ADT According to Various Authors

- Robert Sedgewick
  - An ***ADT*** is a ***data type***...
    - a *set of values* and a *collection of operations* on these values
    - that is *accessed only through an interface*
- Robert Kruse and Alexander Ryba
  - The definition of any ***ADT*** involves ***two parts***
    - A description of *the way in which the components are related* to each other
    - A statement of *the operations that can be performed* on elements of the ADT

ADT According to Various Authors

- Larry Nyhoff
  - An ***ADT*** consists of...
    - a *collection of data items*, and
    - the *basic relationships among them* and *operations that must be performed* on them
  - The word ***abstract*** refers to the fact that...
    - the *data* and the *basic operations and relationships* defined on it are being *studied independently of how they are implemented*
    - we are thinking of *what can be done* with the data, *not how it is done*

ADT According to Various Authors

- Mark Allen Weiss
  - An ***ADT*** is...
    - ...a *set of objects* together with a *set of operations*
  - ADTs are ***abstractions***…
    - ...*no* where in an ADT's definition is there any *mention* of *how the set of operations is implemented*

ADT According to Various Authors

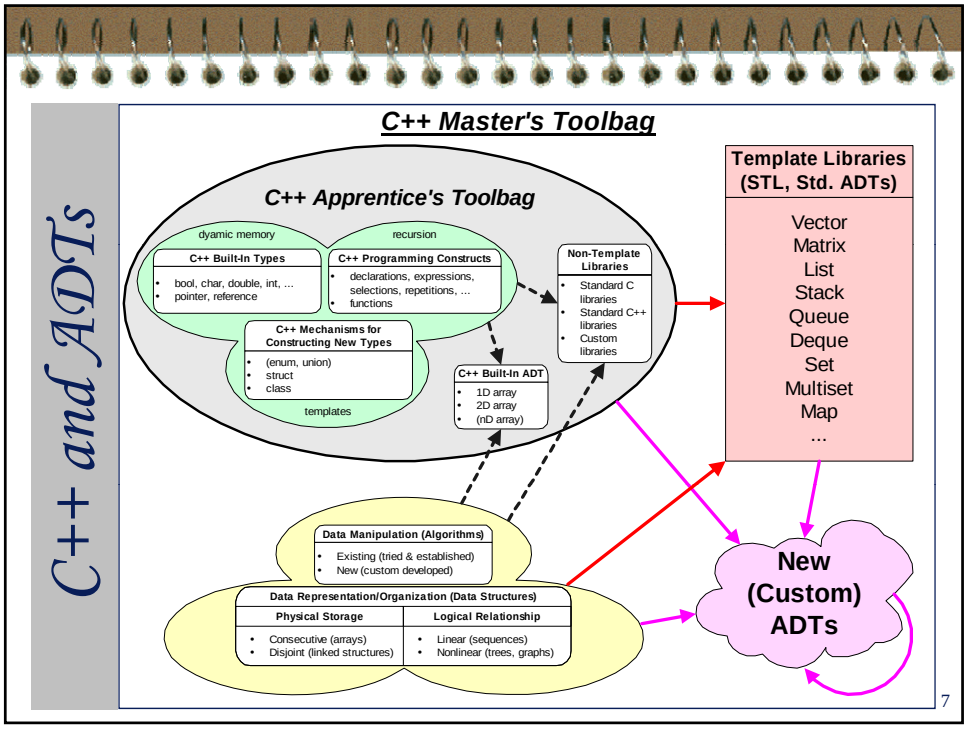
- Carrano, Helman and Veroff
  - An ***ADT*** is…
    - ...a *collection of data* and a *set of operations* on the data
  - You ***can use an ADT's operations***...
    - ...if you *know their specifications*
    - ...*without knowing how the operations are implemented* or *how the data is stored*

ADT According to Various Authors

- Behrouz Forouzan and Richard Gilberg
  - An ***ADT*** is…
    - a *data declaration* packaged together with the *operations* that are allowed on the data
  - This definition implies ***two attributes*** for ADTs
    - The *structures are opaque*
      - we *can use them without knowing how they are implemented*
    - The *operations are opaque*
      - we *know what they will do*; we *don't know how they will do it*

2 Important Aspects of ADTs and C++ Support

- ***Data type***
  - Consists of data and operations for manipulating data
- ***Data abstraction***
  - *The end:* data hiding (including data protection)
  - *The means:*
    - Specification separated from implementation ("what" part separated from "how" part)
    - Data accessed only through interface
- What C++ feature support each of the above?
  - What feature(s) support(s) data and operations?
  - What feature(s) support(s) data hiding?
    - How is ADT specification separated from ADT implementation?
    - How is data access restricted only through interface?



ADT Pervades throughout Most of Textbook

- Chapter 2 sets the stage
  - As you read (or re-read) Chapter 2, note that…
    - The concept of ADT is explicitly referred to only in the opening paragraphs and in the summary
    - C++ support for ADT is illustrated by way of a basic introduction to C++ **`class`**
  - The overall theme of "Concept of ADT and how it is supported by C++" is often missed by students ☹
- Most of the remaining chapters of the book…
  - Illustrate further concept of and C++ support for ADT