# Abstract Data Types

* A data type that separates the logical properties from the implementation details.
* Collection of data items and the operations that must be performed on them is called an *Abstract Data Type* (**ADT**).
* Abstract because we are interested in ***what*** *is to be done, not* ***how*** *it is done.*
* An implementation of an ADT consists of *storage structures* to store the data and *algorithms* for the basic operations.
* The predefined data types int, double, char, bool (AKA simple data types, or primitive types) have their definition separate from their implementation. We use these data types without worrying about how they are implemented.
* The same concepts are used to define user-defined data types.

Data abstraction

* Separation of the definition of a data type from its implementation.

## Data structure

* Data structure refers to a construct in a programming language that can be used to store data.

## C++ Data types

* Fundamental types (Arithmetic, void, pointers)
* Structured Types (array, struct, union, vector, string,..)

# Classes

* Produce ADTs
* Not every class is an ADT because the class may not be defined in a certain way.

Example:

#include <iostream>

using namespace std;

// Class definition

**class BankAccount**

**{**

**public:**

**BankAccount(int dollars, int cents, double rate);**

**BankAccount(int dollars, double rate);**

**BankAccount();**

**void set(int dollars, int cents, double rate);**

**void set(int dollars, double rate);**

**void update();** // Add 1 year of simple interest to balance

**double getBalance();**

**double getRate();**

**void output(ostream& outs);**

**private:**

**double balance;**

**double interest\_rate;**

**//converts a percentage to a fraction**

**double fraction(double percent)**

**};**