PROBLEM SOLVING WITH C
UE23CS151B

Prof. Sindhu R Pai
Department of Computer Science and Engineering

PROBLEM SOLVING WITH C

Multi-Dimensional Arrays

Prof. Sindhu R Pai
Department of Computer Science and Engineering

PROBLEM SOLVING WITH C

Agenda

1. Introduction
2. Two-Dimensional Array: Declaration
3. Two Dimensional Array: Initialization
4. Internal Representation of a 2D Array
5. Pointer and 2D Array
6. Passing 2D array to a function
7. Three Dimensional (3D) Array

Multi-Dimensional Arrays

Introduction

- An array with more than one level or dimension.
- 2-Dimensional and 3-Dimensional and so on.

General form of declaring N-dimensional arrays:

Data_type Array_name[size1][size2][size3]..[sizeN];

Multi-Dimensional Arrays

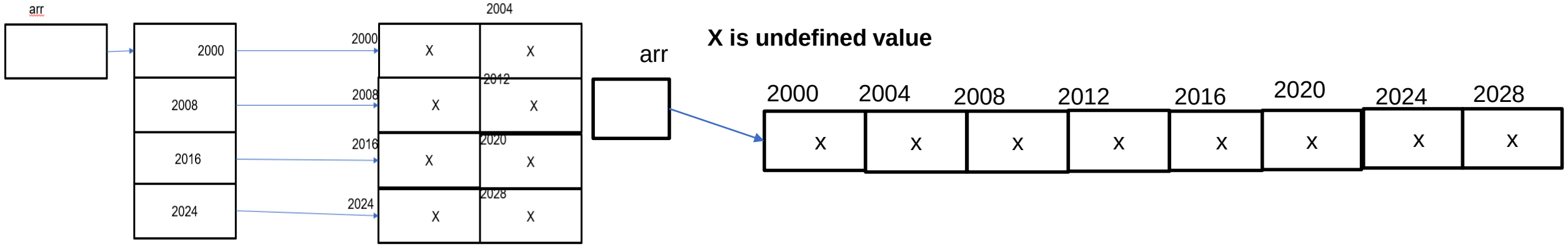
Two – Dimensional Array

- Treated as an array of arrays.
- Every element of the array must be of same type as arrays are homogeneous

Declaration

Syntax: data_type array_name[size_1][size_2]; // size_1 and size_2 compulsory

int arr[4][2];



Two – Dimensional Array

Initialization

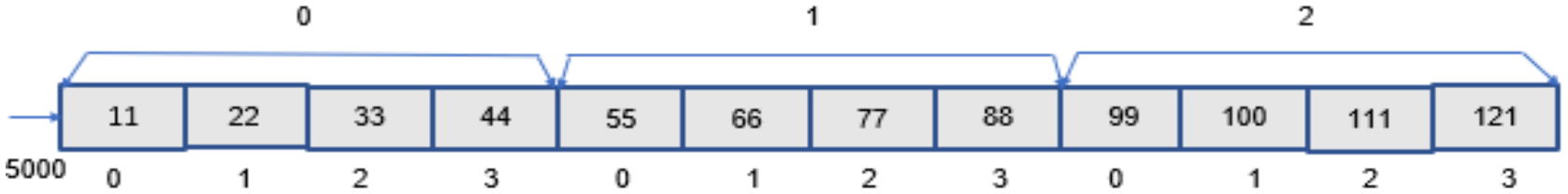
- **data_type array_name[size_1][size_2] = {elements separated by comma};**
- int arr[][] = {11,22,33,44,55,66}; // Error. Column size is compulsory
- int abc[3][2] ={{11,22},{33,44},{55,66}};//valid
- int arr[][2] = {11, 22, 33 ,44,55,66 } ;//valid. Allocates 6 contiguous memory locations and assign the values
- int arr[][3] = {{11,22,33},{44,55},{66,77}}; // partial initialization

Internal Representation of a 2D Array

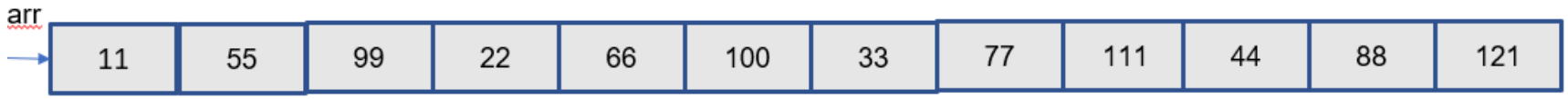
- 2D Array itself is an array, **elements are stored in contiguous memory locations.**

Consider, int arr[3][4] = {{11, 22, 33, 44}, {55, 66, 77, 88}, {99, 100, 111, 121}};

- **Row major Ordering:** All elements of one row are followed by all elements of the next row and so on.



- **Column Major Ordering:** All elements of one column are followed by all elements of the next column and so on.



- **Generally, systems support Row Major Ordering.**

Multi-Dimensional Arrays

Address of an Element in a 2D Array

- **Address of A[i][j] = Base_Address +((i * No. of columns in every row) + j)*size of every element;**
- Consider, int matrix[2][3]={1,2,3,4,5,6}; // base address is 100 and size of integer is 4 bytes

matrix[0][0] 100	1
matrix[0][1] 104	2
matrix[0][2] 108	3
matrix[1][0] 112	4
matrix[1][1] 116	5
matrix[1][2] 120	6

	Column 0	1	2
Row 0	1	2	3
1	4	5	6

- Address of matrix[1][0]=100+((1*3)+0)*4=100+3*4=112

Demo of C Code

- **To read and display a 2D Array**

Pointer and 2D Array

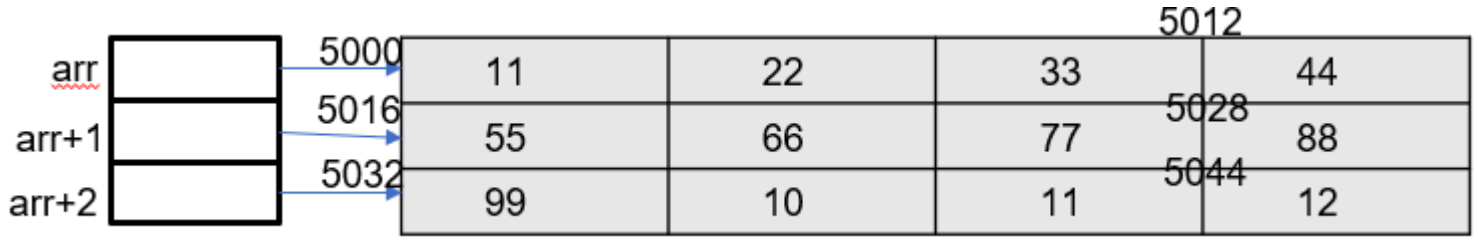
- **Pointer expression for** a[i][j] is ***(*(a + i) + j)**
- Array name is a pointer to a row.
- (a + i) points to ith 1-D array.
- *(a + i) points to the first element of ith 1D array
- Coding examples

Multi-Dimensional Arrays

Pointer and 2D Array continued..

- Consider, int arr[3][4] = {11, 22, 33, 44, 55, 66, 77, 88, 99, 100, 111, 121};

arr – points to 0th elements of arr- Points to 0th 1-D array-5000
arr+1-Points to 1st element of arr-Points to 1st 1-D array-5016
arr+2-Points to 2nd element of arr-Points to 2nd 1-D array
arr+ i Points to ith element of arr ->Points to ith 1-D array



Multi-Dimensional Arrays

Pointer and 2D Array continued..

- int *p = arr; // assigning the 2D array to a pointer results in warning
- Using p[5] results in 66. But p[1][1] results in error. p doesn't know the size of the column.
- Solution is to create a **pointer to an array of integers.**

int (*p)[4] = arr; //subscript([ ]) have higher precedence than indirection(*)

- Think about the **size of p and size of *p!!**

Passing 2D array to a function

- Read and display 2D array using functions

- Write a function to add, subtract and multiply two matrices. Display appropriate message when these two matrices are not compatible for these operations.

- Write a program to take n names from the user and print it. Each name can have maximum of 20 characters.

Three Dimensional (3D) Array

- Accessing each element by using three subscripts
- First dimension represents table ,2nd dimension represents number of rows and 3rd dimension represents the number of columns
- int arr[2][3][2] = { {{5, 10}, {6, 11}, {7, 12}}, {{20, 30}, {21, 31}, {22, 32}} }; //2 table 3 rows 2 coloumns.

Practice programs

1. Given two matrices, write a function to find whether these two are identical.

2.Program to find the transpose of a given matrix.

3.Program to find the inverse of a given matrix.

4.Write a function to check whether the given matrix is identity matrix or not.

5. Write a program in C to find sum of right diagonals of a matrix.

6. Write a program in C to find sum of rows and columns of a matrix

THANK YOU

Prof. Sindhu R Pai

Department of Computer Science and Engineering

sindhurpai@pes.edu