Array based Linear Data Structures

Linear Data Structure

- data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.
- traverses the data elements sequentially.
- only one data element can directly be reached.
- memory is not utilized in an efficient way.
- includes array, linked list, stack and queues.

Types of Arrays

- One Dimensional
- Two Dimensional
- Multi-Dimensional

One Dimensional Arrays

- It is a collection of items of same data type stored at contiguous memory locations.
- we can also store “n” num of variables in same variable Eg: int a[50];
- Elements of the array can be randomly accessed.

Initialization of 1D-Arrays

1. **Passing no value within the initializer:** One can initialize the array by defining the size of the array and passing no values within the initializer.

Eg: int arr[5] = { };

2. By passing specific values within the initializer: One can initialize the array by defining the size of the array and passing specific values within the initializer.

Eg: int arr[5] = { 1 , 2 , 3 , 4 , 5 };

The index of elements within the "{ }", must be less than the size of the array

If the count of elements within the "{ }" is less than the size of the array, the remaining positions are considered to be '0'.

Eg: int arr[5] = { 1 , 2 , 3 } ;

3. By passing specific values within the initializer but not declaring the size:

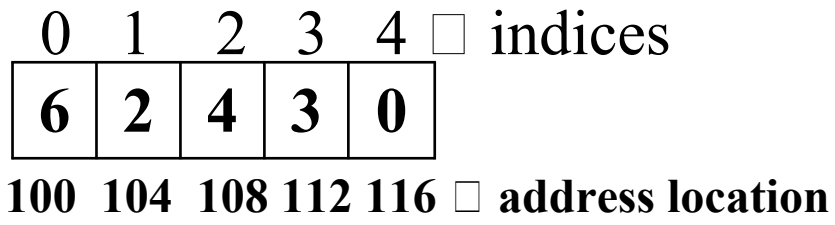
One can initialize the array by passing specific values within the initializer and not particularly mentioning the size, the size is interpreted by the compiler

Eg: int arr[ ] = { 1 , 2 , 3 , 4 , 5 }; static time allocation

int a[5]; run time allocation

Accessing in Arrays

- In arrays it is easier to calculate the position of each element by simply adding an **offset** to a base value, i.e., the memory location of the first element of the array
- Eg:



TYPE 1:

Address of A[i] = B+i*(size of data type) indices starts with 0

Example 1:

Find a[2] of the int data type of base address 100

100+2*4

100+8

108 (m/y location)

where B is the base address and i is the position of values in an array

B+(i-1)*(size of data type) indices starts with 1

TYPE 2:

Address of A[I] = B + W * (I – LB)

I = Subset/index of elements whose address is to be found,

B = Base address,

W = Storage size of one element stored in an array(in byte),

LB = Lower Limit/Lower Bound of subscript(If not specified assume zero).

Example 2:

Given the base address of an array **A[1300 …… 1900]** as **1020** and the size of each element is 2 bytes in the memory, find the address of **A[1700].**