**Abstract Data Types**

**I. Introduction**

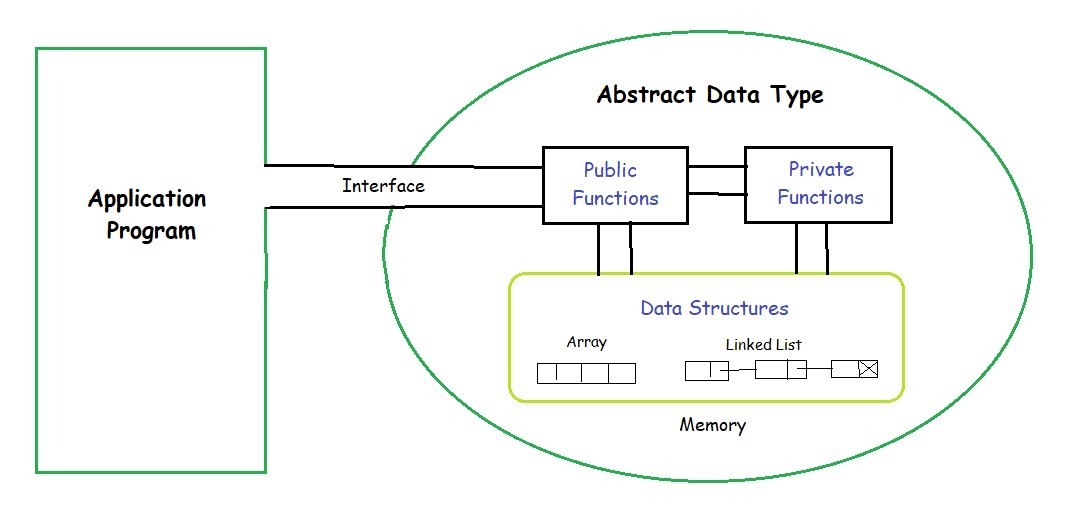
Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.

It is called “abstract” because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as abstraction.

Abstract data types include:

* Values – V
* Operations - O



The user of data type does not need to know how that data type is implemented. Think of ADT as a black box which hides the inner structure and design of the data type.

For example, we will describe an ADT that is a singly linked list used to manage some student data.



**II. Some Abstract Data Types**

**1. Array**

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together This makes it easier to calculate the position of each element by simply adding an offset to a base value. The memory location of the first element of the array.



**2. Linked Lists**

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location. The elements are linked using pointers.



**3. Stack**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

Chart

Description automatically generated

Mainly the following three basic operations are performed in the stack:

* **Push**: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
* **Pop**: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
* **Peek** or **Top**: Returns top element of the stack.
* **isEmpty**: Returns true if the stack is empty, else false.

Example to illustrate for stack: We’re going arrange the stack of books

Push operation: Now we put books one by one on the bookshelf.



Peek and Pop operations: Assumption that we want to take English book to read



**4. Queue**

Like Stack, Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

A good example of the queue is any queue of consumers for a resource where the consumer that came first is served first.

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Chart, diagram

Description automatically generated with medium confidence

Mainly the following four basic operations are performed on queue:

* **Enqueue**: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
* **Dequeue**: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
* **Front**: Get the front item from the queue.
* **Rear**: Get the last item from the queue.



The image above is a real example of a queue, first come first served.

**Why Abstract Data Type became a necessity?**

* Earlier if a programmer wanted to read a file, the whole code was written to read the physical file device. So that is how Abstract Data Type (ADT) came into existence.
* The code to read a file was written and placed in a library and made available for everyone’s use. This concept of ADT is being used in the modern languages nowadays.