Data Structures and Algorithms

Rao Muhammad Umer
Lecturer,
CS and IT Department,
The University of Lahore.
Web: raoumer.github.io

outline

Arrays

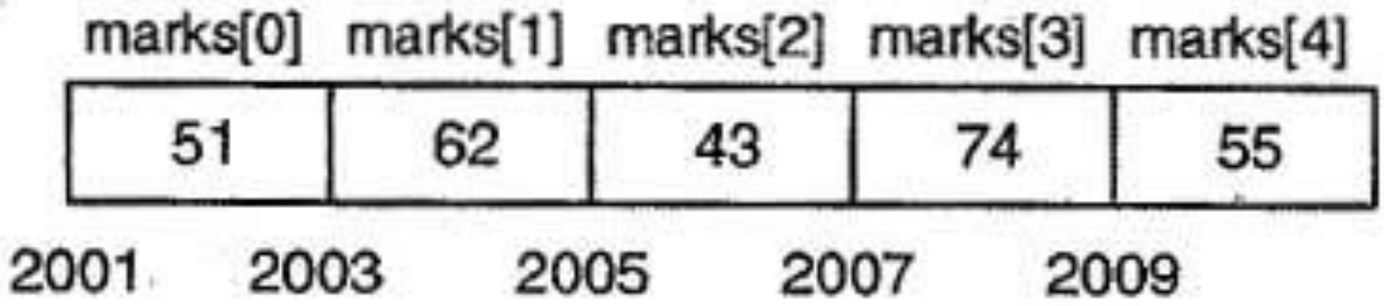
- What are Arrays?
- Array Operations
  - Traversal
  - Insertion
  - Deletion
  - Searching
  - Sorting
  - Merging
  - Reversing

- Static Arrays
- Dynamic Arrays
- Mutli-dimensional Arrays
- Sparse / Dense Matrix

Arrays

- **Finite** collection of **similar** elements stored in adjacent memory locations.
- **Finite:** means that there are specific number of elements in an array
- **Similar:** means that all the elements in an array are of same type.

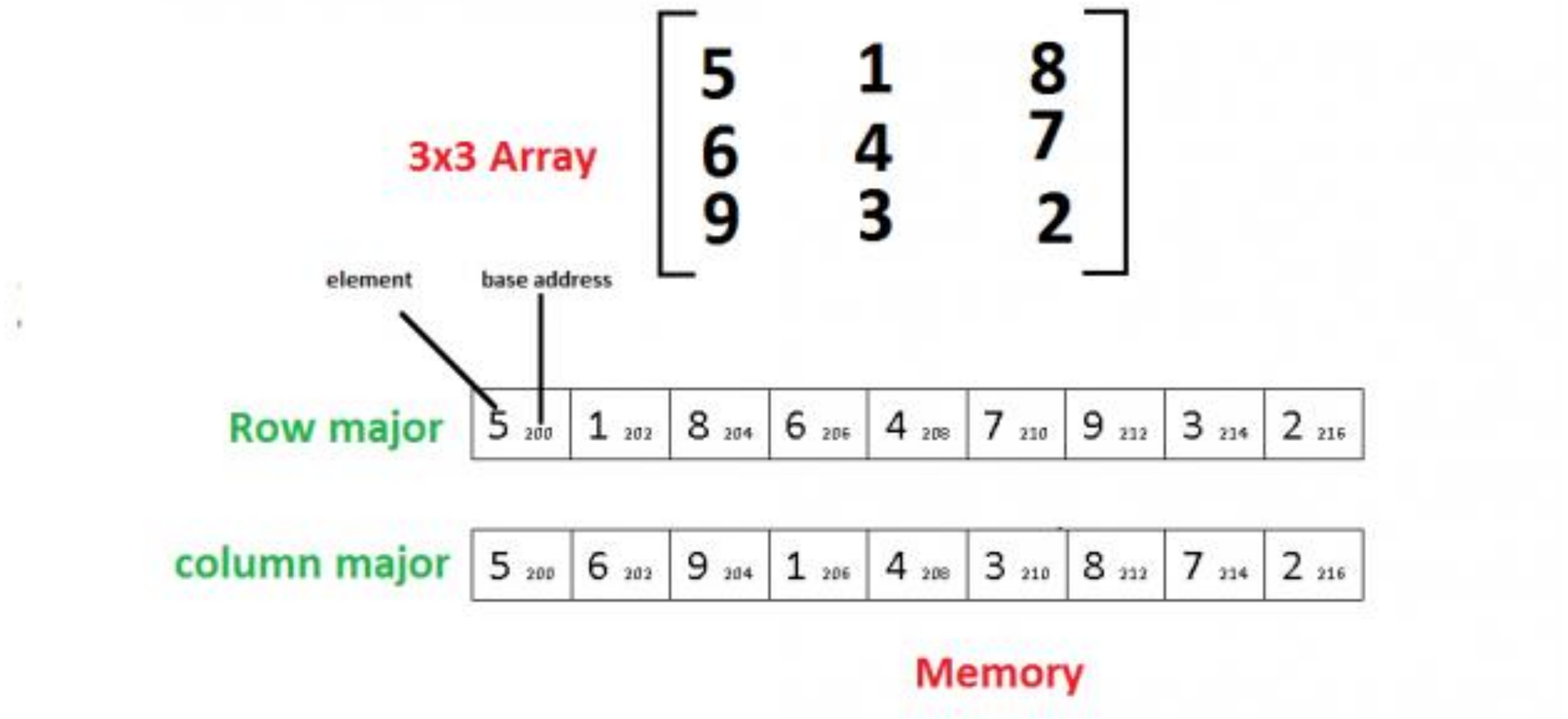
Memory Representation of Array



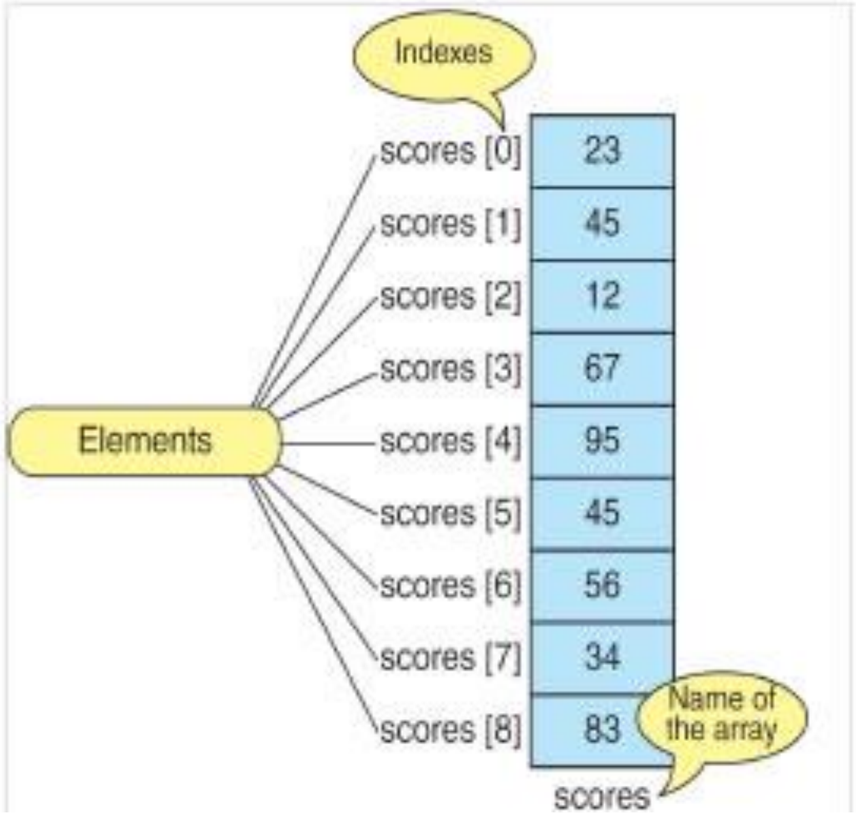
Memory Representation of `marks`

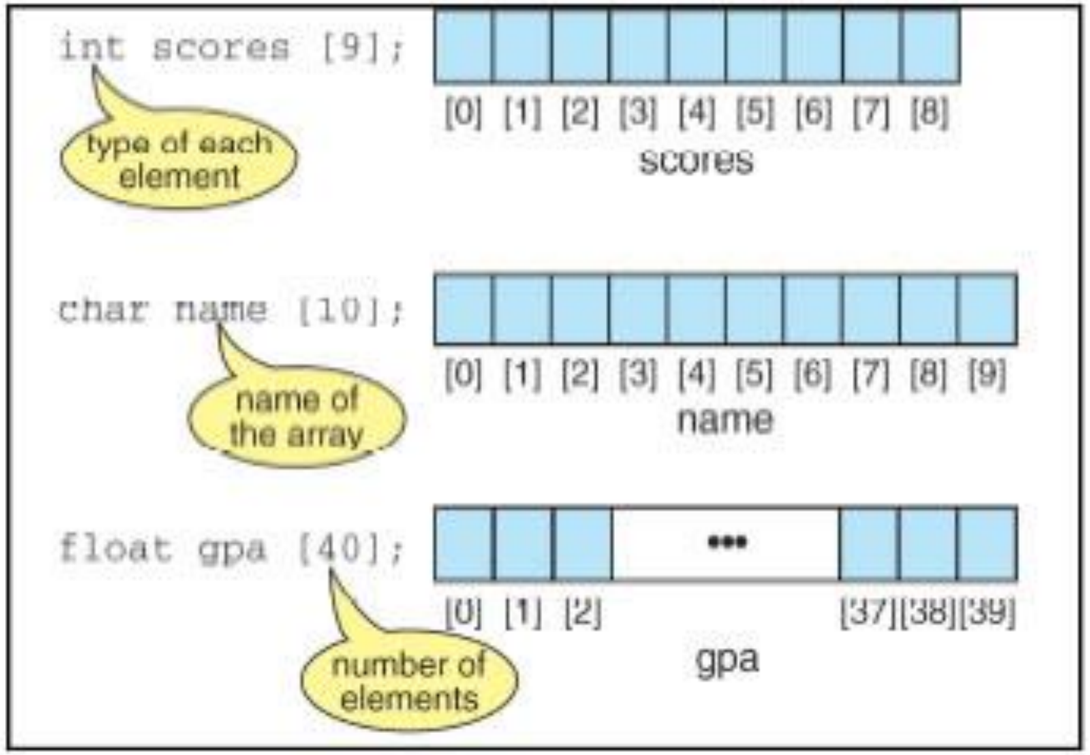
Memory Representation of Array

- **Row Major / Column Major Arrangement**



Array Declaration





Array Operations

- **Traversal:** Processing each element in the array
- **Search:** Finding the location of an element with a given value
- **Insertion:** Adding a new element to an array
- **Deletion:** Removing an element from an array
- **Sorting:** Organizing the elements in some order
- **Merging:** Combining two arrays into a single array
- **Reversing:** Reversing the elements of an array

Array Operations

- **Traversal, Insertion, Deletion, Reversing, Searching**
  - See Animation of these operations
  - See source code in C++ of listed operations

Array Operations

- **Merging of Two Arrays**
  - See Animation of these operations
  - See source code in C++ of merging of two different array

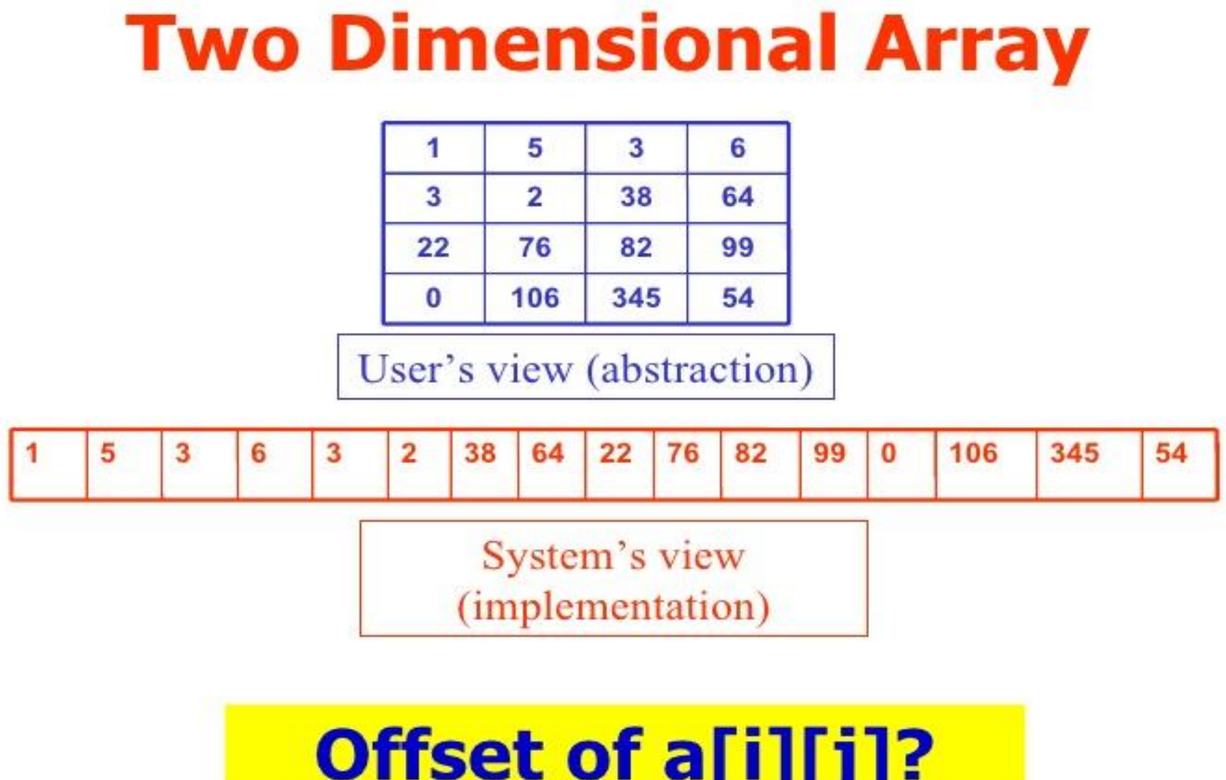
Static vs Dynamic Array

- Arrays in C++ can be:
  - **Static** [size known at compile-time]
  - **Dynamic** [created at run-time]
- Static arrays are allocated room on the stack
  - Size declarator must be a constant
    - literal or named constant
- Dynamic arrays are allocated space from the heap
  - Size declarator can be an integer expression
  - Requires a pointer to use the array
  - Can be created by using new operator

C++ Programming: Program Design Including Data Structures. Fourth Edition

Multi-Dimensional Arrays

- **Two-Dimensional Arrays**

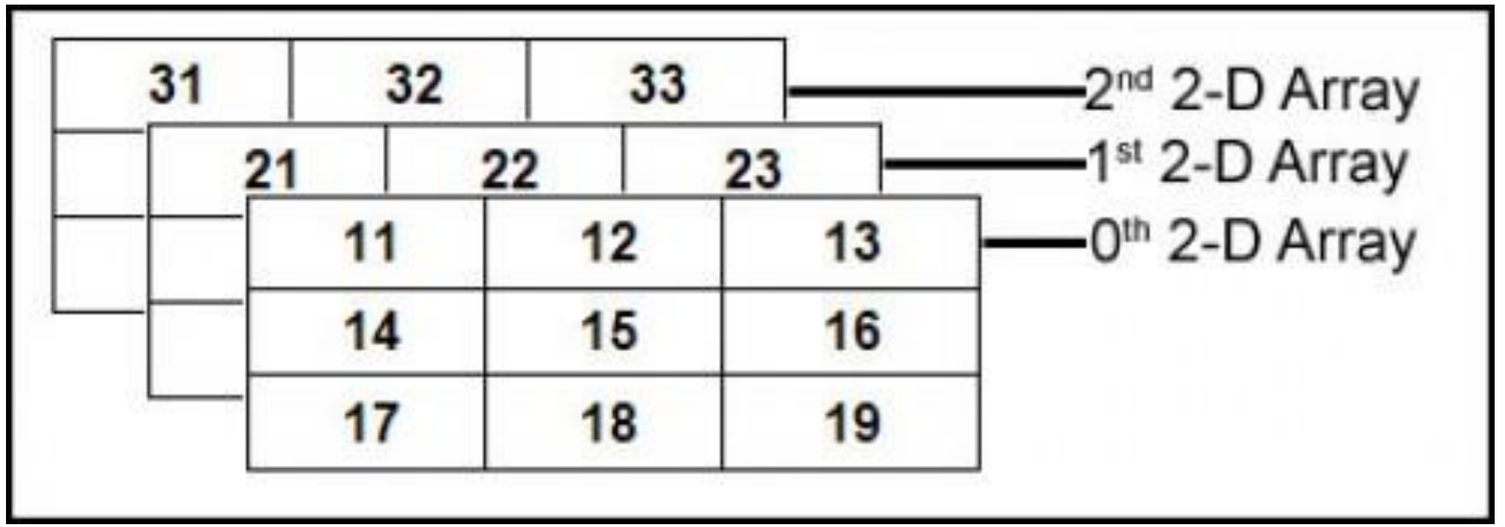


Multi-Dimensional Arrays

- **Two-Dimensional Arrays**
  - See source code in C++ of 2D array

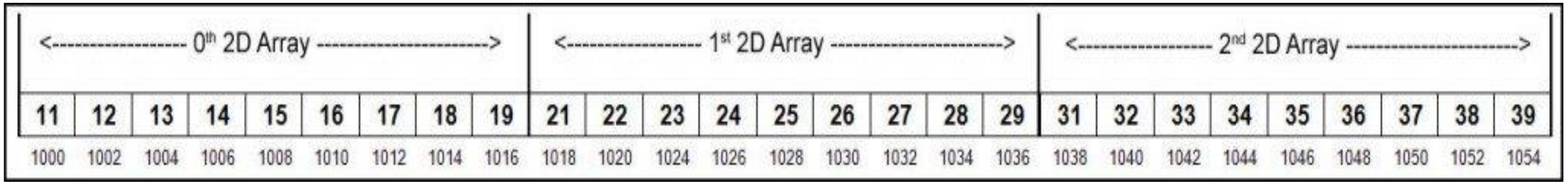
Multi-Dimensional Arrays

- **Three-Dimensional Arrays**



Multi-Dimensional Arrays

- **Memory representation of 3D Array**



Multi-Dimensional Arrays

- **3D Array Declaration and Initialization**

```
int  arr[3][3][3]=
      {
         {
         {11, 12, 13},
         {14, 15, 16},
         {17, 18, 19}
         },
         {
         {21, 22, 23},
         {24, 25, 26},
         {27, 28, 29}
         },
         {
         {31, 32, 33},
         {34, 35, 36},
         {37, 38, 39}
         }
      };
```

Multi-Dimensional Arrays

- **Three-Dimensional Arrays**
  - See source code in C++ of 3D array

Sparse Matrix

- In numerical analysis and computer science, a **sparse matrix** or **sparse array** is a matrix in which most of the elements are zero.
- The number of zero-valued elements divided by the total number of elements (e.g., m × n for an m × n matrix) is called the **sparsity** of the matrix (which is equal to 1 minus the **density** of the matrix).
- Example:

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

https://en.wikipedia.org/wiki/Sparse_matrix

Sparse Matrix Applications

- Large sparse matrices often appear in scientific or engineering applications when solving partial differential equations.
- The concept of sparsity is useful in combinatorics and application areas such as network theory, which have a low density of significant data or connections.

https://en.wikipedia.org/wiki/Sparse_matrix

Sparse Matrix and its representations

- **Compressed Sparse Row (CSR)**

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

- See source code in C++ of spare matrix implementation using array

Dense Matrix

- If most of the elements are nonzero, then the matrix is considered **dense**.

https://en.wikipedia.org/wiki/Sparse_matrix

Acknowledgement

- Mostly Slides taken from Book: **“Data Structures through C++”** by Yashavant P. Kanetkar
- https://en.wikipedia.org/wiki/Sparse_matrix