Chapter – 7
Array, String and Pointers

Course Code: CIS 122 & 122 L

Course Title: Structured Programming

Course Leader: ABK Bhuiyan (Jehad)

Concept

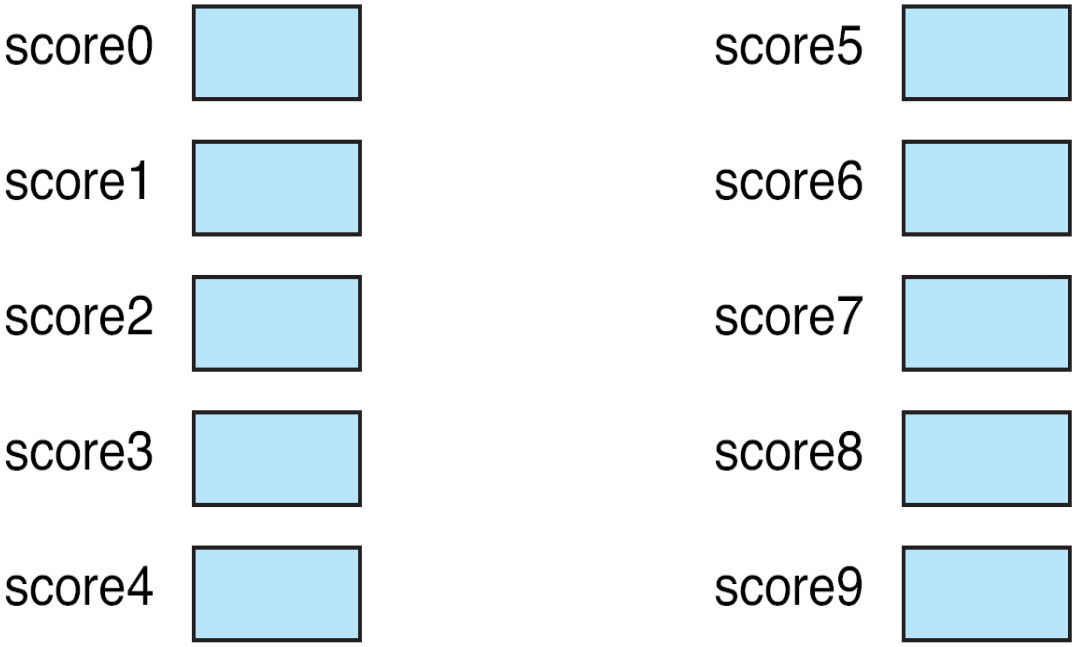
- Imagine we have a problem that requires us to read, process, and print a large number of integers. We must also keep the integers in memory for the duration of the program.

Concept

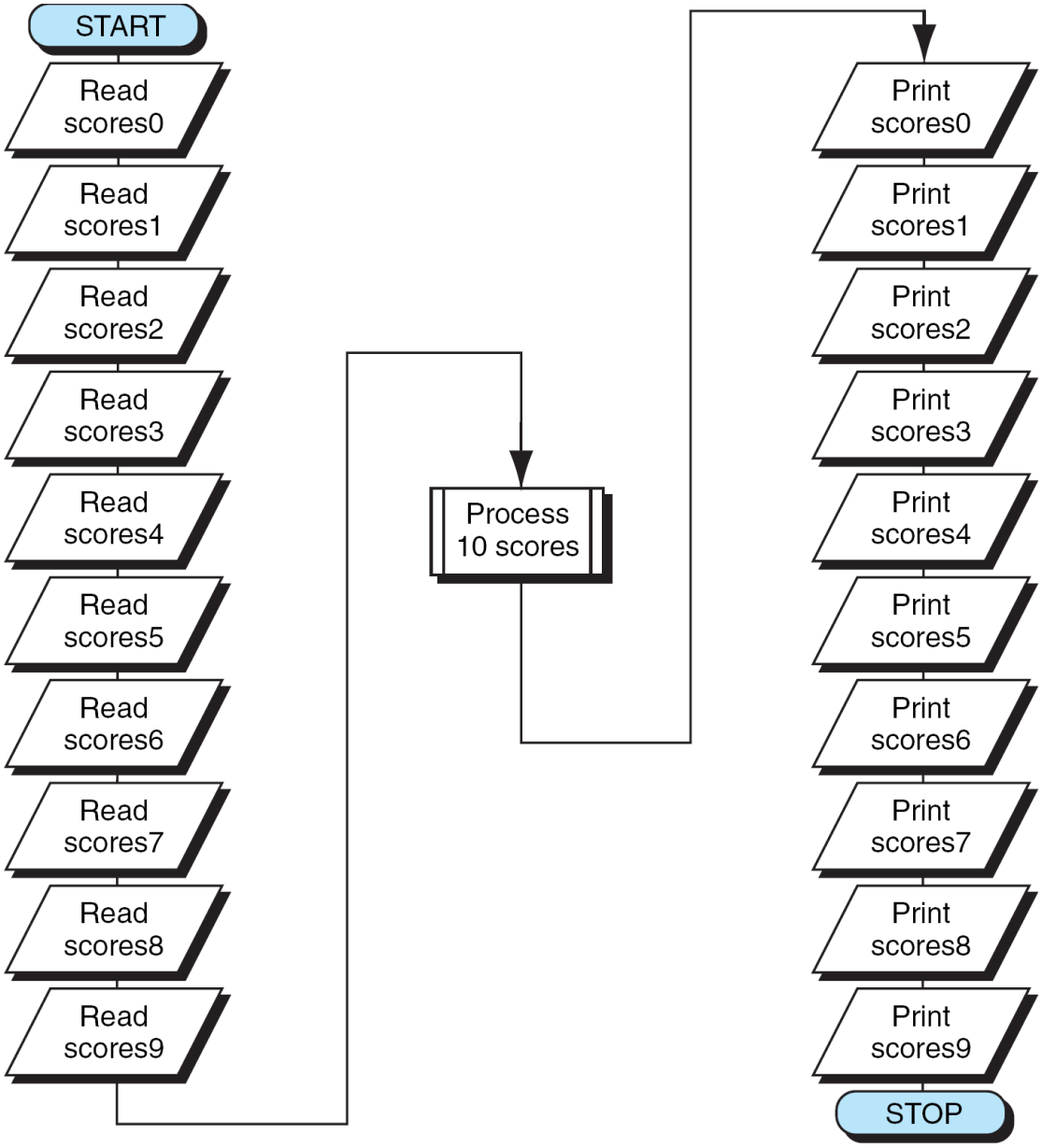
- To process large amounts of data we need a powerful data structure, the array. An array is a collection of elements of the same data type.
- Since an array is a sequenced collection, we can refer to the elements in the array as the first element, the second element, and so forth until we get to the last element.

Problem and traditional approach

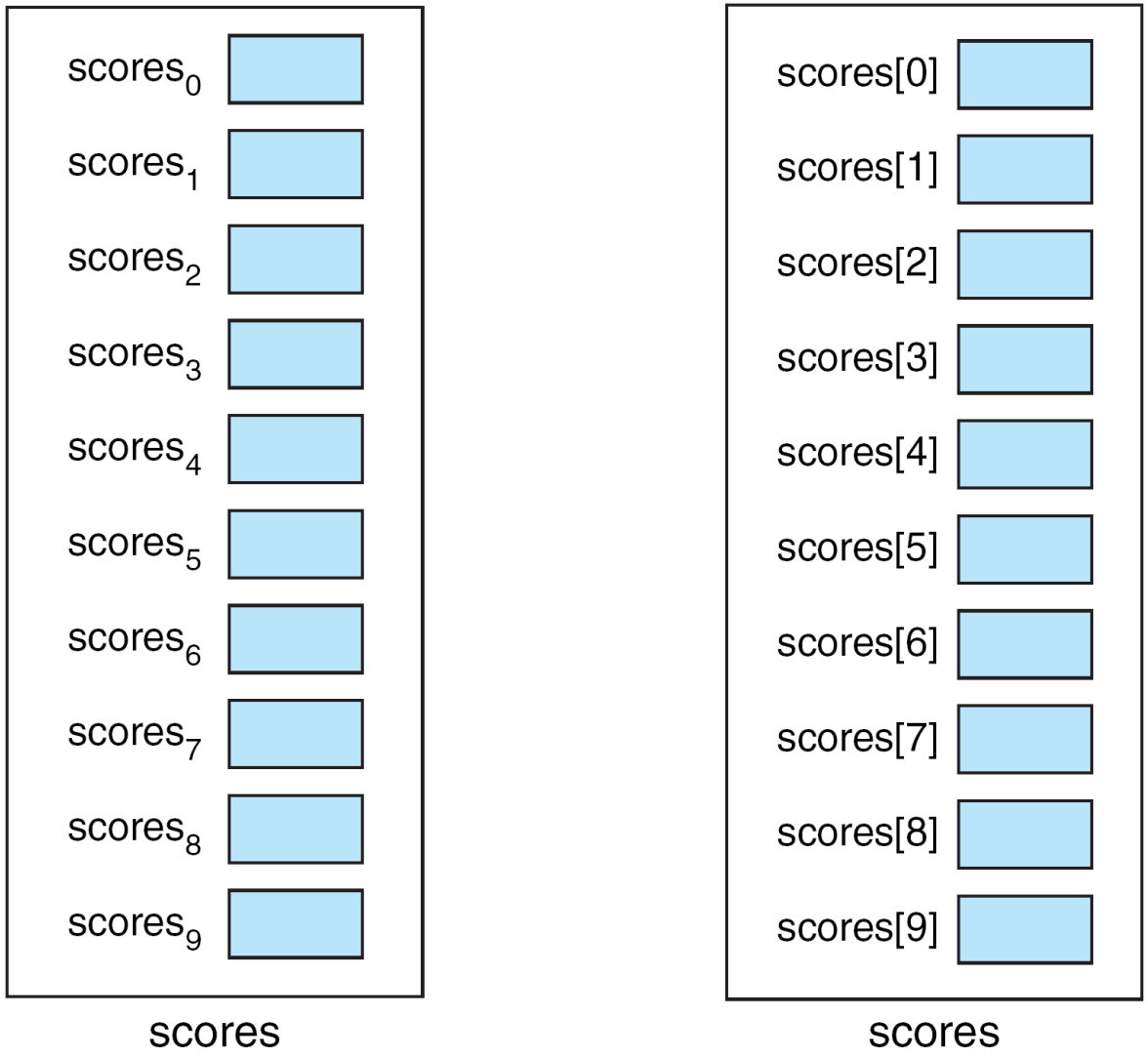
- Lets think about a problem that, you need to store 10 scores into a program. How you can store these into your program?



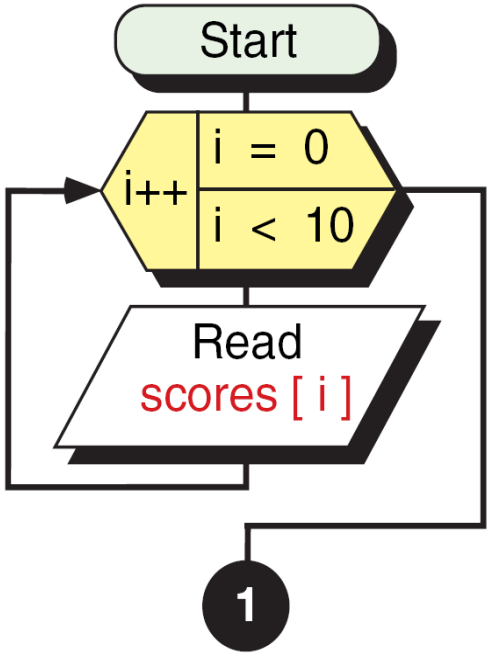
Processing 10 variables

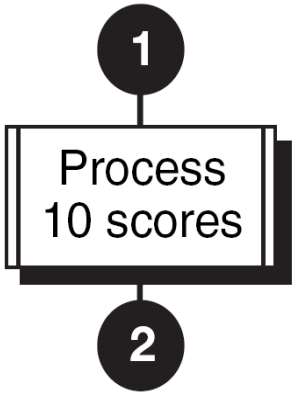


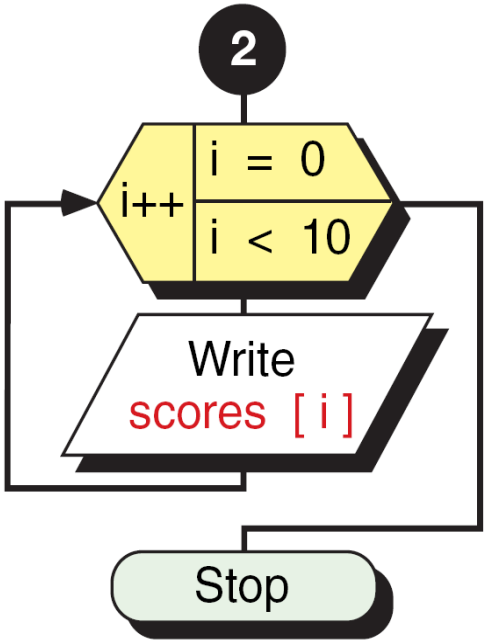
Another way (Array)



Processing array elements using loop







Notion of array

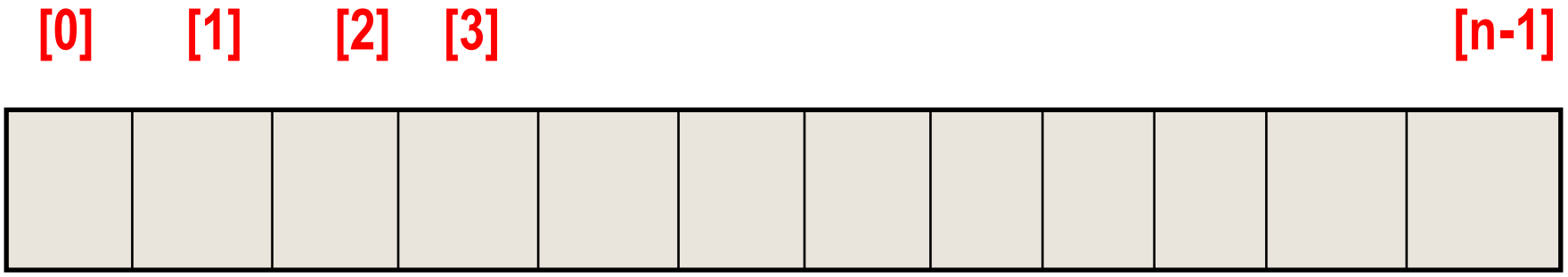
- Array
  - – Homogeneous collection of variables of same type.
  - – Group of consecutive memory locations.
  - – Linear and indexed data structure.
- To refer to an element, specify
  - –Array name
  - –Position number (Index)

Types of Arrays

- There are two types of arrays
  - **Single dimension array**
  - **Multiple dimensions array**

Single dimension array

- Elements of an array can be integers, floats, characters etc.
- All the elements share a common name with an index called subscript.
- In an array of n elements:



Why to use single dimension array?

- **One example:**

- Employees' salaries in a company.
- Each salary is a float.
- All the salaries could be stored as an array of floats.

- Can then do things like:
  - Change an employee's salary
  - Find out an employee's salary
  - Get the total salary bill

- and do it **more easily** than representing each salary as a single float (convenience of representation)

Declaration

- When declaring arrays, specify
  - Data type of array (integers, floats , characters…..)
  - Name of the array.
  - Size: number of elements

```
array_type   array_name[ size ]   ;
```

  - Example:
    - int student[10] ;
    - float my_array [300] ;

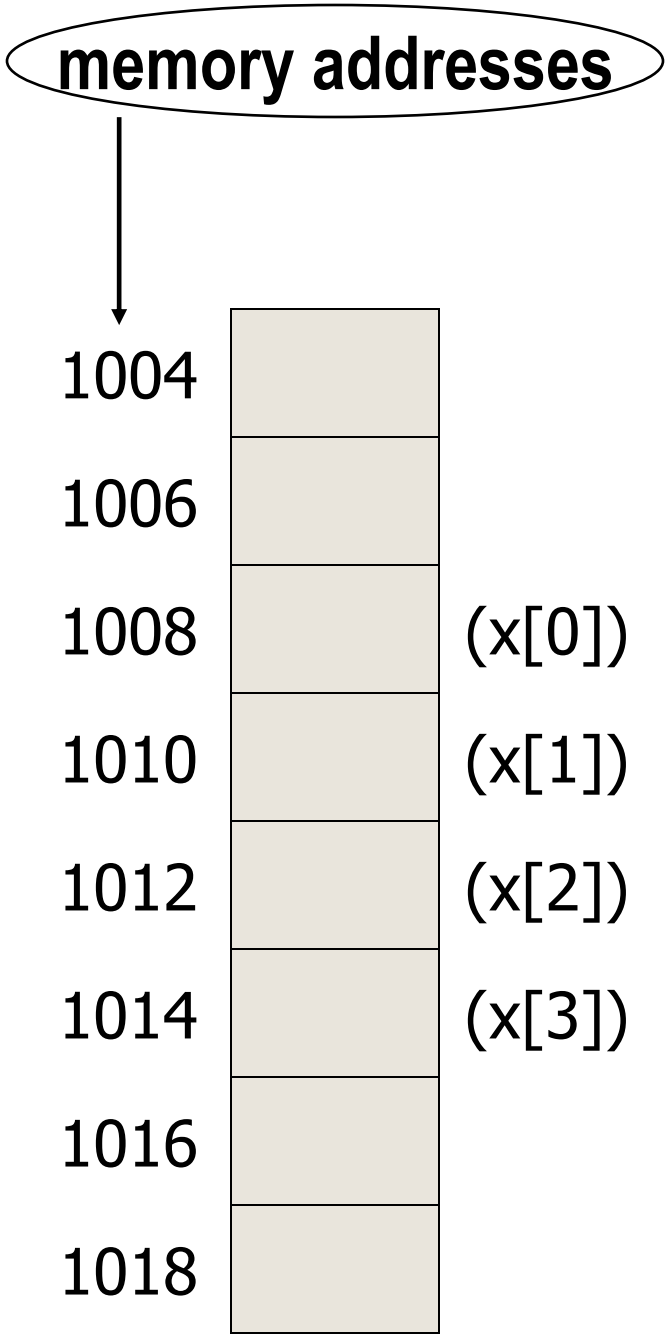
Example

- For example,

```
int  x  [ 4 ] ;
```

Square bracket

- An array of integers of 4 elements.
- Note that the starting memory address is determined by the operating system (just like that of simple variable).
- Contiguous memory locations are allocated.



More examples of array declaration

- Integer array of 20 elements:

```
int  array_1  [ 20 ];
```

- Character array of 50 elements:

```
char array_2 [ 50 ];
```

- Float array of 100 elements:

```
float array_3 [ 100 ];
```

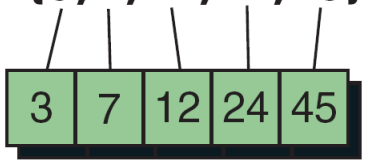
Initializing arrays

- Array elements must be initialized at time of declaration, otherwise they may contain garbage values
- Initialization can be done either at compile time or run time

Compile Time Initialization

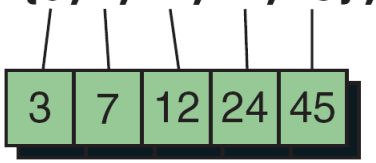
(a) Basic Initialization

```
int numbers[5] = {3,7,12,24,45};
```



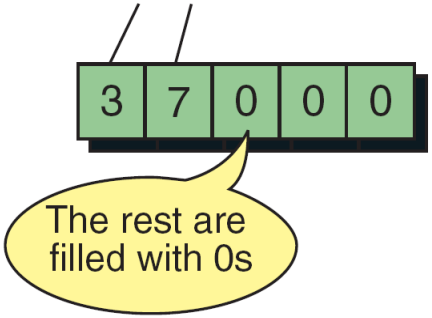
(b) Initialization without Size

```
int numbers[ ] = {3,7,12,24,45};
```



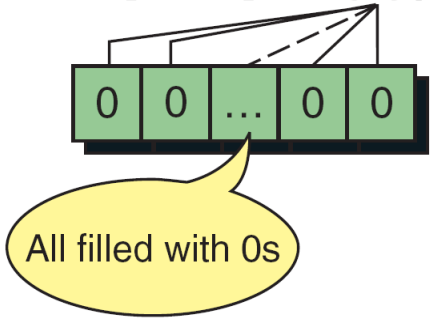
(c) Partial Initialization

```
int numbers[5] = {3,7};
```



(d) Initialization to All Zeros

```
int lotsOfNumbers [1000] = {0};
```



Second way to initialization

The second way is to

Initialize each array element separately

Int id[6];

id[0]=1234;

id[2]=2883;

id[3]=2322;

id[4]=8888;

id[5]=8237;

Run time initialization

- Explicitly initializing an array at run time
- Normally used for large array size
- Example:

```
 for(i=0; i<100; i++)
{
 if(i<50)
        sum[i] = 0;
 else
        sum[i] = 1;
}
```

Basics of character array(String)

- If you are required to store a group of character like your name, city , or your college name, or any word or text you need to define a array of characters.

- A char variable can hold a SINGLE character only like

  char c = 'A' ;

  char c1='B';

- What if you need to store "Sachin Tendulkar" or "MUMBAI" a string.

Character array

- To hold a single string you need to declare a single dimension character array
  - char str [11] ;
- When declaring a character array to hold a string(group of characters), one need to declare the array to be one character longer than the largest string that it will hold
- Example above array str[11] will hold 10 characters and a NULL character ('\0') at the end

#include <string.h>

C String Operations

- Character arrays are a special type of array that uses a "\0" character at the end. As such it has it is own header library called string.h that contains built-in functions for performing operations on these specific array types.
- You must include the string header file in your programs to utilize this functionality.

```
#include <string.h>
```

String Library Functions

- **Reading user input string**

The gets() function enables the user to enter some characters followed by the enter key. All the characters entered by the user get stored in a character array.

```
char s[30];
printf("Enter the string? ");
gets(s);
printf("You entered %s",s);
```

- **Print the string on console**

The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function.

```
char name[50];
printf("Enter your name: ");
gets(name); //reads string from user
printf("Your name is: ");
puts(name);  //displays string
```

String Library Functions

- **Length of a String**

If we had a string, and called the strlen function on it we could get its length.

```
char fname[30] = {"Bob"};
int length = strlen(fname);
```

- **Concatenation of Strings**

The strcat function appends one string to another. We can use this function to concatenate two different strings.

```
char fname[30] = {"Bob"};
char lname[30] = {"by"};
printf("%s", strcat(fname, lname));
```

String Library Functions

- **Compare Two Strings (Case Sensitive)**

Sometimes you want to determine if two strings are the same. For this we have the strcmp function. If strings are same or equal strcmp returns 0 otehrwise it returns 1.

```
printf("%d", strcmp(fname, lname));
```

- **Compare Two Strings (Not Case Sensitive)**

If you do not care whether your strings are upper case or lower case then use **strcmpi** function instead of the strcmp function

```
int strcmpi(string1, string2);
```

String Library Functions

- **Copy Strings**

To copy one string to another string variable, you use the strcpy function.

```
strcpy(string1, string2);
```

```
strcpy(fname, "Bob");
```

- **Reversing the Order of a String**

Will reverse the order of string. So if string was “bobby”, it would become “ybbob”.

```
strrev(string);
```

String Library Functions

- **Converting Uppercase Strings to Lowercase Strings**

This will convert uppercase characters in string to lowercase. So “BOBBY” would become “bobby” function.

```
strlwr(string);
```

- **Converting Lowercase Strings to Uppercase Strings**

This will convert lowercase characters in string to uppercase. So “bobby” would become “BOBBY”.

strupr(string);

Multiple dimension array

- A *multiple dimension array* is an array that has two or more dimensions.
- Two dimensional arrays are the simplest type of multiple dimension arrays. They can be used to represent tables of data, matrices, and other two dimensional objects.
- One of the most obvious example of a two dimensional array is a table.

How will you store a table or matrix?

	Item1	Item2	Item3
Shop1	310	275	365
Shop2	210	190	325
Shop3	405	235	240
Shop4	260	300	380

To store this data, create 2-D array like

int arr[4][3];

Multi-dimensional ARRAYS, Cont.

- When might we use a multi-dimensional array?

- Array of names (i.e., array of strings)
- This generalizes to ***a list of lists***
- Other examples,

  Checkerboard

  Matrices, lists of vectors

Two Dimensional Arrays

- We can see how a two dimensional array looks like a table or matrix.
- Thus, we can represent a two dimensional array as rows and columns.
- To declare a two-dimensional array:

(data type) array_name[# of rows] [# of columns];

2-Dimensional Arrays

Can be illustrated as a table:

Each row is a separate array.

For ex.

	col 0	col 1	col 2	col 3
row 0	1	2	3	4
row 1	5	6	7	8
row 2	9	10	11	12

Two Dimensional Arrays Declarations

- Therefore, a two dimensional array of integers, named x, with 3 rows and 4 columns would be declared as:

```
int x[3][4];
```

- A two dimensional array of characters, named c, with 119 rows and 2 columns would be declared as:

```
char c[119][2];
```

2-D array initialization

- int table[2][3] = {0,0,0,1,1,1};
- Can also be done as:

  int table[2][3] = {{0,0,0},{1,1,1}};

When array is explicitly initialized with all values, first dimension in its declaration can be skipped

Example, array can be declared as:

int table[][3] = {{0,0,0},{1,1,1}};

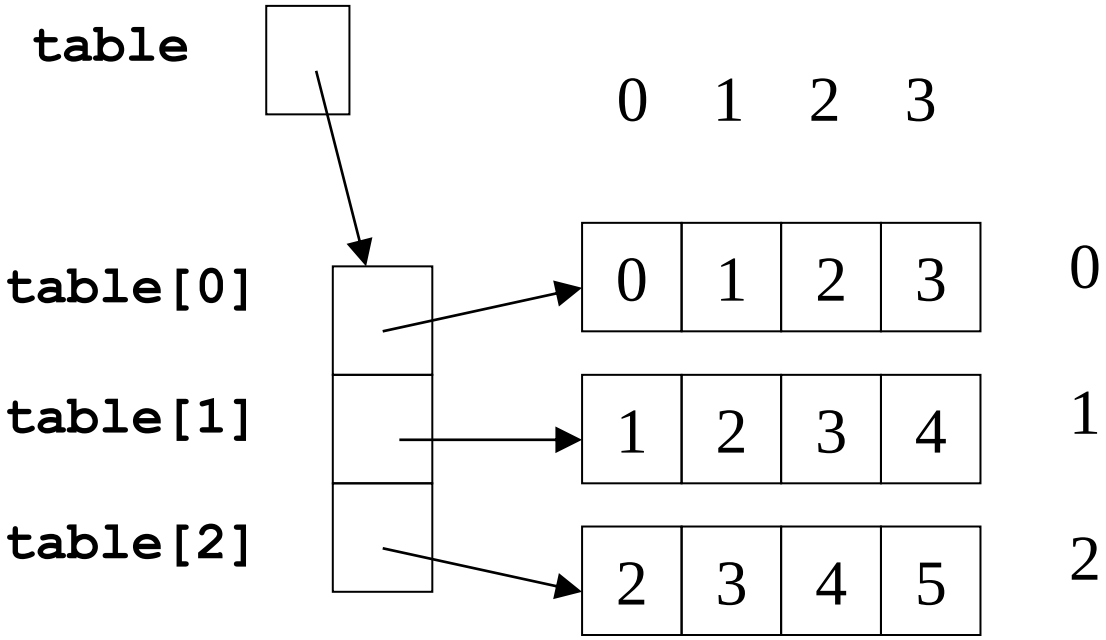
But

int table[2][]={0,0,0,1,1,1}; // error

If some values are missing in initializer, they are set to 0

Two-Dimensional Arrays

```
int MAXROW =3;
int MAXCol =4;
int table[MAXROW][MAXSUB];
Int row, col;
for (row = 0; row < MAXROW; row++)
   for (col = 0; col < MAXCol; col++)
     scanf (“%d”,&table[row][col])
```



Multi-dimensional ARRAYS

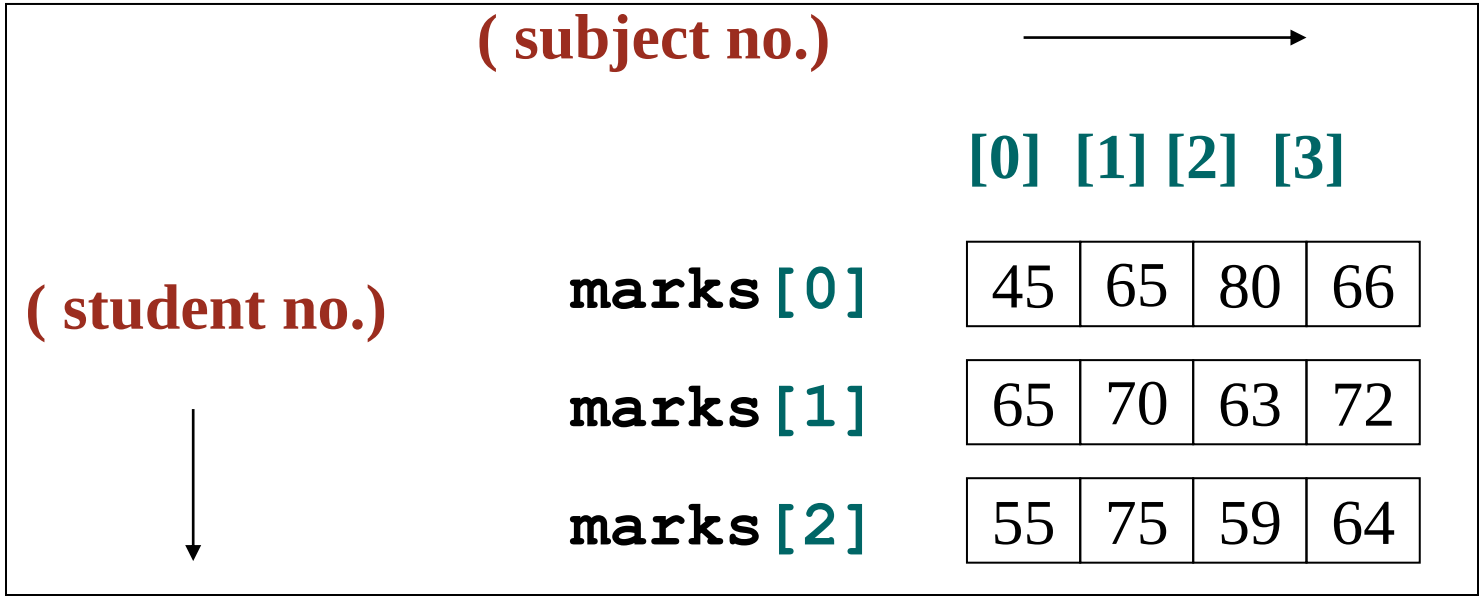
- int daytab[2][13] = {

  {0,31,28,31,30,31,30,31,31,30,31,30,31},

  {0,31,29,31,30,31,30,31,31,30,31,30,31} };
- Note initialization of arrays within an array.
- How to access an individual element
  - daytab[1][4] /* 2nd row 5th element ie30*/
  - NOTE – the following is wrong!

    daytab[1,4] /* wrong! */

Storing the marks of all students in all subjects in a two- dimensional array



Here the marks are stored in an array marks[students][subjects]

i.e. each row contains the marks of all subjects of one student,each column contains the marks of all the students for one subject.

DISADVANTAGE OF ARRAYS

- Memory allocation of array is static:

- Maximum size (maximum number of elements) requested by the programmer would be reserved in the memory irrespective of the usage of the number of elements by the user.The memory space that is unused is wasted. LESS RESOURCE UTILIZATION.
- For example, int test[30]

- Different data types could not be stored in an array