Data Design and Implementation

Outline

• Data
• Abstraction and Build-in Types
Different Views of Data

• **Data**
  – The representation of information in a manner suitable for communication or analysis by humans or machines

• Data are the **nouns** of the programming world:
  – The objects that are manipulated
  – The information that is processed

---

Different Views of Data

• **Data Abstraction**
  • Separation of a data type’s logical properties from its implementation

<table>
<thead>
<tr>
<th>LOGICAL PROPERTIES</th>
<th>IMPLEMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are the possible values?</td>
<td>How can this be done in C++?</td>
</tr>
<tr>
<td>What operations will be needed?</td>
<td>How can data types be used?</td>
</tr>
</tbody>
</table>
Different Views of Data

- **Abstract Data Type**
  - is a specification of
    - a set of data sharing the same features
    - the set of operations that can be performed on the data
  - Implementation independent.
- **A Data Structure** is an concrete implementation of an **Abstract Data Type**.
Different Views of Data

• **Application (or user) level** modeling real-life data in a specific context

• **Logical (or ADT) level** abstract view of the domain and operations

• **Implementation level** specific representation of the structure to hold the data items, and the coding for operations
Different Views of Data

• **Application (or user) level**: Library of Congress, or Baltimore County Public Library.

• **Logical (or ADT) level**: The domain is a collection of books; operations include: check book out, check book in, pay fine, reserve a book.

• **Implementation level**: Representation of the structure to hold the “books” and the coding for operations.
ADT Operators (operations)

- **Constructors**
  - Operation that creates new instances of an ADT; usually a language feature
- **Transformers (mutators)**
  - Operations that change the state of one or more data values in an ADT
- **Observers**
  - Operations that allow us to observe the state of the ADT
- **Iterators**
  - Operations that allow us to access each member of a data structure sequentially

---

C++ Built-In Data Types

- **Simple**
  - Integral: char, short, int, long, enum
  - Floating: float, double, long double
- **Composite**
  - array, struct, union, class
- **Address**
  - pointer, reference
Abstraction & Built-In Types

• **One-dimensional array**
  – A structured composite data type made up of a finite, fixed size collection of ordered homogeneous elements to which direct access is available

```
int numbers[10];
```

• **A two-dimensional array**
  – A structured composite data type made up of a finite, fixed size collection of homogeneous elements having relative positions and to which there is direct access

```
int dataTable[10][6];
```
Abstraction & Built-In Types

Application Level

Can you think of other applications for two-dimensional arrays?

Higher-Level Abstraction

• Class
  – An unstructured type that encapsulates a fixed number of data components (data members) with the functions (member functions) that manipulate them; its predefined operations on an instance of a class are whole assignment and component access

• Client
  – Software that declares and manipulates objects (instances) of a particular class
Higher-Level Abstraction

- **Class specification**
  - A specification of the class members (data and functions) with their types and/or parameters
- **Class implementation**
  - The code that implements the class functions

*Why would you want to put them in separate files?*

Reference

- Reproduced by permission of Jones and Bartlett Publishers International.