

## CSE 143

---

### Abstract Data Types

[Chapter 3]

1/11/00 D-1

## Data Abstraction

---

### What is *Abstraction*?

- An idealization
- A focus on essential qualities, disregarding the “details”
- An emphasis on the *what* rather than the *how*  
*Specification vs. Implementation*
- A problem-solving technique

1/11/00 D-2

## Abstraction in Programming

---

- The type `int` is an abstraction for a way of interpreting bits in memory as a number
- A *struct* is an abstraction of a collection of related data items
- A *function* is a programmer-designed abstraction for some computation
- A *module* is a programmer-designed abstraction that groups related functions and data together and provides an interface

1/11/00 D-3

## Why Abstraction?

---

- Abstractions helps in managing complexity
  - Don't need to know details, just interface
- Treat abstractions as “black box” components to build upon
  - Know what inputs go into box, and what outputs come out, but not what goes on inside the box
  - Hierarchical or layered decomposition

1/11/00 D-4

## Review: Types vs. Instances

---

- **Types**
  - General category
  - Usually few in number
  - Some built in (`int`, `char`, `double`, etc.)
  - Programmer-defined (arrays, structs, enums, classes, etc.)
- **Instances**
  - Particular variables, parameters, etc.
  - May have many instances of a given type

1/11/00 D-5

## Abstract Data Types

---

- *ADTs* have two aspects:
  - Collection of *data*
  - *Operations* that can be applied to data
- **Examples**
  - Integers: arithmetic operations, printing, etc.
  - Boolean: AND, OR, NOT, test if `true`, etc.
  - Grade Transcript: Add, remove classes and grades, change grades, etc.

1/11/00 D-6

## Type = Data + Operations

- More Examples:

- Automatic Teller Machine

- Data: cash available, machine status
    - Operations: get account information, dispense cash, confiscate card, ...

- Telephone network switch

- Data: line status, call information
    - Operations: set up and break down calls, send billing information, test circuits, ...

1/11/00 D-7

## Abstract Data Types

Two separate aspects:

- **Interface**

- Name of new type
  - "Constructors" to make instances
  - Public operations on instances

- **Implementation**

- Data representation of new type
  - Implementation of public operations, constructors
  - Additional private operations

1/11/00 D-8

## Implementer / Client / User

- Implementer (programmer)

- writes the internal details of some part of the system
  - defines interface and implementation

- Client (programmer)

- uses the interface of the "black box" provided by the Implementer
  - does not (directly) use the implementation!

- User (non-programmer)

- sees only the exterior behavior of the system
- Related language for functions: Caller vs. called

1/11/00 D-9

## Textbook example: List ADT

- A list... names, groceries, numbers, etc.
- What do you need to do?
  - Create and destroy a list
  - Find out how long it is
  - Add (insert) new items to it
  - Delete items
  - Look at (retrieve) items
- Vector
  - A list where you can retrieve values by their index

1/11/00 D-10

## Great Ideas, but...

- How do we actually get modularity, abstraction, ADTs, black boxes, etc. in our programs?
- "Encapsulation": wrapping up the data and operations together in a clean package
- Historical note: for many years programmers have struggled to do this. Recent trends in programming languages make it easier.

1/11/00 D-11