VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BCSE102L- Structured and object-oriented programming

Module 2

Dr. P.Keerthika
Associate Professor
School of Computer Science & Engineering
VIT, Vellore.

BCSE102L- Structured and object-oriented programming

Module:1	**C Programming Fundamentals**	**2 hours**
Variables - Reserved words – Data Types – Operators – Operator Precedence - Expressions - Type Conversions - I/O statements - Branching and Looping: if, if-else, nested if, if-else ladder, switch statement, goto statement - Loops: for, while and do…while – break and continue statements.		
Module:2	**Arrays and Functions**	**4 hours**
Arrays: One Dimensional array - Two-Dimensional Array – Strings and its operations. User Defined Functions: Declaration – Definition – call by value and call by reference - Types of Functions - Recursive functions - Storage Classes - Scope, Visibility and Lifetime of Variables.		
Module:3	**Pointers**	**4 hours**
Declaration and Access of Pointer Variables, Pointer arithmetic – Dynamic memory allocation – Pointers and arrays - Pointers and functions.		
Module:4	**Structure and Union**	**2 hours**
Declaration, Initialization, Access of Structure Variables - Arrays of Structure - Arrays within Structure - Structure within Structures - Structures and Functions – Pointers to Structure -		
Module:5	**Overview of Object-Oriented Programming**	**5 hours**
Features of OOP - Classes and Objects - "this" pointer - Constructors and Destructors - Static Data Members, Static Member Functions and Objects - Inline Functions – Call by reference - Functions with default Arguments - Functions with Objects as Arguments - Friend Functions and Friend Classes.		
Module:6	**Inheritance**	**5 hours**
Inheritance - Types of Inheritance: Single inheritance, Multiple Inheritance, Multi-level Inheritance, Hierarchical Inheritance - Multipath Inheritance - Inheritance and constructors.		
Module:7	**Polymorphism**	**4 hours**
Function Overloading - Operator Overloading – Dynamic Polymorphism - Virtual Functions - Pure virtual Functions - Abstract Classes.		
Module:8	**Generic Programming**	**4 hours**
Function templates and class templates, Standard Template Library.		
	Total Lecture hours:	**30 hours**

BCSE102L- Structured and object-oriented programming – Text Books and Reference Books

Text Book(s)	
1.	Herbert Schildt, C: The Complete Reference, 4th Edition, McGraw Hill Education, 2017
2.	Herbert Schildt, C++: The Complete Reference, 4th Edition, McGraw Hill Education, 2017.
Reference Books	
1.	Yashavant Kanetkar, Let Us C: 17th Edition, BPB Publicaitons, 2020.
2.	Stanley Lippman and Josee Lajoie, C++ Primer, 5th Edition, Addison-Wesley publishers, 2012.

BCSE102P- Structured and object-oriented programming Laboratory

Indicative Experiments	
1.	Programs using basic control structures, branching and looping
2.	Experiment the use of 1-D, 2-D arrays and strings and Functions
3.	Demonstrate the application of pointers
4.	Experiment structures and unions
5.	Programs on basic Object-Oriented Programming constructs.
6.	Demonstrate various categories of inheritance
7.	Program to apply kinds of polymorphism.
8.	Develop generic templates and Standard Template Libraries.

Text Book(s)	
1.	Robert C. Seacord, Effective C: An Introduction to Professional C Programming, 1st Edition, No Starch Press, 2020.
Reference Book(s)	
1.	Vardan Grigoryan and Shunguang Wu, Expert C++: Become a proficient programmer by learning coding best practices with C++17 and C++20's latest features, 1st Edition, Packt Publishing Limited, 2020.

BCSE102L- Structured and Object-Oriented Programming

- **Module-2: ARRAYS AND FUNCTIONS**
  - **Arrays – 1D & 2D**
  - **Strings and its operations**
  - **User defined Functions**
    - **Declaration and Definition**
    - **Call by Value and Call by reference**
    - **Types of Functions- Recursive Functions**
  - **Storage Classes**
    - **Scope, Visibility and lifetime of variables**

ARRAYS

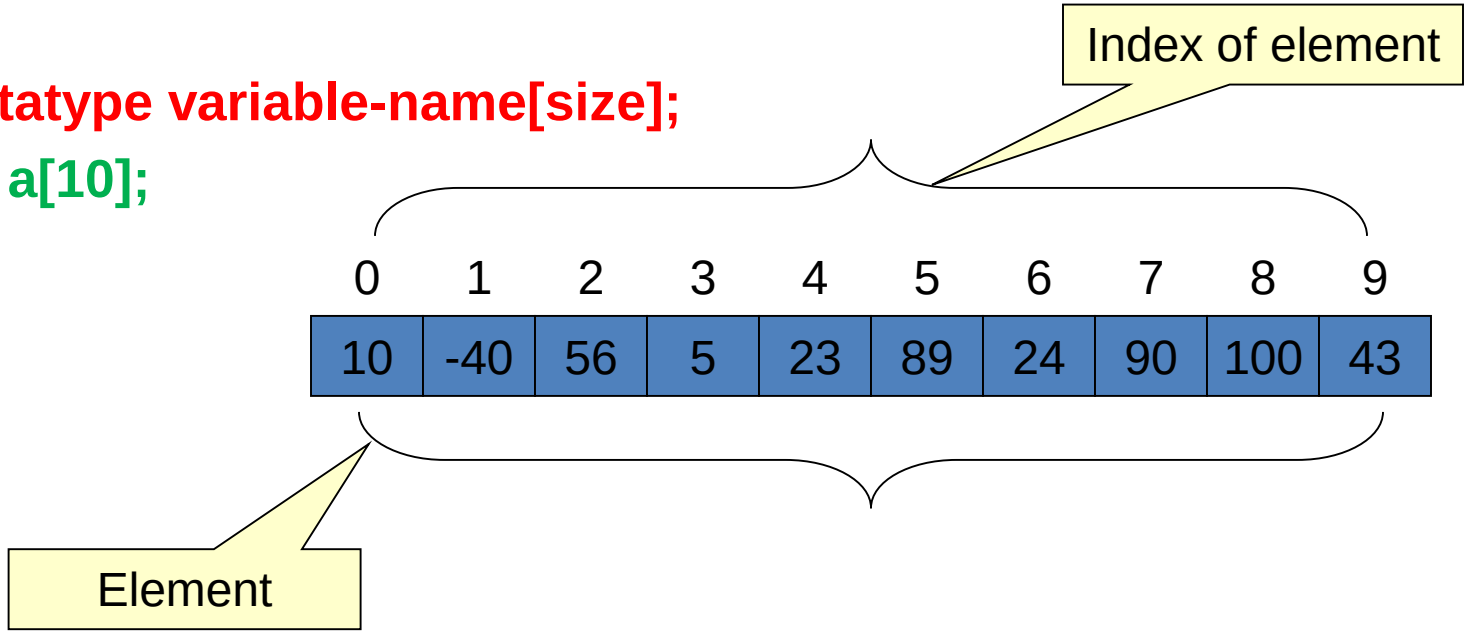
- **Limitations of variable??**
  - Fundamental data types(Ex: char, int, float, double etc,) are very useful in various aspects but the **variables of these types can store only one value at any given time**.
  - Only used to handle limited amounts of data.
  - To process large amounts of data, need a powerful data type that facilitate **efficient storing, accessing and manipulation of data items**

ARRAYS

ARRAYS

- An array is a fixed size sequenced collection of elements (set of elements) of the same data type.
- Example: *Array of integers, Array of floats, Array of chars, Array of user-defined data types*

- **Datatype variable-name[size];**
- **int a[10];**



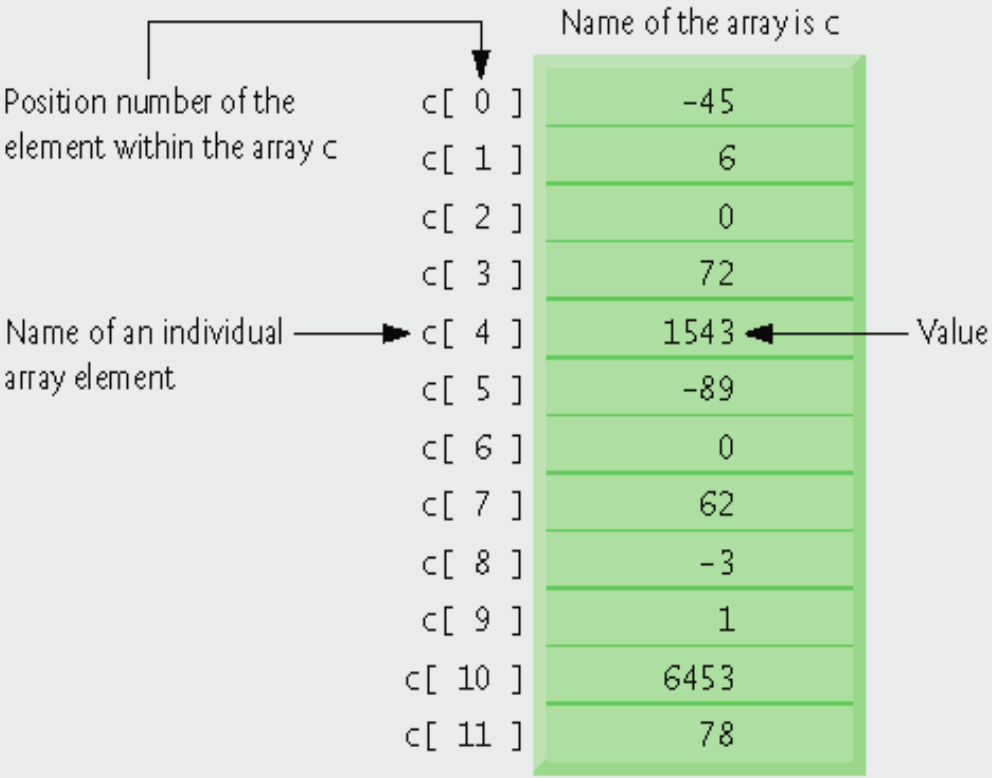
For example :To retrieve the 3rd element in the array: a[3]

printf("\n Value of 3rd Element in the array = %d", a[3]);

ARRAYS

- **Array is a Data Structure. Why??**
  - Provides a convenient structure for representing data.
  - A data structure of a consecutive set of memory locations which stores related data items of same data type.
  - Always remain of the same size, once created - Static
    - Are *static* entities
  - Character arrays can also represent strings
- **Index**
  - Position number used to refer to a specific location / element
  - Placed in *square brackets*
  - First element has the index zero

An Array of 12 Elements



Defining 1D Array

- **Different ways**

int a[3]; *//reserves space; no initialization*

int a[3] = {10,20,30}; *//space reservation + initialization*

int a[ ] = {10,20,30}; *//space reservation + initialization*

- Automatically creates an array of size 3 and initializes to above elements.

float height[5] = {10.5,20.2,30.5,12.5,60.8};

float height[ ] = {10.5,20.2,30.5,12.5,60.8};

Example-1D Array

Array Declaration

```
#include <stdio.h>
int main()
{
        int sub[10], i, n, total = 0, average;
        printf("Enter the number of subjects: ");
        scanf("%d", &n);
        for(i=0; i<n; ++i)
        {
                printf("Enter mark of subject %d: ", i+1);
                scanf("%d", &sub[i]);
                total = total + sub[i];
        }
        printf("Total marks obtained: %d", total);
        average = total/n;
        printf("\n Average = %d", average);
        return 0;
}
```

Defining Character Array

- C language treats character strings as array of characters.
- Size in character string represents the maximum number of characters that the string can hold.

```
char name[10];
```

Character array variable that can hold maximum of 10 characters

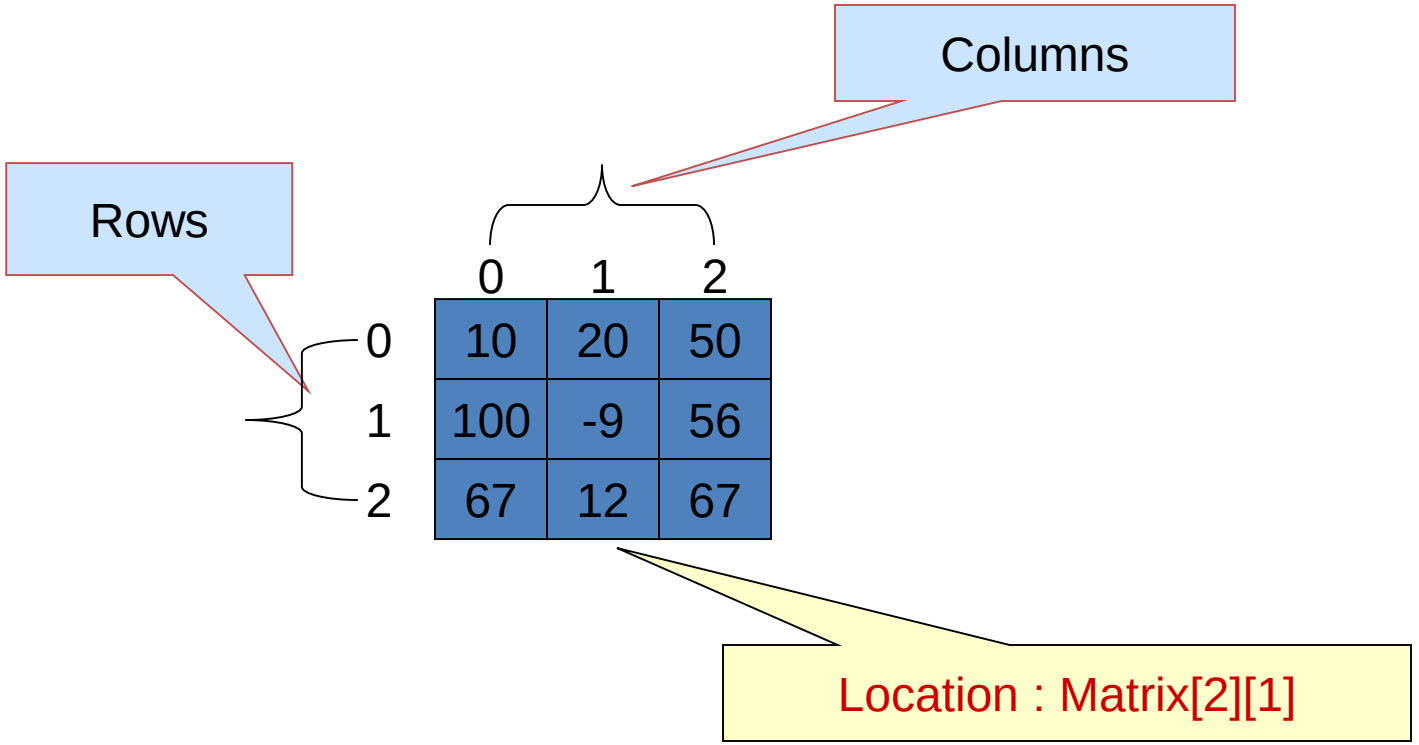
Null Character

- Different ways of defining single dimensional char array:
  - /* '\0' gets appended to the string automatically */
  a. char Name[10] = " Well Done";
  b. char Name[] = "Well Done";
  c. char Name[] = { 'K','I','N', 'G','\0'};

- A character array is called String

2 D Array

- Elements organized in a tabular format (rows & columns)
- **Type Array-Name [row-size][col-size]**
- For example: int Matrix[3][3]



Defining 2D Array

- Different ways of defining a 2D array:

a. /* Reserves space; no initialization */

```
int a[2][2];
```

b. /* Space Reservation + initialization */

```
int a[2][3]={10,20,30,40,50,60};
int a[2][3]={{10,20,30},{40,50,60}};
int a[2][2] = {{10,20} ,
               {30,40}};
int a[][3] = {{10,20,30} ,
              {30,40,50}};
int a[2][3] = {{10,20} ,
               {30}};
 int a[3][5] = {{0} ,{0}, {0}};
```

Defining 2D Array

- **Different ways of defining a 2D array:**

  a. /* Reserves space; no initialization */

  ```
  int a[2][2];
  ```

  b. /* Space Reservation + initialization */

  ```
  int a[2][3]={10,20,30,40,50,60};
  int a[2][3]={{10,20,30},{40,50,60}};
  int a[2][2] = {{10,20} ,
                 {30,40}};
  int a[][3] = {{10,20,30} ,
                {30,40,50}};
  ```

- **Different ways of defining a 2D char array:**

  a. char Name[2] [10];

  b. char Name[2] [10] = { “Ajay” ,“Bobby” };

  c. char Name [] [10]= { “Ajay” ,“Bobby” };

Practice Problems

- Get the array elements from the user and print it one by one.
- Search for an array element and return the position if it is present.
- Insert a new array element in the position entered by the user.
- Delete an element at desired position from an array
- Sort the elements of an array in ascending order.
- Finding the maximum number in an array.
- Find the average of array elements.
- Remove the duplicate element in an array.
- Finding the k^{th} smallest element in an array.
- Matrix Addition
- Matrix Multiplication

Practice Problems

- Get the array elements from the user and print it one by one.
- Search for an array element and return the position if it is present.
- Insert a new array element in the position entered by the user.
- Delete an element at desired position from an array
- Sort the elements of an array in ascending order.
- Finding the maximum number in an array.
- Find the average of array elements.
- Remove the duplicate element in an array.
- Finding the k^{th} smallest element in an array.
- Matrix Addition
- Matrix Multiplication

Example-2D Array

```
Matrix Multiplication()
{
        for(i=0;i<row;i++)
        {
                for(j=0;j<col;j++)
                {
                        mul[i][j]=0;
                        for(k=0;k<col;k++)
                        {
                                mul[i][j]+=a[i][k]*b[k][j];
                        }
                }
        }
}
```

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Thank You