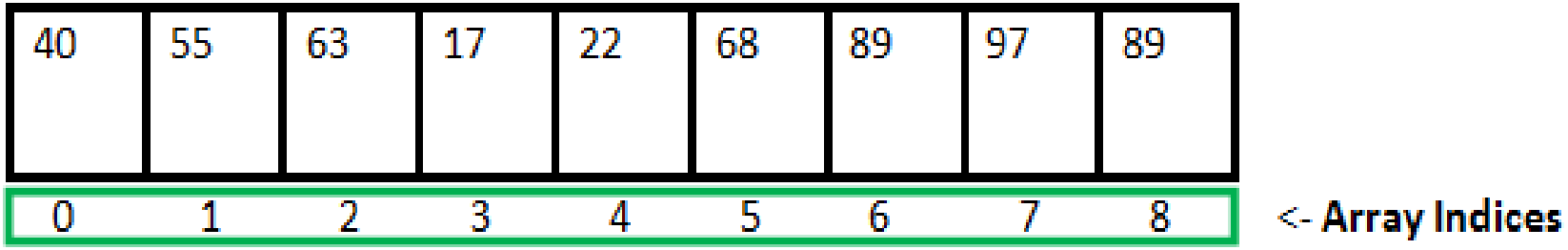
Arrays in C

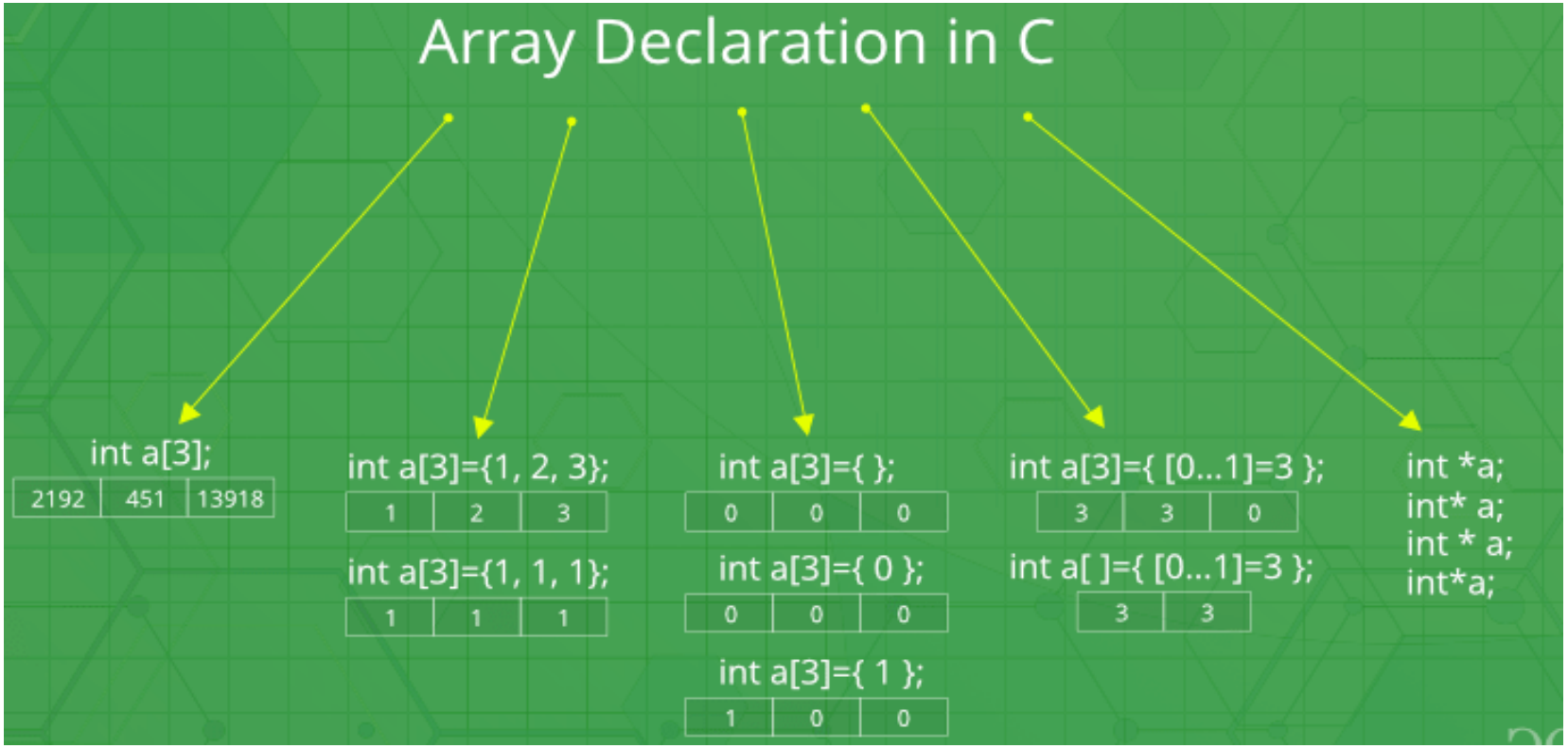
- An array is defined as the **collection of similar type** of data items stored at **contiguous memory locations**.
- Arrays are the **derived data type** in C programming language which can **store the primitive type of data** such as int, char, double, float, etc.
- The array is the simplest data structure where each data element can be **randomly accessed** by using its **index number**.



Array Length = 9
First Index = 0
Last Index = 8

Why do we need arrays?

- Use normal variables (v1, v2, v3, ..) when we have small number of objects, but if we want to store large number of instances, it becomes difficult to manage them with normal variables.
- The idea of array is to represent many instances in one variable



Properties of Array

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Advantage of C Array

- **Code Optimization**: Less code to the access the data.
- **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.
- **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.
- **Random Access**: We can access any element randomly using the array.

Disadvantage of C Array

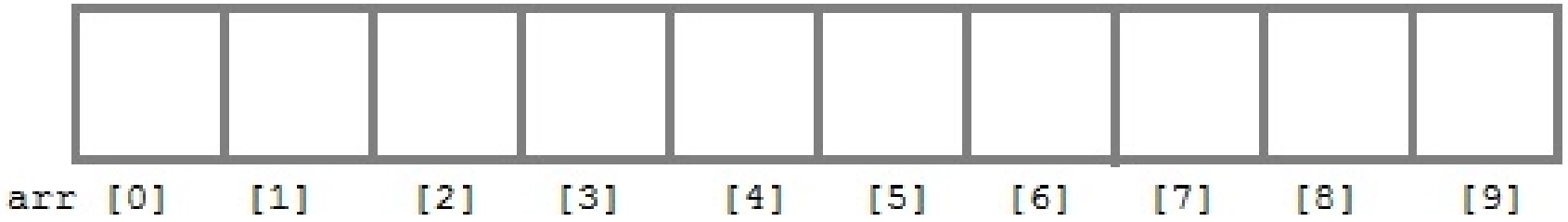
- **Fixed Size**: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList.

Declaring an Array

Syntax ⟶ data_type array_name[array_size];

Example

int arr[10];



- Here int is the data type, arr is the name of the array and 10 is the size of array. It means array arr can only contain 10 elements of int type.
- **Index** of an array starts from 0 to **size-1** i.e first element of arr array will be stored at arr[0] address and the last element will occupy arr[9].

Initialization of an Array

- ✓ After an array is declared it must be initialized. Otherwise, it will contain **garbage** value(any random value).
- ✓ An array can be initialized at either **compile time** or at **runtime**.

Compile time Array initialization

- Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,
- **data-type array-name[size] = { list of values };**

Examples

- int marks[4]={ 67, 87, 56, 77 }; // integer array initialization
- float area[5]={ 23.4, 6.8, 5.5 }; // float array initialization
- int marks[4]={ 67, 87, 56, 77, 59 }; // Compile time error

Example

```
#include<stdio.h>
void main()
{
        int i;
        int arr[] = {2, 3, 4}; // Compile time array initialization
        for(i = 0 ; i < 3 ; i++)
        {
                printf("%d\t",arr[i]);
        }
}
```

Output: 2 3 4

Runtime Array initialization

- ✓ An array can also be initialized at runtime using scanf() function.
- ✓ This approach is usually used for initializing large arrays, or to initialize arrays with user specified values.

Examples

- ✓ int A[10]; // An array of ten integers; A[0], A[1], …, A[9]
- ✓ char name[20]; // An array of 20 characters
- ✓ float nums[50]; // An array of fifty floating numbers; nums[0], nums[1], …,nums[49]
- ✓ int C[ ]; // An array of an unknown number of integers; C[0], C[1], …, C[size-1]

Example 1

```
#include<stdio.h>
void main()
{
        int arr[4];
        int i;
        printf("Enter array element");
        for(i = 0; i < 4; i++)
        {
                scanf("%d", &arr[i]); //Run time array initialization
        }
        for(i = 0; i < 4; i++)
        {
                printf("%d\n", arr[i]);
        }
}
```

Example 2

```
#include<stdio.h>
void main()
{
        int i,n;
        printf("Enter the array size: ");
        scanf("%d", &n);
        int arr[n];
        printf("Enter array element");
        for(i = 0; i < n; i++)
        {
                scanf("%d", &arr[i]);    //Run time array initialization
        }
        for(i = 0; i < n; i++)
        {
                printf("%d\n", arr[i]);
        }.
}
```

Two dimensional Arrays

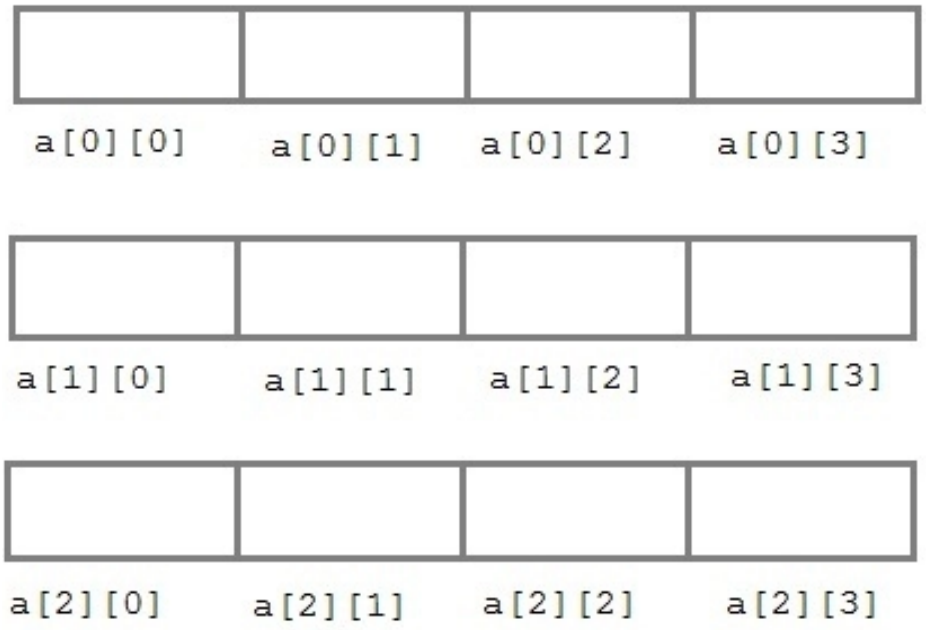
- C language supports multidimensional arrays also.
- The simplest form of a multidimensional array is the two-dimensional array.
- Both the row's and column's index begins from 0.

Two-dimensional arrays are declared as follows,

- **data-type array-name[row-size][column-size];**

/* Example */

- int a[3][4];



Two dimensional Arrays

- An array can also be declared and initialized together.

For example,

- **int arr[][3] = { {0,0,0}, {1,1,1} };**

Note:

- ✓ We have not assigned any row value to our array in the above example. It means we can initialize any number of rows.
- ✓ But, we must always specify number of columns, else it will give a compile time error.
- ✓ Here, a 2*3 multi-dimensional matrix is created.

Compile time initialization of a two dimensional Array

```
#include<stdio.h>
void main()
{
        int i=0,j=0;
        int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
        //traversing 2D array
        for(i=0;i<4;i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
                }//end of j
        }//end of i
    }
```

Compile time initialization of a two dimensional Array

Output:

arr[0][0] = 1

arr[0][1] = 2

arr[0][2] = 3

arr[1][0] = 2

arr[1][1] = 3

arr[1][2] = 4

arr[2][0] = 3

arr[2][1] = 4

arr[2][2] = 5

arr[3][0] = 4

arr[3][1] = 5

arr[3][2] = 6

Runtime initialization of a two dimensional Array

```
#include <stdio.h>
void main ()
{
        int arr[3][3],i,j;
        for (i=0;i<3;i++)
        {
                for (j=0;j<3;j++)
                {
                        printf("Enter a[%d][%d]: ",i,j);
                        scanf("%d",&arr[i][j]);
                }
        }
```

Runtime initialization of a two dimensional Array

```
        printf("\n printing the elements ....\n");
        for(i=0;i<3;i++)
        {
                printf("\n");
                for (j=0;j<3;j++)
                {
                printf("%d\t",arr[i][j]);
                }
        }
}
```

Runtime initialization of a two dimensional Array

Output:

```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34
Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78
printing the elements ....
56 10 30
34 21 34
45 56 78
```