ADT (Abstract Data Type)

An Abstract Data Type (ADT) consists of data type together with a set of operations, which define how the type may be manipulated.

ADT exists conceptually and concentrate on the mathematical properties of the data type ignoring implementation constraints and details.

The advantages offered by ADT include:

- Modularity
- Precise specifications
- Information hiding
- Simplicity
- Integrity
- Implementation independence

Stack as ADT

Stack can be defined as ADT:

i) Finite sequence of elements
ii) Operations on the elements like:
- CreateEmptyStack (S): Create or make stack S be an empty stack.
- Push (S, x): Insert x at one end of the stack, called its top of the stack.
- Pop (S): If stack S is not empty, then delete the element at its top.
- Top (S)/Peek(S): If stack S is not empty, then retrieve the element at its top.
- IsFull (S): Determine whether stack S is full or not. Return true if S is full; return false otherwise.
- IsEmpty (S): Determine whether stack S is empty or not. Return true if S is empty stack; return false otherwise.

Queue as ADT

Queue can be defined as ADT:

i) Finite sequence of elements
ii) Operations on the elements like:
- MakeEmpty (q): To make q as an empty queue.
- IsEmpty (q): To check whether the queue q is empty or not. Return how if q is empty, return false otherwise.
- IsFull (q): To check whether the queue (q) is full or not. Return true if q is full, return false otherwise.
- Enqueue (q, x): To insert an item x at the rear of the queue, if and only if q is not full.
- Dequeue (q): To delete an item from the front of queue (q), if and only if q is not empty.
- Traverse (q): To read entire queue, i.e. to display the content of the queue.

Linked list as an ADT

Linked list can be defined as ADT:

i) Finite sequence of elements
ii) Operations on the elements like:
- Create (): Create or make a node.
- Insert (x): Insert x to linked list.
- Delete (): If linked list is not empty then delete given node.
- Traverse (): Display all of the nodes of given linked list.
- IsEmpty (): Determine whether linked list is empty of not. Return true if it is empty; return false otherwise.
- Find () or Search (): Find out given node from linked list.
- Count (): Count number of nodes of given linked list.
- Free (): Release memory space of given node of linked list.

Graph as an ADT

i) Finite sequence of elements
ii) Operations on the elements like:
- Graph () creates a new, empty graph.
- addVertex (vert) adds an instance of Vertex to the graph.
- addEdge (fromVert, toVert) adds a new, directed edge to the graph that connects two vertices.
- addEdge (fromVert, toVert, weight) adds a new, weighted, directed edge to the graph that connects two vertices.
- getVertex (vertKey) finds the vertex in the graph name vertKey.
- getVertices () returns the list of all vertices in the graph.