Unit - 5

Types of Function & Introduction to Arrays

Array

- An array is a collection of elements of the same type that are referenced by a common name.
- Compared to the basic data type (int, float & char) it is a derived data type.
- Why need to use array type?

  Consider the following issue:

- **We have a list of 1000 students' marks of an integer type. If using the basic data type (int), we will declare something like the following**

Array

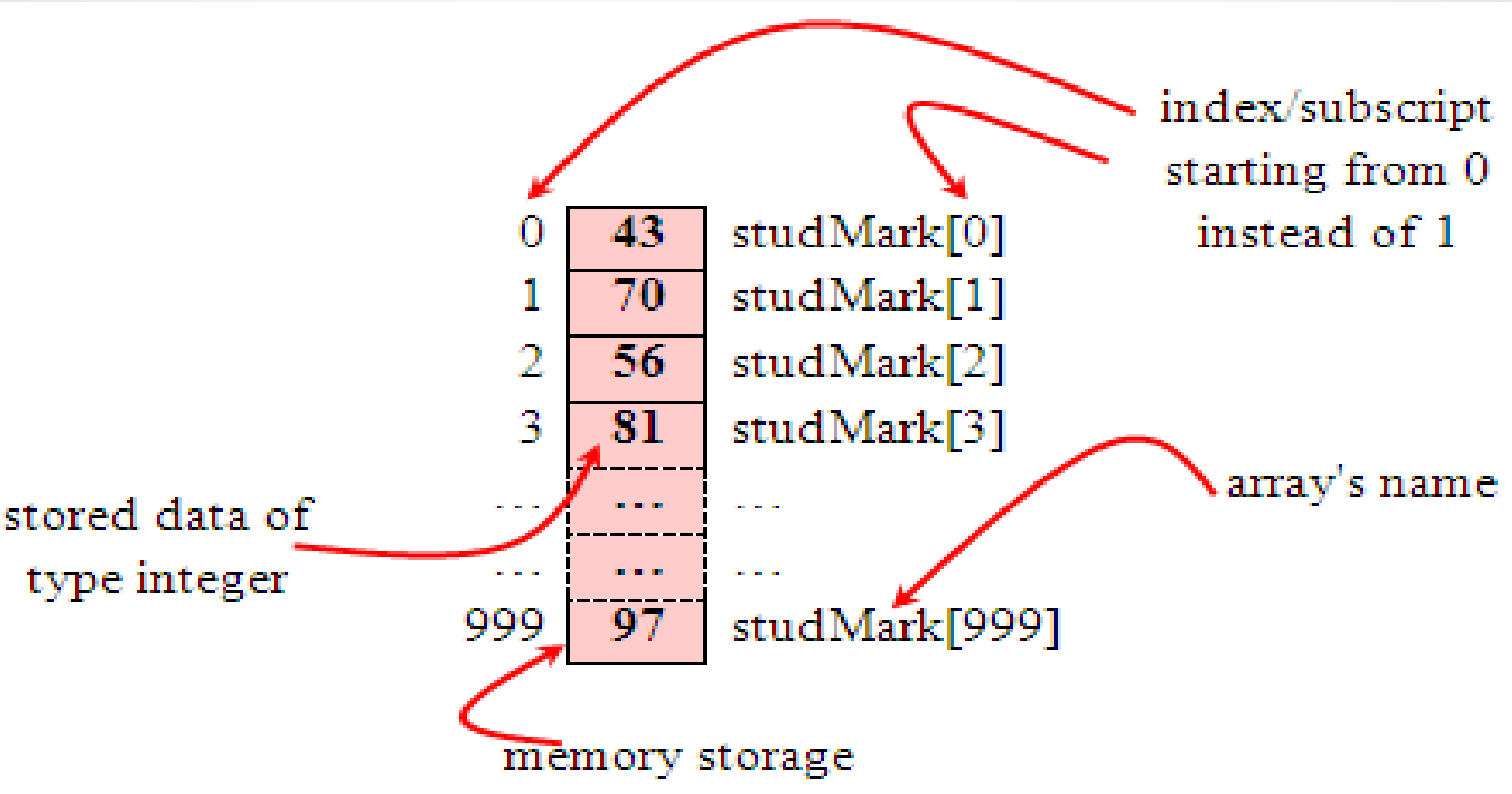
```
int main(void)
{
    int studMark1, studMark2, studMark3, studMark4, …,
    …, studMark998, stuMark999, studMark1000;
    …
    return 0;
}
```

- By using an array, we just declare like this,

```
int  studMark[1000];
```

- This will reserve 1000 contiguous memory locations for storing the students' marks.

Array


index/subscript
starting from 0
instead of 1
0
1
2
3
...
...
999
43
70
56
81
...
...
97
studMark[0]
studMark[1]
studMark[2]
studMark[3]
...
...
studMark[999]
stored data of
type integer
array's name
memory storage

Array

- We can use index or subscript to identify each element or location in the memory.
- For example, studMark[0] will refer to the first element of the array.
- Thus by changing the value of index, we could refer to any element in the array.
- So, array has simplified our declaration and manipulation of the data.

Array Dimension

- Dimension refers to the array's size, which is how big the array is.
- A single or one dimensional array declaration has the following form,

Syntax : array_element_data_type array_name[array_size];

- Here, *array_element_data_type* define the base type of the array, which is the type of each element in the array.
- *array_name* is any valid C identifier name that obeys the same rule for the identifier naming.
- *array_size* defines how many elements the array will hold.

Array Dimension

- For example, to declare an array of 30 characters, that construct a people name, we could declare,

```
char   cName[30];
```

- In this statement, the array character can store up to 30 characters with the first character occupying location cName[0] and the last character occupying cName[29].
- Note that the index runs from 0 to 29. In C, an index always starts from 0 and ends with array's (size-1).
- So, take note the difference between the array size and subscript/index terms.

Two Dimensional Array

- A two dimensional array has two subscripts/indexes. It is also called Multi dimensional array.
- **The first subscript refers to the row, and the second, to the column.**
- Its declaration has the following form,

  data_type array_name[1st dimension size][2nd dimension size];
- For example,

  int number[3][4];

  float number_1[20][25];
- The first line declares variable number as an integer array with 3 rows and 4 columns.
- Second line declares a variable number_1 as a floating-point array with 20 rows and 25 columns.

Two Dimensional Array

float x[3][4];

- Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as a table with 3 rows and each row has 4 columns.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Introduction to Strings

- Definition: A string is a sequence of characters terminated with a null character ('\0').
- Characteristics:
- - Represented as an array of characters.
- - Can include letters, numbers, symbols, etc.
- - Commonly used in programming for text manipulation.

Declaration and Initialization of Strings

- Declaration: char str[50]; (Declares a string of size 50)
- Initialization:
- - Method 1: char str[] = "Hello";
- - Method 2: char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
- Strings are always terminated with a null character ('\0').

Reading a String from the User

- Using scanf: scanf("%s", str); (Note: Does not read spaces)
- Using gets: gets(str); (Reads until a newline is encountered, but is unsafe and deprecated)

String Array

- Definition: An array of strings is a two-dimensional array of characters.
- Example: char names[3][20] = {"Alice", "Bob", "Charlie"};
- Usage: Used to store a list of strings (e.g., names, words).

String Library Functions

- strlen(str): Returns the length of the string str.
- strcmp(str1, str2): Compares two strings str1 and str2.
- strcpy(dest, src): Copies string src to dest.
- strcat(dest, src): Appends string src to the end of dest.
- strlwr(str): Converts all characters of str to lowercase.
- strupr(str): Converts all characters of str to uppercase.

String Functions

No.	Function	Description
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
5)	strrev(string)	returns reverse string.
6)	strlwr(string)	returns string characters in lowercase.
7)	strupr(string)	returns string characters in uppercase.