Problem Solving Through programming in C
Course Code: ONL1001

ARRAYS

Ms. SHUBHRA DWIVEDI
School - SCOPE
VIT-AP Amaravati

ARRAYS

- 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 an aggregate or derived data type.
- All the elements of an array occupy a set of contiguous memory locations.
- 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…"

```
int  studMark0, studMark1, studMark2, ..., studMark999;
```

ARRAYS

- Can you imagine how long we have to write the declaration part by using normal variable declaration?

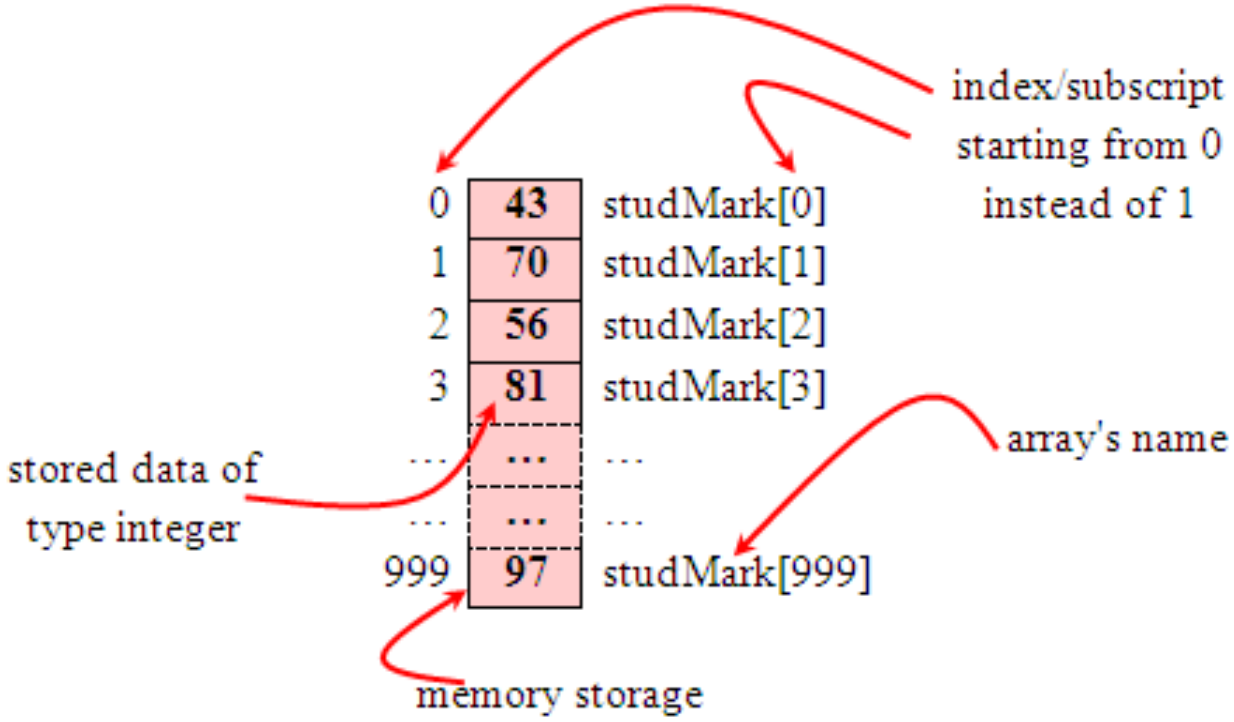
```
int main(void)
{
   int studMark1, studMark2, studMark3,
   studMark4, …, …, studMark998, stuMark999,
   studMark1000;
   …
   …
   return 0;
}
```

ARRAYS

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

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

- This will reserve 1000 contiguous memory locations for storing the students' marks.
- Graphically, this can be depicted as in the following figure.



ARRAYS

- This absolutely has simplified our declaration of the variables.
- We can use index or subscript to identify each element or location in the memory.
- Hence, if we have an index of `jIndex`, `studMark[jIndex]` would refer to the *`jIndexth`* element in the array of `studMark`.
- For example, `studMark[0]` will refer to the first element of the array.
- Thus by changing the value of `jIndex`, we could refer to any element in the array.
- So, array has simplified our declaration and of course, manipulation of the data.

ARRAYS

One Dimensional Array: Declaration

- 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,

```
array_element_data_type array_name[array_size];
```

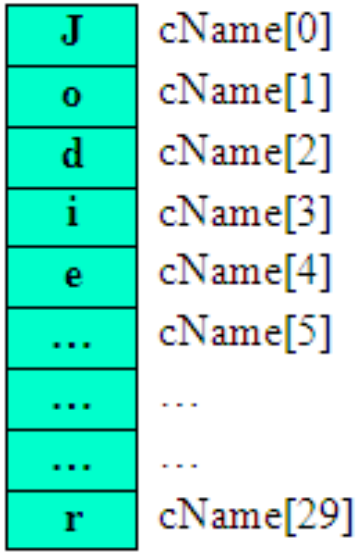
- Here, *array_element_data_type* define the base type of the array, which is the type of each element in the array.
- *array_name* is any valid C / C++ identifier name that obeys the same rule for the identifier naming.
- *array_size* defines how many elements the array will hold.

ARRAYS

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

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

- Which can be depicted as follows,



- 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.

ARRAYS

- Examples of the one-dimensional array declarations,

```
int       xNum[20], yNum[50];
float     fPrice[10];
char      chLetter[70];
```

- The first example declares two arrays named `xNum` and `yNum` of type `int`. Array `xNum` can store up to 20 integer numbers while `yNum` can store up to 50 numbers.
- The second line declares the array `fPrice` of type `float`. It can store up to 10 floating-point values.
- The third line declares the array `chLetter` of type char. It can store a string up to 69 characters.
- Why 69 instead of 70? Remember, a string has a null terminating character (`\0`) at the end, so we must reserve for it.

ARRAYS

Array Initialization

- An array may be initialized at the time of declaration.
- Giving initial values to an array.
- Initialization of an array may take the following form,

```
type    array_name[size] = {a_list_of_value};
```

- For example:

```
int     idNum[7] = {1, 2, 3, 4, 5, 6, 7};
float   fFloatNum[5] = {5.6, 5.7, 5.8, 5.9, 6.1};
char    chVowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};
```

- The first line declares an integer array `idNum` and it immediately assigns the values 1, 2, 3, ..., 7 to `idNum[0]`, `idNum[1]`, `idNum[2]`,...,`idNum[6]` respectively.
- The second line assigns the values 5.6 to `fFloatNum[0]`, 5.7 to `fFloatNum[1]`, and so on.
- Similarly the third line assigns the characters '`a`' to `chVowel[0]`, '`e`' to `chVowel[1]`, and so on. Note again, for characters we must use the single apostrophe/quote (') to enclose them.
- Also, the last character in `chVowel` is `NULL` character ('`\0`').

String and Character Array

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A string is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

For example: The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.

Declaration of strings: Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.

char str_name[size];

In the above syntax str_name is any name given to the string variable and size is used define the length of the string, i.e the number of characters strings will store. Please keep in mind that there is an extra terminating character which is the Null character (‘\0’) used to indicate termination of string which differs strings from normal character arrays.

Initializing a String: A string can be initialized in different ways. We will explain this with the help of an example. The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
If you follow the rule of array initialization then you can write the above statement as follows −

char greeting[] = "Hello";
Following is the memory presentation of the above defined string in C/C++ −