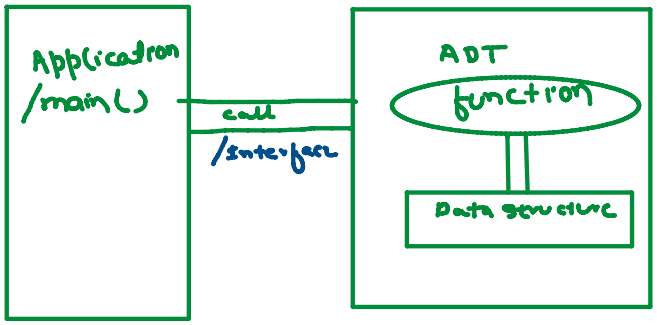
ABSTRACT DATA TYPES (ADT)

: ADTs are like user defined data types which defines operations on values using functions without specifying what is there inside the functions and how the operations are performed

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.



⇒ The program which uses data structure is called a client program.

→ It has access to the ADT i.e interface

• The program which implements the data structure is known as the Implementation

Think of ADT as a black box which hide the inner structure and design of the data type from the user.

Example: Stack ADT

A stack consists of elements of same type arranged in sequential order. In which an element may be inserted or deleted only at one end, called top of the stack.

Operations:

- an element into the stack

Operations:

PUSH() – Insert an element into the stack

POP() – Delete an element into the stack

IsEmpty() → checks if stack is empty

IsFull() → checks if stack is full;

Peek() → Return the element at the top of the stack without removing it, if the stack is not empty.

ADVANTAGE:

① Abstraction:
Let say, if someone wants to use the stack in the prg, then he can simply use push and and pop operations without knowing its implementation.

② Reusability: one implementation can be used by multiple client programs.

③ Memory utilization: only those operation (prg) bring in memory which required.

④ This can be used by any type of user (need not to be expert in DS)