Array

Array:

- An array is an aggregate data type or derived data type.
- An array is a collection of data of the same type by a common name i.e. array is the collection of homogeneous data type which contains all the similar elements.
- Using single array variable we can store n number of elements but of **same data types**.
- If it contains int data type, all the data of the array must be of integer, if it contains float data type, then all the data must be of float.
- A specific element of an array is accessed by an **index number.**

Types of an Array:

1. One-Dimensional Array (1-D array)
2. Multi-Dimensional Array-Two-Dimensional Array (2-D array)

1. One-Dimensional Array:

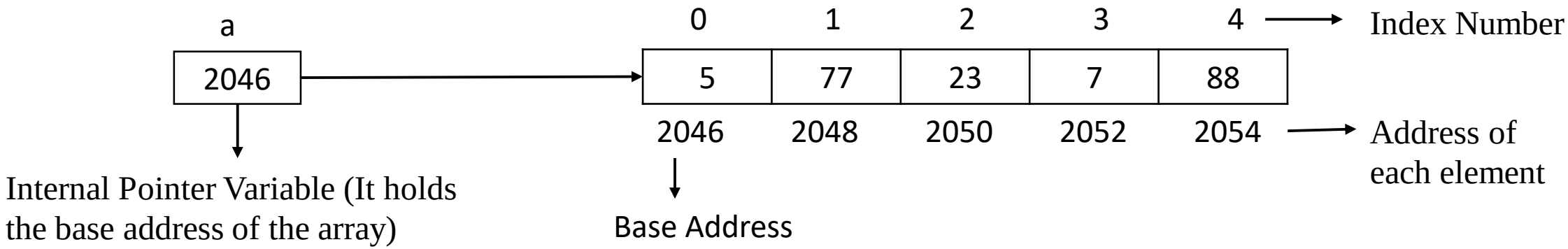
Declaration of 1-D array:

Syntax: datatype name-of-array-variable[size];

Example: int a[5];

-We can process an array element with the help of index number only.

-Index number always starts from 0 to size-1.



Size of an array:

- The amount of storage required to hold an array is directly related to its type and size.

Total size of array in bytes= size of (base type) * length of array

- The address of starting element of an array is called **base address** of an array.

Example: int a[5];

Total size of an array variable(a) =2*5=10 bytes.

Other way to declare an array:

1. Syntax: datatype array_name[size];

   Example: int num[5];

   char letter[40];

2. Syntax: datatype array_name[size]= {list of elements};

   Example: int num[5]={2,5,7,8,99};

3. Syntax: datatype array_name[ ]= {list of elements};

   Example: int num[ ]={44,3,78,9,23};

Initialization of an Array:

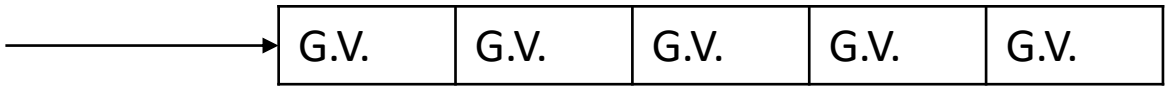
- Each array element can be initialized when an array is declared.
- The initial values must appear in the order in which they will be assigned to the individual array elements, enclosed in curly braces { } and separated by commas.

For Example:

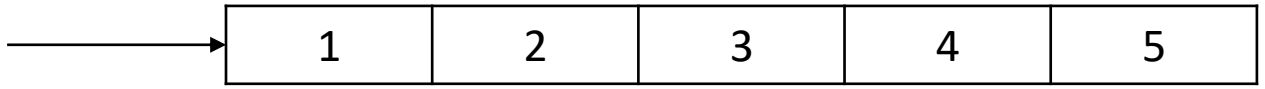
```
int n[10] = {4, 6, 88, 8, 99, 34, 2, 1, 44, 100};
float x[4] = { 4.5, 77.2, -8.55, 3.99};
char ch[3] = {‘R’, ‘E’, ‘D’};
```

Local Declaration:

-Local declaration means inside the method(function).

int n[5]; 

All the locations are initialize with the garbage value.

int n[5]={1, 2, 3, 4, 5}; 

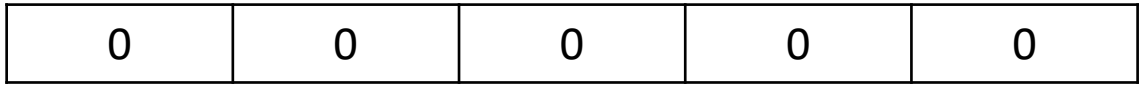
int n[5] = {5, 20}; 

If we assign at least one value, then rest of the value will be zero.

Global Declaration:

- Global declaration of variable means declaration of variable outside to all the functions in the program.

int a[5]; ⟶

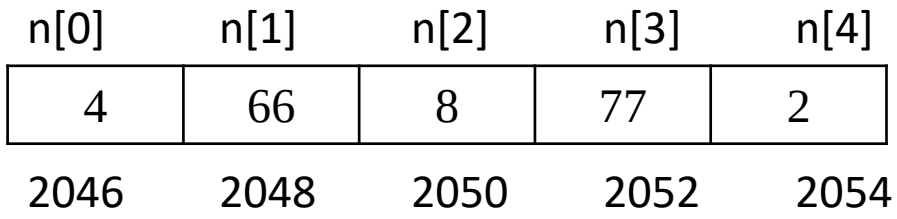


For global variable default integer value is zero.

Processing or Accessing Array Element:

- Each array element is accessed by specifying an array name followed by subscript/index [enclosed in square bracket].

int n[5]={4, 66, 8, 77,2};



For Reading Purpose:

```
for(i=0;i<5;i++)
{
        scanf("%d", &n[i]);
        }
```

For Printing Purpose:

```
for(i=0;i<5;i++)
{
        printf("%d", n[i]);
        }
```

In case of scanf we need to provide address of each location but in case of printf directly we are providing the location.

Searching in an Array:

- Searching in an array is the process of finding an element with in the list of an array element.
- A search algorithm is an algorithm that accepts an argument or a search key and tries to find a record in a given list of array element.

Searching Algorithms:

1. Linear or Sequential Search
2. Binary Search

1. Sequential or Linear Search:

A linear search, also known as a sequential search, is a method of finding an element within a list. It checks each element of the list sequentially until a match is found or the whole list has been searched.

2. Binary Search Algorithm:

- Binary search algorithm is an extremely efficient search technique.
- The array must be ordered (sorted) to apply the binary search.

The working mechanism of binary search algorithms is as follows:

1. If the array or list contains only one element, the problem is trivial. Otherwise,
2. Find the middle element of a sorted array and compare the item being searched with the item at the middle element of the array.
3. It they are equal then search is successful.
4. If the middle element is greater than the item being searched for, the search process is repeated in the first half of the array.
5. If the middle element is less than the item being searched for, the search process is repeated in the second half of the array.

Two-Dimensional Array (2-D Array):

- An array of arrays is known as 2D array.
- The two dimensional (2D) array in C programming is also known as matrix.
- A matrix can be represented as a table of **rows and columns.**

Declaration of 2-D Array:

data_type array_name[size1][size2];

Where size1 is the size of rows and size2 is the size of columns.

Example: int a[2][3];

Total number of elements=size of rows * size of columns

Advantages of Array:

- It is used to represent multiple data items of the same type by using the only single name.
- We can easily access each element of the array.
- Not to declare too many variables.
- Array elements are stored in a continuous memory location.
- 2D arrays are used to represent matrices.

Disadvantages of Array:

- We can not change the size of array at the run time.
- It can store only similar data type.
- Since array is of fixed size, if we allocate more memory than requirements, then the memory space will be wasted. And if we allocate less memory than requirement, then it will create problem.
- C environment does not have checking mechanism for array sizes.

String

String:

- A string is a sequence of characters terminated with a null character (‘\0’).
- Null character (‘\0’) is a special character used in the string to represent the end of the string appended by the compiler.
- String can be represented by character array.

Declaration and Initialization of string:

Declaration of string:

Syntax: char name_of_string[size];

Example: char str[10];

Initialization of String:

1. char c[] = "abcd";
2. char c[50] = "abcd";
3. char c[] = {'a', 'b', 'c', 'd', '\0'};
4. char c[5] = {'a', 'b', 'c', 'd', '\0'};

Reading Strings:

We use format specifier **%s** to read string and it reads characters until it encounters a white space character.

Simple Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        char str[20];
        printf("Enter a string: ");
        scanf("%s",str);
        printf("Your entered string is %s",str);
        getch();
        return 0;
}
```

gets() and puts():

- gets( ) is used to scan or read a line of text from a standard input (stdin) device and store it in the String variable. When it reads the newline character, then the C gets function will terminate.
- puts() used to print the string read by the gets function.

Example of gets() and puts():

```
#include<stdio.h>
#include<conio.h>
int main()
{
        char str[20];
        printf("Enter a string: ");
        gets(str);
        puts(str);
        getch();
        return 0;
}
```

String Functions:

- There are various **string library functions** defined under the header file **string.h**.
- String functions are used to manipulate strings.

1. strlen ( ):

This function is used to returns the number of characters as an output, excluding the terminating null character.

Syntax: strlen(string);

2. strcat ( ):

This function is used to combines two strings together to form a whole string.

Syntax: strcat(destination_string, source_string);

3. strcmp ( ):

This function is used to compares two strings such that if two strings are equal then it returns 0 else if string1 is greater than string two it returns 1 and if string1 is less than string2 it returns -1.

Syntax: strcmp(string1,string2);

4. strrev ( ):

This function is used to reverse the given string.

Syntax: strrev(string);

5. strcpy ( ):

This function is used to copies source string to destination string.

Syntax: strcpy(destination_string, source_string);

6. strlwr ( ):

This function is used to changes uppercase strings to lowercase.

Syntax: strlwr(string);

7. strupr ( ):

This function is used to changes lowercase strings to uppercase.

Syntax: strupr(string);

Examples:

- Program to count the total number of character present in the string.
- Program to reverse the given input string.
- Program to check the given string is palindrome or not.
- Program to count the total number of vowels containing in the string.
- Program to count the total number of words containing in the string.
- Program to count the total number of alphabets, digits and other symbol in the given input string.