UNIT I

Introduction to Data Structures: Definition, Uses, Types.

Arrays: Abstract Data Types and the C++ Class, Array as an Abstract Data Type, Representation of Arrays, Matrices, Special Matrices Sparse Matrices, Strings.

1.1 Introduction To Data Structures

1.1.1 Definition

Q1. Define data structure.

Ans :

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage.

In simple language, Data Structures are structure programmed to store ordered data, so that various operations can be performed on it easily.

Q2. Give in simple words about the basic terminology used for data structures. **(Imp.)**

Ans :

Basic Terminology

Data structures are the building blocks of any program or the software. Choosing the appropriate data structure for a program is the most difficult task for a programmer. Following terminology is used as far as data structures are concerned.

Data

Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.

Group Items

Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.

Record

Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.

File

A File is a collection of various records of one type of entity, for example, if there are 60 employees in the class, then there will be 20 records in the related file where each record contains the data about each employee.

Attribute and Entity

An entity represents the class of certain objects. it contains various attributes. Each attribute represents the particular property of that entity.

Field

Field is a single elementary unit of information representing the attribute of an entity.

1.1.2 Uses

Q3. Mention the importance of data structures. **(Imp.)**

Ans :

1. Storing Data

Data structures are used for efficient data persistence, such as specifying the collection of attributes and corresponding structures used to store records in a database management system.

2. Managing Resources and Services

Core operating system (OS) resources and services are enabled through the use of data

structures such as linked lists for memory allocation, file directory management and file structure trees, as well as process scheduling queues.

3. Data Exchange

Data structures define the organization of information shared between applications, such as TCP/IP packets.

4. Ordering and Sorting

Data structures such as binary search trees - also known as an ordered or sorted binary tree - provide efficient methods of sorting objects, such as character strings used as tags. With data structures such as priority queues, programmers can manage items organized according to a specific priority.

5. Indexing.

Even more sophisticated data structures such as B-trees are used to index objects, such as those stored in a database.

6. Searching.

Indexes created using binary search trees, B-trees or hash tables speed the ability to find a specific sought-after item.

7. Scalability.

Big data applications use data structures for allocating and managing data storage across distributed storage locations, ensuring scalability and performance. Certain big data programming environments - such as Apache Spark - provide data structures that mirror the underlying structure of database records to simplify querying.

1.1.3 Types

Q4. Explain briefly, various types of data structures.

Ans : **(Imp.)**

Here are various types of data structures, and the use and application of a particular type depend on the context in which the data structure has to be applied.

Types of Linear Data Structures

Types of linear data structures are given below:

Arrays

An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float or double.

The elements of array share the same variable name but each one carries a different index number known as subscript. The array can be one dimensional, two dimensional or multidimensional.

The individual elements of the array age are:

age[0], age[1], age[2], age[3],.... age[98], age[99]

Linked List

Linked list is a linear data structure which is used to maintain a list in the memory. It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointer to its adjacent node.

Stack

Stack is a linear list in which insertion and deletions are allowed only at one end, called top.

A stack is an abstract data type (ADT), can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: - piles of plates or deck of cards etc.

Queue

Queue is a linear list in which elements can be inserted only at one end called rear and deleted only at the other end called front.

It is an abstract data structure, similar to stack. Queue is opened at both end therefore it follows First-In-First-Out (FIFO) methodology for storing the data items.

Non Linear Data Structures

This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.

Types of Non Linear Data Structures are given below:

Trees

Trees are multilevel data structures with a hierarchical relationship among its elements known as nodes. The bottommost nodes in the herierchy are called leaf node while the topmost node is called root node. Each node contains pointers to point adjacent nodes.

Tree data structure is based on the parent-child relationship among the nodes. Each node in the tree can have more than one children except the leaf nodes whereas each node can have atmost one parent except the root node. Trees can be classfied into many categories which will be discussed later in this tutorial.

Graphs

Graphs can be defined as the pictorial representation of the set of elements (represented by vertices) connected by the links known as edges. A graph is different from tree in the sense that a graph can have cycle while the tree can not have the one.

1.2 ARRAYS

1.2.1 Abstract Data Types And The C++ Class

Q5. What is abstract data type?

Ans : **(Imp.)**

Abstract Data Type (ADT)

Abstract Data Type (ADT) is a data type that allows the programmer to use it without concentrating on the details of its implementation.

A class can be treated as an abstract data type by separating its specification from the implementation of its operations.

This separation between the specifications and the implementation can be obtained by,

(i) Considering and placing all the member variables in the private section of the class.

(ii) Placing all the needed operations in the public section and describing the ways of using these member functions.

(iii) Considering all the helping functions as private member functions and placing them in private section of the class.

Class class-name
Private 1. Member Variables 2. Helping Functions
Public Operations needed by the programmer

Table: Class as an ADT

The interface of an ADT specifies how to use it in our program and consists of the public member functions along with the accompanying comments that describes the way to use these public member-functions.

The implementation of an ADT specifies the way to realize an interface as a C++ code and consists of private members of a class and the definitions of both the private and public member functions.

To use an ADT in our program, the only thing we need to know is its interface, but not its implementation i.e., implementation is not required to write the main program.

Q6. Explain with an example, how class can be used as ADT.

Ans :

Abstract Data Type Model

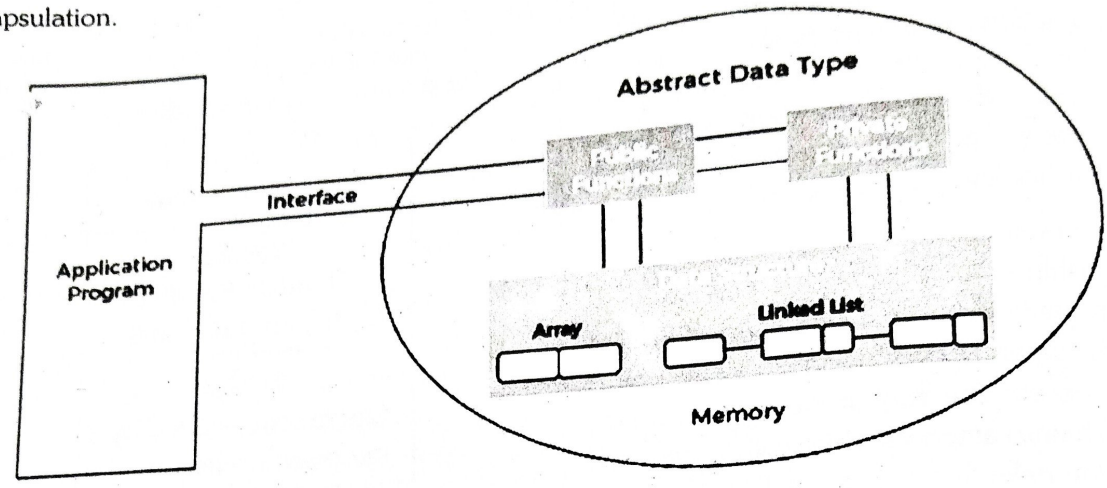
Before knowing about the abstract data type model, we should know about abstraction and encapsulation.

Abstraction

It is a technique of hiding the internal details from the user and only showing the necessary details to the user.

Encapsulation

It is a technique of combining the data and the member function in a single unit is known as encapsulation.



The above figure shows the ADT model. There are two types of models in the ADT model, i.e., the public function and the private function. The ADT model also contains the data structures that we are using in a program. In this model, first encapsulation is performed, i.e., all the data is wrapped in a single unit, i.e., ADT. Then, the abstraction is performed means showing the operations that can be performed on the data structure and what are the data structures that we are using in a program.

Let's understand the abstract data type with a real-world example

If we consider the smartphone. We look at the high specifications of the smartphone, such as:

- 4 GB RAM
- Snapdragon 2.2ghz processor
- 5 inch LCD screen
- Dual camera
- Android 8.0

The above specifications of the smartphone are the data, and we can also perform the following operations on the smartphone:

- **call():** We can call through the smartphone.
- **text():** We can text a message.
- **photo():** We can click a photo.
- **video():** We can also make a video.

The smartphone is an entity whose data or specifications and operations are given above. The abstract/logical view and operations are the abstract or logical views of a smartphone.

The implementation view of the above abstract/logical view is given below:

```
class Smartphone
{
    private:
    int ramSize;
    string processorName;
    float screenSize;
    int cameraCount;
    string androidVersion;
    public:
    void call();
    void text();
    void photo();
    void video();
}
```

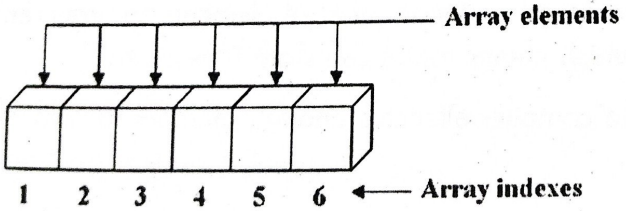
The above code is the implementation of the specifications and operations that can be performed on the smartphone. The implementation view can differ because the syntax of programming languages is different, but the abstract/logical view of the data structure would remain the same. Therefore, we can say that the abstract/logical view is independent of the implementation view.

1.2.2 Array As An Abstract Data Type

Q7. Explain, how Array can be used as ADT.

Ans : **(Imp.)**

The array is an abstract data type (ADT) that holds a collection of elements accessible by an index.



One-dimensional array with six elements

- The elements stored in an array can be anything from primitives types such as integers to more complex types like instances of classes.
- An element is stored in a given index and they can be retrieved at a later time by specifying the same index.
- The Array (ADT) have one property, they store and retrieve elements using an index. The array (ADT) is usually implemented by an Array (Data Structure).
- To fulfill the ADT an array must implement at least a way to retrieve the data from a given index and to store the data in a given index.

  set(i, v) → Sets the value of index i to v

  get(i) → Returns the value of index i in the array V

ADT-Operations

We can have a whole lot of different operations on the array we created, but we'lllimit ourselves to some basic ones.

- Max()
- Min()
- Search (num)
- Insert (i, num)
- Append(x)

1.2.3 Representation of Arrays

Q8. What is array? Give brief introduction about array.

Ans : **(Imp.)**

Definition

- Arrays are defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- Array is the simplest data structure where each data element can be randomly accessed by using its index number.
- For example, if we want to store the marks of a student in 6 subjects, then we don't need

to define different variable for the marks in different subject. instead of that, we can define an array which can store the marks in each subject at a the contiguous memory locations.

The array marks[10] defines the marks of the student in 10 different subjects where each subject marks are located at a particular subscript in the array i.e. marks[0] denotes the marks in first subject, marks[1] denotes the marks in 2nd subject and so on.

Properties of the Array

1. Each element is of same data type and carries a same size i.e. int = 4 bytes.
2. Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
3. Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of data element.

for example, in C language, the syntax of declaring an array is like following:

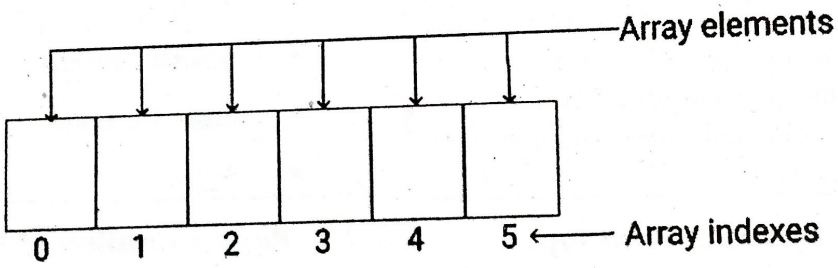
int arr[10]; **char** arr[10]; **float** arr[5]

Q9. Explain, how array can be represented.

Ans :

Types of Arrays and their Representation

Arrays and its representation is given below.



- **Array Index:** The location of an element in an array has an index, which identifies the element. Array index starts from 0.
- **Array element:** Items stored in an array is called an element. The elements can be accessed via its index.
- **Array Length:** The length of an array is defined based on the number of elements an array can store. In the above example, array length is 6 which means that it can store 6 elements.

When an array of size and type is declared, the compiler allocates enough memory to hold all elements of data.

For example, an array **face [10]** will have 10 elements with index starting from 0 to 9 and the memory allocated contiguously will be 20 bytes. The compiler knows the address of the first byte of the array only. Also, the address of the first byte is considered as the memory address for the whole array.

Normal variables (a1, a2, a3, ...) can be used when we have a small number of elements, but if we want to store a large number of elements, it becomes difficult to manage them with normal variables. Representing many elements with one variable name is the basic idea of arrays.

Let's try to understand this by taking one example. Assume, we can declare an array of size 10 in the following way.

int a[10];

Here a itself is a pointer which contains the memory location of the first element of the array. Now for accessing the first element, we will write a[0] which is internally decoded by the compiler as ***(a + 0)**.

In the same way, the second element can be accessed by **a[1]** or ***(a + 1)**. As **a** contains the address of the first element so for accessing the second element we have to add 1 to it. That's why here we have written ***(a +1)**. In an array, the index describes the offset from the first element, i.e. the distance from the first element.

Now let's assume that array indexing starts at 1 instead of 0. In this case for accessing the first element, we have to write **a[1]** which is internally decoded as ***(a + 1 – 1)**.

Observe here that we have to perform one extra operation i.e. subtraction by 1. This extra operation will greatly decrease the performance when the program is big. That's why to avoid this extra operation and improve the performance, array indexing starts at 0 and not at 1.

Array Operation

Now that we know the basic idea behind an array, let us now look at the various operations that can be performed on arrays.

- **Traverse:** Print all the elements in the array one by one.
- **Insertion:** Adds an element at the given index.
- **Deletion:** Deletes an element at the given index.
- **Search:** Searches an element in the array using the given index or the value.
- **Update:** Updates an element at the given index.

Q10. Explain different types of arrays with an example.

Ans :

Types of Arrays

The various types of arrays are as follows.

- One dimensional array
- Multi-dimensional array

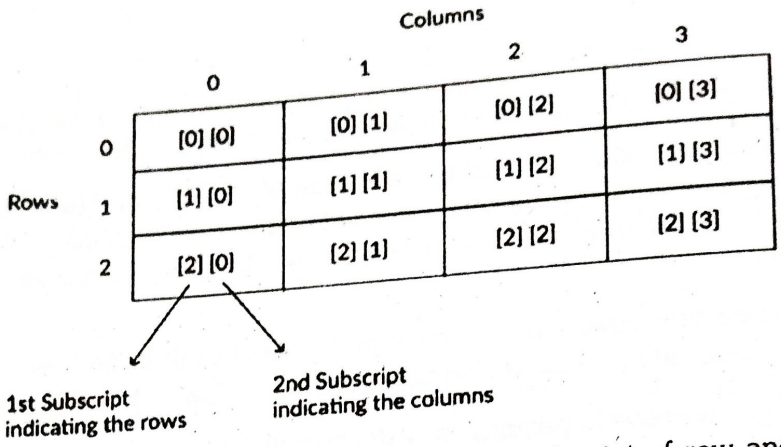
One-Dimensional Array

A one-dimensional array is also called a single dimensional array where the elements will be accessed in sequential order. This type of array will be accessed by the subscript of either a column or row index.

Multi-Dimensional Array

When the number of dimensions specified is more than one, then it is called as a multi-dimensional array. Multidimensional arrays include 2D arrays and 3D arrays.

Two-dimensional Array



A two-dimensional array will be accessed by using the subscript of row and column index. For traversing the two-dimensional array, the value of the rows and columns will be considered. In the two-dimensional array **face [3] [4],** the first index specifies the number of rows and the second index specifies the number of columns and the array can hold 12 elements (3 * 4).

Similarly, in a three-dimensional array, there will be three dimensions. The array **face [5] [10] [15]** can hold 750 elements (5 * 10 * 15).

Declaration/Initialization of Arrays

```
// A sample program for Array Declaration
#include <stdio.h>
int main()
{
int one_dim [10];      # declaration of 1D array
int two_dim [2][2];   #declaration of 2D array
int three_dim [2][3][4] = {
        { {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },
        { {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }
    }; #declaration of 3D array. Here the elements are also defined.return 0;
}
```

1.2.4 Matrices

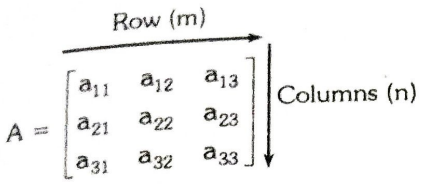
Q11. What is a matrix? Write , how a matrix can be represented in data structure.

Ans :

(Imp.)

A matrix can be defined as a two-dimensional array having 'm' rows and 'n' columns. A matrix with m rows and n columns is called m × n matrix. It is a set of numbers that are arranged in the horizontal or vertical lines of entries.

For example



It can also be called as double-dimensioned array.

Uses

- To represent class hierarchy using Boolean square matrix.
- For data encryption and decryption.
- To represent traffic flow and plumbing in a network.
- To implement graph theory of node representation.

In C/C++, we can define multidimensional arrays in simple words as an array of arrays. Data in multidimensional arrays are stored in tabular form (in row-major order).

The general form of declaring N-dimensional arrays:

data_type array_name[size1][size2]....[sizeN];

data_type: Type of data to be stored in the array.

Here data_type is valid C/C++ data type

array_name: Name of the array

size1, size2,... ,sizeN: Sizes of the dimensions

Q12. Write a c++ program to add two matrices.

Ans :

Add Two Matrices using Multi-dimensional Arrays

```
#include<iostream>
using namespace std;
int main()
{
   int r, c, a[100][100], b[100][100], sum[100][100], i, j;
   cout << "Enter number of rows (between 1 and 100): ";
   cin >> r;
   cout << "Enter number of columns (between 1 and 100): ";
   cin >> c;
   cout << endl << "Enter elements of 1st matrix: " << endl;
```

```
        // Storing elements of first matrix entered by user.
        for(i = 0; i < r; ++i)
          for(j = 0; j < c; ++j)
          {
            cout << "Enter element a" << i + 1 << j + 1 << " : ";
            cin >> a[i][j];
          }
        // Storing elements of second matrix entered by user.
        cout << endl << "Enter elements of 2nd matrix: " << endl;
        for(i = 0; i < r; ++i)
          for(j = 0; j < c; ++j)
          {
            cout << "Enter element b" << i + 1 << j + 1 << " : ";
            cin >> b[i][j];
          }
        // Adding Two matrices
        for(i = 0; i < r; ++i)
          for(j = 0; j < c; ++j)
            sum[i][j] = a[i][j] + b[i][j];
        // Displaying the resultant sum matrix.
        cout << endl << "Sum of two matrix is: " << endl;
        for(i = 0; i < r; ++i)
          for(j = 0; j < c; ++j)
          {
            cout << sum[i][j] << "  ";
            if(j == c - 1)
              cout << endl;
          }
      return 0;
    }
```

Output

```
Enter number of rows (between 1 and 100): 2
Enter number of columns (between 1 and 100): 2
```

```
Enter elements of 1st matrix:
Enter element a11: -4
Enter element a12: 5
Enter element a21: 6
Enter element a22: 8

Enter elements of 2nd matrix:
Enter element b11: 3
Enter element b12: -9
Enter element b21: 7
Enter element b22: 2
Sum of two matrix is:
-1  -4
13  10
```

1.2.5 Special Matrices

Q13. Write about various types of special matrices. **(Imp.)**

Ans :

Triangular, Symmetric, Diagonal

We have seen that a matrix is a block of entries or two dimensional data. The size of the matrix is given by the number of rows and the number of columns. If the two numbers are the same, we called such matrix a square matrix.

To square matrices we associate what we call the **main diagonal** (in short the diagonal). Indeed, consider the matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Its diagonal is given by the numbers a and d. For the matrix

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & k \end{pmatrix}$$

its diagonal consists of a, e, and k. In general, if A is a square matrix of order n and if a_{ij} is the number in the i^{th}-row and j^{th}-colum, then the diagonal is given by the numbers a_{ii}, for $i=1,..,n$.

The diagonal of a square matrix helps define two type of matrices: upper-triangular and lower-triangular. Indeed, the diagonal subdivides the matrix into two blocks: one above the diagonal and the other one below it. If the lower-block consists of zeros, we call such a matrix upper-triangular. If the upper-block consists of zeros, we call such a matrix lower-triangular.

For example, the matrices.

$$\begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ 0 & e & f \\ 0 & 0 & k \end{pmatrix}$$

are upper-triangular, while the matrices

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } \begin{pmatrix} a & 0 & 0 \\ d & e & 0 \\ g & h & k \end{pmatrix}$$

are lower-triangular. Now consider the two matrices.

$$A = \begin{pmatrix} a & 0 & 0 \\ d & e & 0 \\ g & h & k \end{pmatrix} \text{ and } B = \begin{pmatrix} a & d & g \\ 0 & e & h \\ 0 & 0 & k \end{pmatrix}$$

The matrices A and B are triangular. But there is something special about these two matrices. Indeed, as you can see if you reflect the matrix A about the diagonal, you get the matrix B. This operation is called the transpose operation. Indeed, let A be a nxm matrix defined by the numbers a_{ij}, then the transpose of A, denoted A^T is the mxn matrix defined by the numbers b_{ij} where $b_{ij} = a_{ji}$. For example, for the matrix.

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & k \\ t & r & s \end{pmatrix}$$

We have

$$A^T = \begin{pmatrix} a & d & g & t \\ b & e & h & r \\ c & f & k & s \end{pmatrix}$$

Properties of the Transpose operation

If X and Y are mxn matrices and Z is an nxk matrix, then

1. $(X+Y)^T = X^T + Y^T$
2. $(XZ)^T = Z^T X^T$
3. $(X^T)^T = X$

A symmetric matrix is a matrix equal to its transpose. So a symmetric matrix must be a square matrix. For example, the matrices

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

are symmetric matrices. In particular a symmetric matrix of order n, contains at most $\frac{n(n+1)}{2}$ different numbers.

A diagonal matrix is a symmetric matrix with all of its entries equal to zero except may be the ones on the diagonal. So a diagonal matrix has at most n different numbers other than 0. For example, the matrices.

$$\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \text{ and } \begin{pmatrix} a & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & b \end{pmatrix}$$

are diagonal matrices. Identity matrices are examples of diagonal matrices. Diagonal matrices play a crucial role in matrix theory. We will see this later on.

Example. Consider the diagonal matrix

$$A = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$

1.2.6 Sparse Matrices

Q14. What is Sparse Matrix? **(Imp.)**

Ans :

In computer programming, a matrix can be defined with a 2-dimensional array. Any array with 'm' columns and 'n' rows represent a m X n matrix. There may be a situation in which a matrix contains more number of ZERO values than NON-ZERO values. Such matrix is known as sparse matrix.

When a sparse matrix is represented with a 2-dimensional array, we waste a lot of space to represent that matrix. For example, consider a matrix of size 100 X 100 containing only 10 non-zero elements. In this matrix, only 10 spaces are filled with non-zero values and remaining spaces of the matrix are filled with zero. That means, totally we allocate 100 X 100 X 2 = 20000 bytes of space to store this integer matrix. And to access these 10 non-zero elements we have to make scanning for 10000 times. To make it simple we use the following sparse matrix representation.

Q15. Explain various ways to represent sparse matrix.

Ans : **(Imp.)**

Sparse Matrix Representations

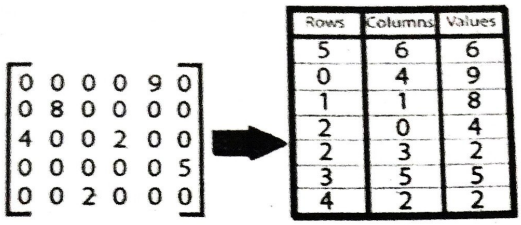
A sparse matrix can be represented by using TWO representations, those are as follows...

1. Triplet Representation (Array Representation)
2. Linked Representation

Triplet Representation (Array Representation)

In this representation, we consider only non-zero values along with their row and column index values. In this representation, the 0^{th} row stores the total number of rows, total number of columns and the total number of non-zero values in the sparse matrix.

For example, consider a matrix of size 5 X 6 containing 6 number of non-zero values. This matrix can be represented as shown in the image...



Rows	Columns	Values
5	6	6
0	4	9
1	1	8
2	0	4
2	3	2
3	5	5
4	2	2

In above example matrix, there are only 6 non-zero elements (those are 9, 8, 4, 2, 5 & 2) and matrix size is 5 X 6. We represent this matrix as shown in the above image. Here the first row in the right side table is filled with values 5, 6 & 6 which indicates that it is a sparse matrix with 5 rows, 6 columns & 6 non-zero values. The second row is filled with 0, 4, & 9 which indicates the non-zero value 9 is at the 0th-row 4th column in the Sparse matrix. In the same way, the remaining non-zero values also follow a similar pattern.

Implementation of Array Representation of Sparse Matrix using C++

```
#include<iostream>
using namespace std;
int main()
{
  // sparse matrix of class 5x6 with 6 non-zero values
  int sparseMatrix[5][6] =
  {
    {0 , 0 , 0 , 0 , 9, 0 },
    {0 , 8 , 0 , 0 , 0, 0 },
    {4 , 0 , 0 , 2 , 0, 0 },
    {0 , 0 , 0 , 0 , 0, 5 },
    {0 , 0 , 2 , 0 , 0, 0 }
  };

// Finding total non-zero values in the sparse matrix
  int size = 0;
  for (int row = 0; row < 5; row++)
    for (int column = 0; column < 6; column++)
      if (sparseMatrix[row][column] != 0)
        size++;
  // Defining result Matrix
  int resultMatrix[3][size];
  // Generating result matrix
```

```
int k = 0;
for (int row = 0; row < 5; row++)
   for (int column = 0; column < 6; column++)
      if (sparseMatrix[row][column] != 0)
      {
         resultMatrix[0][k] = row;
         resultMatrix[1][k] = column;
         resultMatrix[2][k] = sparseMatrix[row][column];
        k++;
      }

// Displaying result matrix
cout<<"Triplet Representation : "<<endl;
for (int row=0; row<3; row++)
{
   for (int column = 0; column<size; column++)
      cout<<resultMatrix[row][column]<<" ";

   cout<<endl;
}
return 0;
}
```

1.2.7 Strings

Q16. What is string? Explain various ways to represent a string.

Ans :

(Imp.)

In C++, the string can be represented as an array of characters or using string class that is supported by C++. Each string or array element is terminated by a null character. Representing strings using a character array is directly taken from the 'C' language as there is no string type in C.

Implementation of String Arrays

In C++, strings can be represented using three ways.

1. **Using Two-dimensional Character Arrays**

   This representation uses the two-dimensional arrays where each element is the intersection of a row and column number and represents a string

2. **Using String Keyword**

   We can also use the string keyword of C++ to declare and define string arrays.

3. Using STL Vectors

We can use STL vectors wherein each element of a vector is a string.

Now, let's discuss each of the above methods and also see the programming examples for each representation.

Using Two-dimensional Character Arrays

String arrays or an array of strings can be represented using a special form of two-dimensional arrays. In this representation, we use a two-dimensional array of type characters to represent a string.

The first dimension specifies the number of elements i.e. strings in that array and the second dimension specifies the maximum length of each element in the array.

So we can use a general representation as shown below.

char "stringarrayname" ["number of strings"] ["maximum length of the string"]

For example, consider the following declaration:

```
char string_array[10] [20];
```

The above declaration declares an array of strings named 'string_array' which has 10 elements and the length of each element is not more than 20.

We can declare and initialize an array of animals using strings in the following way:

```
char animals [5] [10] = {"Lion", "Tiger", "Deer", "Ape", "Kangaroo"};
```

Let us see a programming example using the concept of two-dimensional character arrays to better understand the concept.

```
#include <iostream>
usingnamespacestd;
intmain()
{
    charstrArray[5] [6] = {"one", "two", "three", "four", "five"};
    cout<<"String array is as follows:"<<endl;
    for(inti=0;i<5;i++)
    {
        cout<<"Element "<<i<<"= "<<strArray[i]<<endl;
    }
  return0;
}
```

Output:

```
String array is as follows:
Element 0 = one
Element 1 = two
Element 2 = three
Element 3 = four
Element 4 = five
```

In the above program, we have declared an array of strings called strArray of size 5 with the max length of each element as 10. In the program, we initiate a for loop to display each element of the array. Note that we just need to access the array using the first dimension to display the element.

Easy access to elements is one of the major advantages of 2-D arrays. They are indeed simple to program.

The major drawback of this type of representation is, both the dimensions of array i.e. number of elements and the maximum length of the element are fixed and cannot be changed as we want.

Secondly, we specify the maximum length of each element as the second dimension during the declaration of the array. If the string length is specified as 100, and we have all the elements that are lesser in length, then the memory is wasted.

Using string Keyword

In this, we use the keyword 'string' in C++ to declare an array of strings. Unlike character arrays, here we have only 1D array. The sole dimension specifies the number of strings in the array.

The general syntax for an array of strings declaration using the string keyword is given below:

string "array name" ["number of strings"];

Note that we do not specify the maximum length of string here. This means that there is no limitation on the length of the array elements.

As an example, we can declare an array of color names in the following way.

```
string colors[5];
```

We can further initialize this array as shown below:

```
string colors[5] = {"Red", "Green", "Blue", "Orange", "Brown"};
```

Given below is a C++ program to understand the string keyword and its usage in an array of strings.

```
#include <iostream>
usingnamespacestd;
intmain()
{
    string numArray[5] = {"one", "two", "three", "four", "five"};
    cout<<"String array is as follows:"<<endl;
    for(inti=0;i<5;i++)
    {
        cout<<"Element "<<i<<"= "<<numArray[i]<<endl;
    }

return0;
}
```

Output:

String array is as follows:

Element 0 = one

Element 1 = two

Element 2 = three

Element 3 = four

Element 4 = five

Using STL Vectors

We can also use STL vectors for declaring and defining dynamic arrays. Thus to define an array of strings we can have an STL vector of type string.

This declaration of an array of strings using vector is shown below:

vector<string> "stringarray_Name";

Referring to the above declaration, we can declare a vector "subjects" in the following way:

vector<string> mysubjects;

Note that we can assign elements to the vector by using the "push_back" method or any other STL vector methods.

Given below is a programming example using C++ to demonstrate the usage of the STL vector to represent an array of strings.

```
#include <iostream>
#include <vector>
usingnamespacestd;
intmain()
{
    vector <string> myNumbers;

    myNumbers.push_back("one");
    myNumbers.push_back("two");
    myNumbers.push_back("three");
    myNumbers.push_back("four");
    myNumbers.push_back("five");

    cout<<"String array is as follows:"<<endl;
    for(inti=0;i<5;i++)
    {
        cout<<"Element "<<i<<"= "<<myNumbers[i]<<endl;
```

```
        }

    return0;
}
```

Output:

String array is as follows:

Element 0 = one

Element 1 = two

Element 2 = three

Element 3 = four

Element 4 = five

Short Question and Answers

1. Define data structure.

Ans :

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage.

In simple language, Data Structures are structure programmed to store ordered data, so that various operations can be performed on it easily.

2. Types of data structures.

Ans :

Types of linear data structures are given below:

Arrays

An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float or double.

The elements of array share the same variable name but each one carries a different index number known as subscript. The array can be one dimensional, two dimensional or multidimensional.

The individual elements of the array age are:

age[0], age[1], age[2], age[3],..... age[98], age[99]

Linked List

Linked list is a linear data structure which is used to maintain a list in the memory. It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointer to its adjacent node.

Stack

Stack is a linear list in which insertion and deletions are allowed only at one end, called top.

A stack is an abstract data type (ADT), can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: - piles of plates or deck of cards etc.

Queue

Queue is a linear list in which elements can be inserted only at one end called rear and deleted only at the other end called front.

It is an abstract data structure, similar to stack. Queue is opened at both end therefore it follows First-In-First-Out (FIFO) methodology for storing the data items.

Non Linear Data Structures

This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.

Types of Non Linear Data Structures are given below:

Trees

Trees are multilevel data structures with a hierarchical relationship among its elements known as nodes. The bottommost nodes in the herierchy are called leaf node while the topmost node is called root node. Each node contains pointers to point adjacent nodes.

Tree data structure is based on the parent-child relationship among the nodes. Each node in the tree can have more than one children except the leaf nodes whereas each node can have atmost one parent except the root node. Trees can be classfied into many categories which will be discussed later in this tutorial.

Graphs

Graphs can be defined as the pictorial representation of the set of elements (represented by vertices) connected by the links known as edges. A graph is different from tree in the sense that a graph can have cycle while the tree can not have the one.

3. Abstract Data Type

Ans :

Abstract Data Type (ADT) is a data type that allows the programmer to use it without concentrating on the details of its implementation.

A class can be treated as an abstract data type by separating its specification from the implementation of its operations.

This separation between the specifications and the implementation can be obtained by,

(i) Considering and placing all the member variables in the private section of the class.

(ii) Placing all the needed operations in the public section and describing the ways of using these member functions.

(iii) Considering all the helping functions as private member functions and placing them in private section of the class.

4. What is array?

Ans :

- Arrays are defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- Array is the simplest data structure where each data element can be randomly accessed by using its index number.
- For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variable for the marks in different subject. instead of that, we can define an array which can store the marks in each subject at a the contiguous memory locations.

The array marks[10] defines the marks of the student in 10 different subjects where each subject marks are located at a particular subscript in the array i.e. marks[0] denotes the marks in first subject, marks[1] denotes the marks in 2nd subject and so on.

5. Types of Arrays

Ans :

The various types of arrays are as follows.

- One dimensional array
- Multi-dimensional array

One-Dimensional Array

A one-dimensional array is also called a single dimensional array where the elements will be accessed in sequential order. This type of array will be accessed by the subscript of either a column or row index.

Multi-Dimensional Array

When the number of dimensions specified is more than one, then it is called as a multi-dimensional array. Multidimensional arrays include 2D arrays and 3D arrays.

6. What is a matrix?

Ans :

A matrix can be defined as a two-dimensional array having 'm' rows and 'n' columns. A matrix with m rows and n columns is called m × n matrix. It is a set of numbers that are arranged in the horizontal or vertical lines of entries.

For example

Row (m)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$ Columns (n)

It can also be called as double-dimensioned array.

Uses

- To represent class hierarchy using Boolean square matrix.
- For data encryption and decryption.
- To represent traffic flow and plumbing in a network.
- To implement graph theory of node representation.

In C/C++, we can define multidimensional arrays in simple words as an array of arrays. Data in multidimensional arrays are stored in tabular form (in row-major order).

The general form of declaring N-dimensional arrays:

data_type array_name[size1] [size2].... [sizeN];

data_type: Type of data to be stored in the array.

Here data_type is valid C/C++ data type

array_name: Name of the array

size1, size2,... ,sizeN: Sizes of the dimensions.

7. What is Sparse Matrix?

Ans :

In computer programming, a matrix can be defined with a 2-dimensional array. Any array with 'm' columns and 'n' rows represent a m X n matrix. There may be a situation in which a matrix contains more number of ZERO values than NON-ZERO values. Such matrix is known as sparse matrix.

When a sparse matrix is represented with a 2-dimensional array, we waste a lot of space to represent that matrix. For example, consider a matrix of size 100 X 100 containing only 10 non-zero elements. In this matrix, only 10 spaces are filled with non-zero values and remaining spaces of the matrix are filled with zero. That means, totally we allocate 100 X 100 X 2 = 20000 bytes of space to store this integer matrix. And to access these 10 non-zero elements we have to make scanning for 10000 times. To make it simple we use the following sparse matrix representation.

8. What is string?

Ans :

In C++, the string can be represented as an array of characters or using string class that is supported by C++. Each string or array element is terminated by a null character. Representing strings using a character array is directly taken from the 'C' language as there is no string type in C.

Implementation of String Arrays

In C++, strings can be represented using three ways.

1. **Using Two-dimensional Character Arrays**

   This representation uses the two-dimensional arrays where each element is the intersection of a row and column number and represents a string

2. **Using String Keyword**

   We can also use the string keyword of C++ to declare and define string arrays.

3. **Using STL Vectors**

   We can use STL vectors wherein each element of a vector is a string.

Now, let's discuss each of the above methods and also see the programming examples for each representation.

9. Importance of data structures.

Ans :

i) **Storing Data:** Data structures are used for efficient data persistence, such as specifying the collection of attributes and corresponding structures used to store records in a database management system.

ii) **Managing Resources and Services:** Core operating system (OS) resources and services are enabled through the use of data structures such as linked lists for memory allocation, file directory management and file structure trees, as well as process scheduling queues.

iii) **Data Exchange:** Data structures define the organization of information shared between applications, such as TCP/IP packets.

iv) **Ordering and Sorting:** Data structures such as binary search trees -also known as an ordered or sorted binary tree - provide efficient methods of sorting objects, such as character strings used as tags. With data structures such as priority queues, programmers can manage items organized according to a specific priority.

v) **Indexing**: Even more sophisticated data structures such as B-trees are used to index objects, such as those stored in a database.

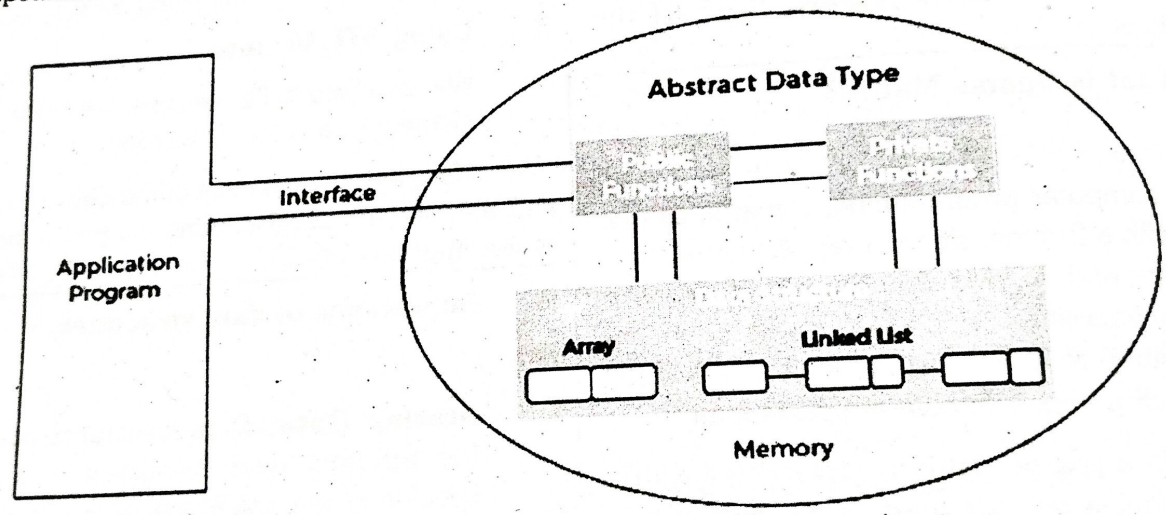
vi) **Searching:** Indexes created using binary search trees, B-trees or hash tables speed the ability to find a specific sought-after item.

vii) **Scalability:** Big data applications use data structures for allocating and managing data storage across distributed storage locations, ensuring scalability and performance. Certain big data programming environments - such as Apache Spark - provide data structures that mirror the underlying structure of database records to simplify querying.

10. Encapsulation

Ans :

It is a technique of combining the data and the member function in a single unit is known as encapsulation.



The above figure shows the ADT model. There are two types of models in the ADT model, i.e., the public function and the private function. The ADT model also contains the data structures that we are using in a program. In this model, first encapsulation is performed, i.e., all the data is wrapped in a single unit, i.e., ADT. Then, the abstraction is performed means showing the operations that can be performed on the data structure and what are the data structures that we are using in a program.

Choose the Correct Answers

1. Elements in an array are accessed _____. [a]
   (a) randomly (b) sequentially
   (c) exponentially (d) logarithmically

2. Which of the following is a linear data structure? [a]
   (a) Array (b) AVL Trees
   (c) Binary Trees (d) Graphs

3. What is the disadvantage of array data structure? [a]
   (a) The amount of memory to be allocated should be known beforehand.
   (b) Elements of an array can be accessed in constant time.
   (c) Elements are stored in contiguous memory blocks.
   (d) Multiple other data structures can be implemented using arrays.

4. How is the 2nd element in an array accessed based on pointer notation? [b]
   (a) *a + 2 (b) *(a + 2)
   (c) *(*a + 2) (d) &(a + 2)

5. Index of an array starts with? [d]
   (a) 1 (b) 2
   (c) 3 (d) 0

6. Data types of the array include? [d]
   (a) int (b) float
   (c) char, double, struct (d) All of these

7. int a[5]={1,3,2}. What will be the value of a[4]? [a]
   (a) 3 (b) 2
   (c) 0 (d) Garbage value

8. Which matrix has most of the elements (not all) as Zero? [c]
   (a) Identity Matrix (b) Unit Matrix
   (c) Sparse Matrix (d) Zero Matrix

9. Which of the following is not the method to represent Sparse Matrix? [d]
   (a) Dictionary of Keys (b) Linked List
   (c) Array (d) Heap

10. Which of the following is the disadvantage of sparse matrices over normal matrices? [d]
    (a) Size (b) Speed
    (c) Easily compressible (d) Algorithm complexity

Fill in the Blanks

1. _____ is a data type that allows the programmer to use it without concentrating on the details of its implementation.
2. Container of objects of similar types is also called as _____
3. Size of an Array does not required, when _____
4. _____ Operator is used to allocates memory to array variable in c++
5. The matrix contains m rows and n columns. The matrix is called Sparse Matrix if _____
6. _____ Array is defined as an array of characters
7. We use the _____ keyword in C++ to declare an array of strings.
8. A _____ array the elements will be accessed in sequential order.
9. Elements of the array can be randomly accessed with the given _____ and _____
10. To define an array of strings we can have an _____of type string.

Answers

1. Abstract Data Type (ADT)
2. An array
3. Initialization is part of the definition
4. New
5. Total number of Zero elements > (m*n)/2
6. Character array
7. 'String'
8. One-dimensional
9. Base address and the size of data element
10. STL vector