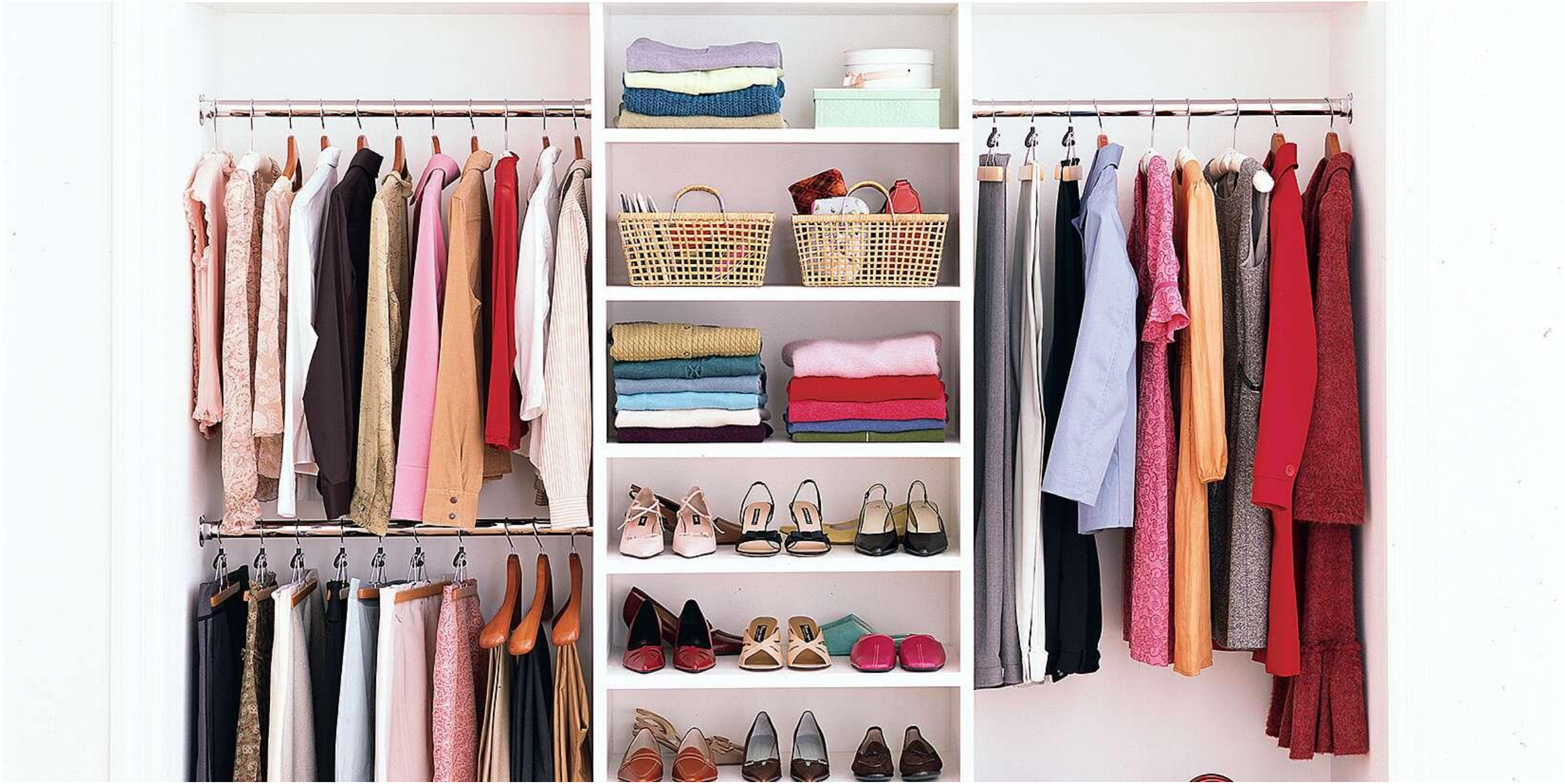
ASSISTANT PROFESSOR

DEPT. OF COMPUTER ENGINEERING

NIT KURUKSHETRA







# Data Structures

- A data structure is a storage that is used to store and organize data
- It is a way of arranging data on a computer so that it can be accessed and updated efficiently
- It is also used for processing, retrieving, and storing data

# Data Structures

- When designing code, we need to pay particular attention to the way data is structured
- If data isn't stored efficiently or correctly structured, then the overall performance of the code will be reduced

# Data Structures

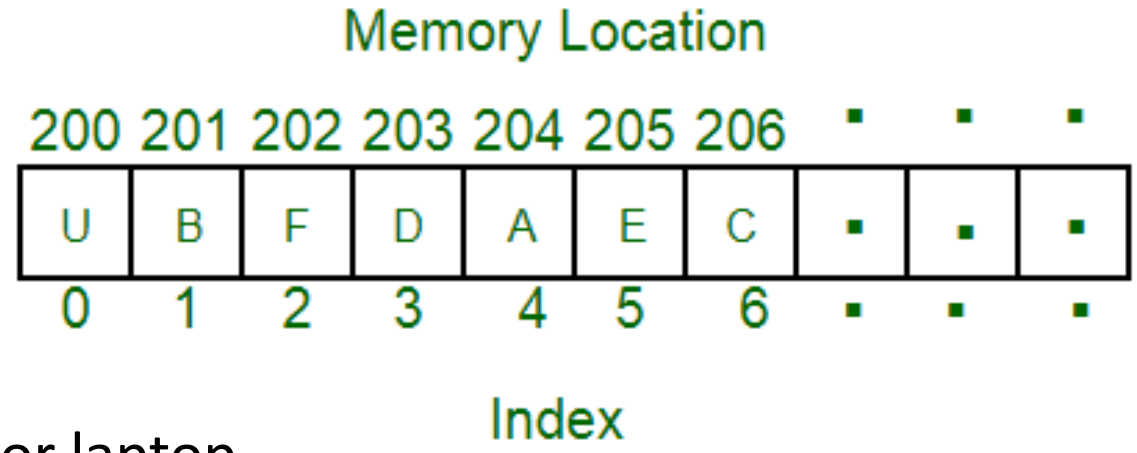
- Data structures make it easy for users to access and work with the data they need in appropriate ways
- Most importantly, data structures frame the organization of information so that machines and humans can better understand it

# Array Data Structure

- In an array, elements in memory are arranged in contiguous memory
- All the elements of an array are of the same type
- And, the type of elements that can be stored in the form of arrays is determined by the programming language

# Examples

- Online ticket booking
- Contacts on a cell phone
- To store images of a specific size on phone or laptop



# Arrays

- An array is a collection of data that holds fixed number of values of same type

- Size and type of arrays cannot be changed after its declaration

**For Example:**

- If you want to store marks of 100 students you can create an array for it.
- **float marks[100];**

# Declaration of an Array in C

## Syntax:

**data_type array_name [array_ size];**

**For Example:**

```
float marks[5];
```

# Elements of an array and How to access them?

You can access elements of an array by indices.

```
float marks[5];
```

Suppose you declared an array **marks** as above.

The first element is **marks[0]**, second element is **marks[1]** and so on

## Few Key points:

- Arrays have 0 as the first index not 1. In this example, marks[0]
- If the size of an array is n, to access the last element, **(n-1)** index is used. In this example, marks[4]
- Suppose the starting address of marks[0] is 2120d. Then, the next address, marks[1], will be 2124d, address of marks[2] will be 2128d and so on. Its because the size of a float is 4 bytes

# Initialize an array

Its possible to initialize an array during declaration

**For example:**

```
int mark[5] ={9,4,6,3,5};
```

Another method of initialize array during declaration

```
int mark[ ] ={9,4,6,3,5};
```

**Here,**

mark[0] is equal to 9
mark[1] is equal to 4
mark[2] is equal to 6
mark[3] is equal to 3
mark[4] is equal to 5

# Important thing to remember when working with C arrays:

Suppose you declared an array of 10 elements. Lets say,

```
int testArray[10];
```

You can use the array members from **testArray[0] to testArray[9].**

If you try to access array elements outside of its bound, lets say testArray[12], the compiler may not show any error
However, this may cause unexpected output (undefined behavior)

## Arrays are of three types:

1. One-dimensional arrays.
2. Two-dimensional arrays.
3. Multidimensional arrays.

# One-dimensional Array

A list of item can be given one variable name using only one subscript and such a variable is called a single subscripted variable or one dimensional array

## The Syntax for an array declaration is:

type variable-name[size];

## Example:

```
float height[50];
int groupt[10];
char name[10];
```

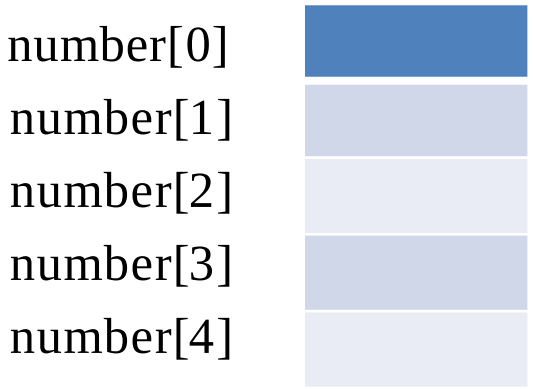
The type specifies the type of the element that will be contained in the array, such as **int, float, or char** and the size indicates the maximum number of elements that can be stored inside the array

Now as we declare an array

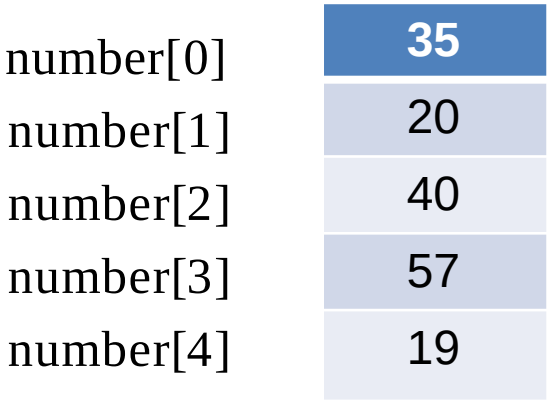
**int number[5];**

Then the computer reserves five storage locations as the size of the array is 5:

**Reserved Space**




**Storing values after Initialization**



# Initialization of one dimensional array:

After an array is declared, its elements must be initialized

In C programming an array can be initialized at either of the following stages:

**At compile time**

**At run time**

# Compile Time Initialization

The general form of initialization of array is:

**type array-name[size] ={list of values};**

The values in the list are separated by commas

**For example:**

**int number[3] ={0,5,4};**

- ✓ It will declare the variable ‘number’ as an array of size 3 and will assign the values to each elements
- ✓ If the number of values in the list is less than the number of elements then only that many elements will be initialized. The remaining will be set to zero automatically
- ✓ Remember, if we have more initializers than the size, the compiler will produce an error

# Run Time Initialization

**An array can also be explicitly initialized at run time**

For example, consider the following segment of a C program

```
for(i=0;i<10;i++)
{
    scanf("%d", &x[i]);
}
```

In the run time initialization of the arrays looping statements are almost compulsory

Looping statements are used to initialize the values of the arrays one by one using assignment operator or through the keyboard by the user

# Question 1

**Write a C program which will ask users to enter 5 numbers and store them in an array.**

# Solution 1

```
#include<stdio.h>
void main()
{
int array[5];
int i;
printf("Enter 5 numbers to store them in array \n");
for(i=0;i<5;i++)
{
scanf("%d", &array[i]);
}
printf("Element in the array are: \n");
for(i=0;i<5;i++)
{
printf("Element stored at a[%d]=%d \n", i, array[i]);
}
}
```

# Output

```
Enter 5 numbers to store them in array
10
20
30
40
50
Element in the array are:
Element stored at a[0]=10
Element stored at a[1]=20
Element stored at a[2]=30
Element stored at a[3]=40
Element stored at a[4]=50
```

# Question 2

**Write a C program using one-dimensional array to find mean of n numbers.**

# Solution 2

```
#include <stdio.h>
int main()
{
int n, i;
float sum = 0, mean;
printf("Enter the number of elements: ");
scanf("%d", &n);
float a[n];
printf("Enter %d numbers:\n", n);
for (i = 0; i < n; i++) {
   scanf("%f", &a[i]);
   sum += a[i];
}
mean = sum / n;
printf("The mean of the given numbers is: %.2f\n", mean);
return 0;
}
```

# Output

```
Enter the number of elements: 5
Enter 5 numbers:
10
20
30
40
50
The mean of the given numbers is: 30.00
```

# Thank You!