Good link:
https://opendsa-server.cs.vt.edu/ODSA/Books/CS3/html/ADT.html

type:

.A type is a collection of values.
.for example boolean type consists of values *true* or *false*.
.The integers also form a type. An integer is a ***simple type*** because its
Values contains no subparts.

data type:

.A data type is type together with a collection of operations to manuplate the type.
For example:
An integer variable is a member of the integer data type.
Addition is an example of an operation on the integer data type.

*.SRC:Mastering Data Structures Through C Language By J. B. Dixit
*.when we consider a primitive type we are actually referring to two things:
(1).a data item with certain characteristics and
(2).the permissible operations on that data.

List!!!
there are two traditional implementations for the list data type: the linked list and the array-based list.

Data structure:

.Data Structures Definition: In computer science, a data structure is a *data organization*, *management* and *storage format* that enables efficient Access and modification.

What is abstract data type?
ans).
*.ADTs are entities that are definitions of data and operations but do not have implementation details.

*eg:
.Abstract data types (pid_t, size_t)
.a **Tree** is a widely used **abstract data type** (ADT)

abstraction:

***.abstract:** నైరూప్య,సంక్షేపించు, సంగ్రహించు

*.abstract is an overview or summary of something, without considering the minute details.
*.***what is abstraction?***. in computer science,abstraction is the technique for managing the complexity.
*.in other words, abstraction can be told as the level of complexity with which the user interfaces. abstraction do provides comforts to the user becoz it hides the complex details beneath it.

Concept of Abstraction:
*Allows us to consider the **high-level characterstics** of something **without Getting bogged down in the details**.

For example:
**process abstraction* in procedural programming like C, we can use (pre-defined) functions without concern how they really works inside.

Data Abstraction:
.We know what a data type can do
.How it is done is hidden

Abstract Data Type (ADT):
.Defines a particular data structure in terms of data and operations.
.Offers an interface of the objects (instances of an ADT)

An ADT consists of
.Declaration of data
.Declaration of operations
.Encapsulations of Data and Opertions: data is hidden from user and can Be manipulated only by means of operations.

Implementation of abstract data type (ADT)
- .Hidden from the user
- .Same ADT may be implemented in different ways in different Languages.
- .Some language offers in-built ADT's(user define types)

.ADTs support modular design which is very important in software development

Benefits:
.easy to modify, maintain
.profitable
.reusable

Client Benefits:
.simple to use and understand
.familiar
.cheap
.component-based

abstract data type:
---
*.An *abstract data type* (ADT) is the specification of a data type within some language, independent of an implementation. The interface for the ADT is defined in terms of a type and a set of operations on that type. The behavior of each operation is determined by its inputs and outputs. An ADT does not specify *how* the data type is implemented. These implementation details are hidden from the user of the ADT and protected from outside access, a concept referred to as *encapsulation*.

*.A data type is considered to be abstract if it is defines the type of operations it perform.

*.I let u know the use by giving an example ,before 20 years, we had computers,which are very difficult to use a becoz of the long commands we have to type it to speak to it.But now a days,the interface has been become more convinent and gives us comfort, has been interact with mouse and we have icon to speek to it.so,a good lover of abstraction can provide efficiency and productavity giving us confort and convenience..

*.**Abstract Data Type(ADT):** Type defined in terms of its data items and associated operations, **not its implementaiton.**

*.ADT
*. "Model" of a data type
*. Describes
      properties
      operations
*.it doesnt talk about how to perform these operations,how to implement these operations,or how to implement propartirti

*. ADT is actually a Theoretical concept
*. implemented by data structure.

   -3 -3 -1 0 1 2 3
*.Describe ?
  -ve or +ve numbers

*.operations
  addition,subtraction
  multiplication
  division(not by 0).

==========
*.SRC: introduction to datstrucutes in c by ashok n.kamithane
```
    struct date {
       int dd;
       int mm;
       int yy;
    };
    struct date d;// date is abstract data type.
```

==========
*.src: Advanced Topics in C: Core Concepts in Data Structures By Noel
  Kalicharan
q).example of ADT ?
ans).
*.Stack and Queues are examples of ADT.
*.We are familiar with the notion of declaring variables of a given type(double, say) and then performing operations on those variables(for example, add,multiply, and assign) without needing to know "how" those variables are stored in the computer.In this scenario, the compiler designer can change the way a "double" variable is stored, and the programmer would not have to change any programs that use "double" variables. this is an example of an abstract data type.

*.An **"abstract data type"** is one that allows a user to manipulate the data type without any knowledge of how the data type is represented in the computer.

*.in other words, as far as the user is concerned ,all he needs to know are the operations that can be performed on hte data type.The person who is implementing the data type is free to change its implementation without affecting the users.

In stack implementation:
---
*.it is important to observe that the code in main that uses the stack does so via the function "initStack,push,pop", and "empty" and makes no assumption about how the stack elements are stored.this is the hallmark of an abstract data type--it can be used wihtout the user needing to know how it is implemented.

==========

*.Src : Mastering Data Structures Through C Language By J. B. Dixit
*.Concept of "abstract data types".

*.An **abstract data type (ADT)** is more a way of looking at a data structure: ***focusing on what it does and ignoring how it does its job.*** A stack or a queue is an example of an ADT. it is important to understand that both stacks and queues can be implemnted using an array. It is also possible to mplement stacks and queues using a linked list. this demonstrates the *"abstract"* nature of stacks and queues: how they can be considered separately from their implementation.

*.To best descrie the term "Abstract Data Type", it is bst to break the term down into "data type" and then "abstract".

*.in C++, or java, any class represetns a data type , in the sense that a class is made up of data(fields) and permissible operations on that data i.e., functions(methods).By extension,when a data storage structure like a stack or queue is represented by a class, it too can be referred to as a data type. A stack is different in many ways from an int, but they are both defined as a certain arrangement of data and a set of operations on that data.

*.**.Abstact: now lets look at the "abstract" portion of the phrase. the word abstract in our context stands for *"considered apart from the detailed specifications or implementation"*.

*.In c++ or java ,an Abstract Data Type is a class considered without regard to its implementation. it can be thought as a "description" of the data in the class and a list of operations that can be carried out on the data and instructions on how to use these operations.

*.What is excluded though ?,you shold be told what functions(methods) to call, how to call them, and the results that should be expected, but not HOW they work.

*.We can further extend the meaning of the ADT when applying it to data structures such as a stack and queue. in C++ or java, as with an class, it means the data and the operations that can be performed on it. in this context, alhtough, even the fundamentals of how the data is stored should be invisible to the user.Users not only should not know how the functions(methods) work, they should also not know what sructures are being used to store the data.

    Consider for example the stack class. the end user knows that push() and pop() (among other similar methods) exist an how they work.the user doesn't and shouldn't have to know how push() and pop() work, or whether

data is stoed in an array, a linked list, or some other data structure like a tree.

***.The interface:**
*.The ADT specification is often called an interface. it's what the user of the class actually sees. in C++ or java, this would often be the public methods. consider for example, the stack class-- the public funcitons(methods) push() and pop() and similar functions (mehtods) from the interface would be ublished to the end user.

*.An "Abstract data type (ADT)" is more a way of looking at a data structure: focussing on what it does and ignoring how it does its job. for example, a stack or a queue.It is important to understand that both stacks and queues can be implemented usign an array. it is also possible to implement stacks and queues using a linked list. this demonstrates the " abstract" nature of stacks and queues : how they can be considered separatel;y from their implementation.
*.to best describe the term "ABstract datat type", it is best to break the term down into "data type" and then *"abstract"*.

*.in C++, or java, any class represetns a data type , in the sense that a class is made up of data(fields) and permissible operations on that data i.e., functions(methods).By extension,when a data storage structure like a stack or queue is represented by a class, it too can be referred to as a data type. A stack is different in many ways from an int, but they are both defined as a certain arrangement of data and a set of operations on that data.
==========