An **abstract data type (ADT)** is a mathematical model of a data type. It specifies what it means for data to be of a certain type, but it does not specify how such a type is implemented in code or in hardware. An ADT is defined by:

1. a set of **values**, a class of all of the possible objects of that type (not their representations in computer memory but the abstract objects themselves, e.g. the integers, the alphabetic characters, the days of the week, every list, every tree, etc.);
2. a set of **operations** (an interface of functions) that are necessarily defined over objects of this type; and
3. a set of **constraints** that the operations must satisfy.

To implement an ADT, one can represent the values in a **data structure** and program **subroutines** for the operations in such a way that the constraints are satisfied. Each instance would be a member of the class of objects described by the ADT and would behave accordingly. Thus, ADTs are conceptual entities that unify various implementations under a common functional description.

All ADTs have a create() operation (or *constructor*) that yields an instance of the ADT that is *distinct* from any other instances of the ADT that might currently be in use.

Variables

An abstract variable V is a mutable entity.

Operations:

- store(V, x), where x is a value of unspecified type; and
- fetch(V), which yields a value.

Constraints:

- fetch(V) always returns the value x used in the most recent store(V, x) operation on the same variable V.
- If U and V are distinct variables, the sequence {store(U, x); store(V, y)} is equivalent to {store(V, y}; store(U, x)}.
- Fetching the value of a variable before any value is stored under it is generally considered an invalid operation.

Primitive ADTs

The definition of an abstract variable V may optionally restrict values x to members of a set described by some *type*.

Boolean

Values: true, false

Operations:

- Conjunction (AND)
- Disjunction (OR)
- Exclusive disjunction (XOR)
- Equivalence (==)
- Negation (NOT)

Integer

Values: . . . , -2, -1, 0, 1, 2, . . .

Operations:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Increment (++)
- Decrement (–)
- Equivalence (==)
- Less than (<)
- Greater than (>)
- Less than or equal (<=)
- Greater than or equal (>=)

Constraints:

- Division by zero is undefined.

Character

Values: any symbol that is part of your character set (e.g. ASCII, UTF-8)

Operations:

- Concatenation

Characters are sometimes implemented as integers (such as in C), so they may also have arithmetic operations.

Float

Reference

Operations:

- Dereference

Collections or Containers

A collection or container is a grouping of a variable number of objects (generally of the same type).