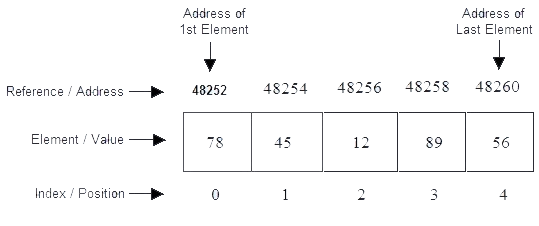
**Lecture 8**

**ARRAY**

* **Array** : is a collection of similar data type. A single variable can hold only one value at a time, If we want a variable to store more than one value of same type we use array.  C++ provides a data structure, **the array**, which stores a fixed-size sequential collection of elements of the same type.
* Instead of declaring individual variables, such as num0, num1, ..., and num99, you declare one array variable such as **ns** and use:

ns[0], ns[1], and ..., ns[99] to represent individual variables.

* A specific element in an array is accessed by an **index**. All arrays consist of adjacent memory locations (مواقع متجاورة في الذاكرة).
* The lowest address corresponds to the first element and the highest address to the last element.



* **Address** of first element is random, address of next element depend upon the type of array. Here, the type is integer and integer takes two bytes in memory, therefore every next address will increment by two.
* **Index** of array will always starts with zero.

**Declaration of Arrays:** Declaration of array means creating sequential blocks of memory to hold fixed number of values.

**Syntax of array declaration :**

**Data-type** **Array-name** [ **size of Array** ] **;**

**Example of array declaration :**

**int Array [5]; //Statement 1**

**float Array [10]; //Statement 2**

**char[50]; //Statement 3**

In the above example, statement 1 will allocate memory for an integer array which will hold five values and statement 2 will allocate memory for float point array which will hold ten values. statement 3 will allocate memory for an char array which will hold fifteen values.

**Initialization of Array**

Initialization means assigning value to the declared Array. You can initialize C++ array elements either **one by one** or using a **single statement** as follows:

float A1 [5 ] = { 78.1, 45.22, 0.5, 89.14, 56.7 };

In the above example we are declaring and initializing an array at same time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1[0] A1[1] A1[2] A1[3] A1[4] | | | | |
| 78.1 | 45.22 | 0.5 | 89.14 | 56.7 |

If we assign another values to some elements in A1 array such that:

A1[2]=33.50;

A1[4] = 22.4; // assigns element number 5th in the array a **value of 22**.

Then the array A1 will be :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1[0] A1[1] A1[2] A1[3] A1[4] | | | | |
| 78.1 | 45.22 | 33.50 | 89.14 | 22.4 |

**Few key notes:**

* Arrays have 0 as the first index not 1. In this example, A1[0] is the first element.
* If the size of an array is n, to access the last element, (n-1) index is used. In this example, A1[4] is the last element.
* Suppose the starting address of A1[0] is 2120. Then, the next address of A1[1], will be 2124, and address of A1[2] will be 2128 and so on. It's because the size of a float is 4 bytes.

**Example:**

#include<iostream.h>

void main()

{

int array [5]; // declaration of array

int i;

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

{

cout << "\nEnter any number : ";

cin >> array [i]; //Input array from user.

}

for(i=0;i<5;i++) //Output array to screen.

cout << array [i];

}

Output :

Enter any number : 78

Enter any number : 45

Enter any number : 12

Enter any number : 89

Enter any number : 56

78, 45, 12, 89, 56, **Pointer and Array**

**Question:** //--- Print array in reverse order