ABSTRACT DATA TYPES (ADTS)

- An abstract data type is an abstraction of a data structure that provides only the interface to which the data structure must adhere. The interface does not give any specific details about something should be implemented or in what programminglanguage.
- In other words, we can say that abstract data types are the entities that are definitions of data and operations but do not have implementation details. In this case, we know the data that we are storing and the operations that can be performed on the data, but we don't know about the implementation details.
- The reason for not having implementation details is that every programming language has a different implementation strategy for example; a C data structure is implemented using structures while a C++ data structure is implemented using objects and classes.

3.1.1 Abstract data type model

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.

Figure 3.1 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

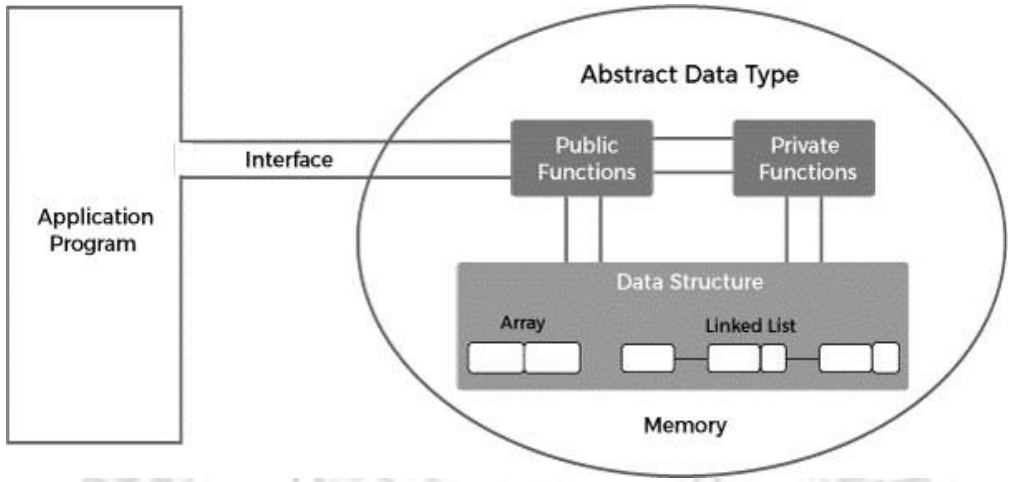

Fig 3.1 Abstract Data Type Model

Example 3.1

Suppose we have an index array of size 4. We have an index location starting from 0, 1, 2, 3. Array is a data structure where the elements are stored in a contiguous location. The memory address of the first element is 1000, second element is 1004, third element is 1008, and the fourth element is 1012. Since it is of integer type so it will occupy 4 bytes and the difference between the addresses of each element is 4 bytes. The values stored in an array are 10, 20, 30 and 40. These values, index positions and the memory addresses are the implementations.

The abstract or logical view of the integer array can be stated as:

- It stores a set of elements of integer type.
- It reads the elements by position, i.e., index.
- It modifies the elements by index
- It performs sorting

LIST ADT

- The list can be defined as an abstract data type in which the elements are stored in an ordered manner for easier and efficient retrieval of the elements. List Data Structure allows repetition that means a single piece of data can occur more than once in a list.
- In the case of multiple entries of the same data, each entry of that repeating

datais considered as a distinct item or entry. It is very much similar to the array but the major difference between the array and the list data structure is that array stores only homogenous data in them whereas the list (in some programming languages) can store heterogeneous data items in its object. List Data Structure isalso known as a sequence.

- The list can be called Dynamic size arrays, which means their size increased as we go on adding data in them and we need not to pre-define a static size for the list

For example,

numbers = [1, 2, 3, 4, 5]

- In this example, 'numbers' is the name of the List Data Structure object and it hasfive items stored in it. In the object named numbers, we have stored all the elements of numeric type. In the list, the indexing starts from zero, which means if we want to access or retrieve the first element of this list then we need to use index zero and similarly whenever we want to access any element from this list named numbers. In other words, we can say that element 1 is on the index 0 and element 2 is on index 1 and similarly for further all elements.

Mixed_data = [205, ‘Mathu’, 8.56]

- In this second example, mixed_data is the name of the list object that stores the data of different types. In the mixed_data list, we have stored data of three types, first one is the integer type which is id '205', after the integer data we have storeda string type data having the value ‘Mathu’ stored at index 1 and at last the indexvalue 2, we have stored a float type data having the value '8.56'.
- To access the elements of the mixed_data list, we need to follow the same approach as defined in the previous example.
- And we can add more data to these defined List objects and that will get appended at the last of the list. For example, if we add another data in the mixed_data list, it will get appended after the float value object having value '8.56'. And we can add repeating values to these list-objects.

3.2.1 Operations on the List Data Structure

Add or Insert Operation:

In the Add or Insert operation, a new item (of any data type) is added in the List Data Structure or Sequence object.

Replace or reassign Operation:

In the Replace or reassign operation, the already existing value in the List object is changed or modified. In other words, a new value is added at that particular index of the already existing value.

Delete or remove Operation:

In the Delete or remove operation, the already present element is deleted or removed from the Dictionary or associative array object.

Find or Lookup or Search Operation:

In the Find or Lookup operation, the element stored in that List Data Structure orSequence object is fetched.

ARRAY-BASED IMPLEMENTATION

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

Properties of array

- Each element in an array is of the same data type and carries the same size that is4 bytes.
- Elements in the array are stored at contiguous memory locations from which the first element is stored at the smallest memory location.
- 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 the data element.

Representation of an array

- Array can be represented in various ways in different programming languages. As an illustration, let's see the declaration of array in C language

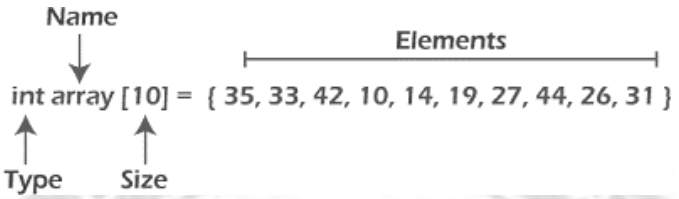

Fig. 3.2: Illustration of an Array

As per the above illustration of array, there are some of the following important points -

- Index starts with 0.
- The array's length is 10, which means we can store 10 elements.
- Each element in the array can be accessed via its index.

Memory allocation of an array

- Data elements of an array are stored at contiguous locations in the main memory. The name of the array represents the base address or the address of the first element in the main memory. Each element of the array is represented by proper indexing.
- We can define the indexing of an array in the below ways
  - 0 (zero-based indexing): The first element of the array will be arr[0].
  - 1 (one-based indexing): The first element of the array will be arr[1].
  - n (n - based indexing): The first element of the array can reside at any random index number
- Fig 3.3 shows the memory allocation of an array arr of size 5. The array follows a 0-based indexing approach. The base address of the array is 100 bytes. It is the address of arr[0]. Here, the size of the data type used is 4 bytes; therefore, each element will take 4 bytes in the memory.

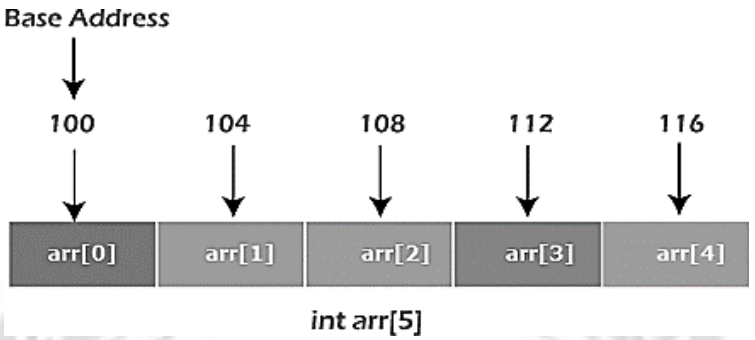

Fig. 3.3: Memory allocation of an array

Access an element from the array

The information given below are required to access any random element from the array

- Base Address of the array.
- Size of an element in bytes.
- Type of indexing, array follows.

The formula to calculate the address to access an array element

Byte address of element A[i] = base address + size * (i - first index)

Here, size represents the memory taken by the primitive data types. As an instance, int takes 2 bytes, float takes 4 bytes of memory space in C programming.

Example 3.2

Suppose an array, A[-10..... +2] having Base address (BA) = 999 and size of an element = 2 bytes, find the location of A[-1].L(A[-1]) = 999 + 2 x [(-1) - (-10)]

= 999 + 18

= 1017

Basic operations of an Array

Basic operations supported in the array

- Traversal - This operation is used to print the elements of the array.
- Insertion - It is used to add an element at a particular index.
- Deletion - It is used to delete an element from a particular index.

- Search - It is used to search an element using the given index or by the value.
- Update - It updates an element at a particular index.

Complexity of Array operations

Time and space complexity of various array operations are described below.

Time Complexity

Operation	Average Case	Worst Case
Access	O(1)	O(1)
Search	O(n)	O(n)
Insertion	O(n)	O(n)
Deletion	O(n)	O(n)

Space Complexity

In array, space complexity for worst case is O(n).

Limitations of Array

- The size of the array must be known in advance before using it in the program.
- Increasing the size of the array is a time taking process. It is almost impossible to expand the size of the array at run time.
- All the elements in the array need to be contiguously stored in the memory. Inserting an element in the array needs shifting of all its predecessors.

Advantages of Array

- Arrays are good for storing multiple values in a single variable.
- Traversing an array is a very simple process; we just need to increment the base address of the array in order to visit each element one by one.
- Any element in the array can be directly accessed by using the index.
- Sorting and searching a value in an array is easier.
- Arrays are best to process multiple values quickly and easily.

Disadvantages of Array

- Array is homogenous. It means that the elements with similar data type can be stored in it.