Data Structures

---

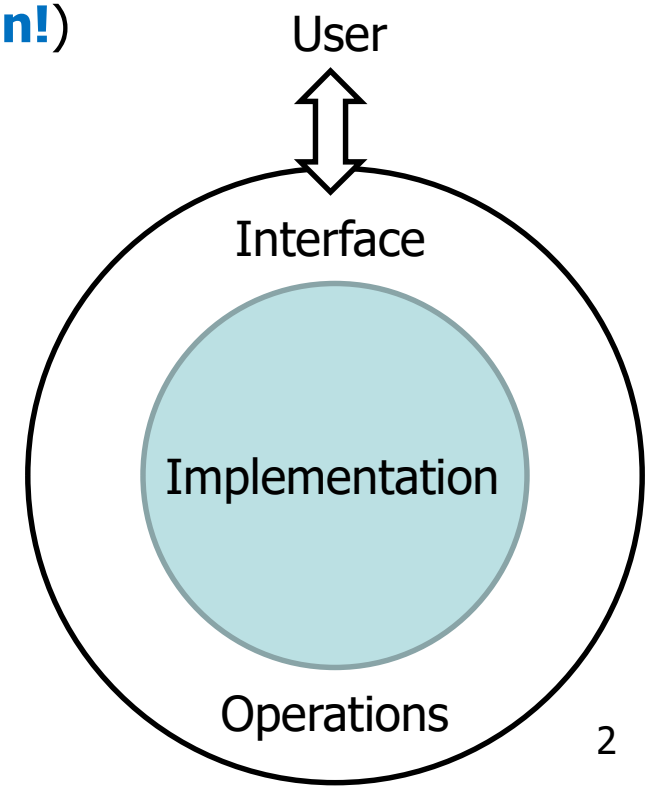
2. Abstract Data Types

Abstract Data Types (1)

- A definition of data type solely in terms of
  - Set of related data items (or values)
  - Set of operations on the data

- Separation of logical properties from the implementation details
  - Hide implementation details (**Encapsulation!**)

- What not how is focus



Abstract Data Types – Examples

- Whole numbers (integers)
  - Operations
    - arithmetic operations (addition, subtraction, etc.)

- Flight reservation
  - List of seats
  - Operations
    - Find empty seat
    - Reserve a seat
    - Cancel a seat assignment

Abstract Data Types (2)

- ADTs definition consists of
  - Storage structures (i.e., data structures) to store data items
  - Algorithms for basic operations

- Storage structures/data structures used in the implementation
  - Provided in a language (primitive or build-in)
  - Built from the language constructs (user-defined)

→Separation of a data type from its implementation

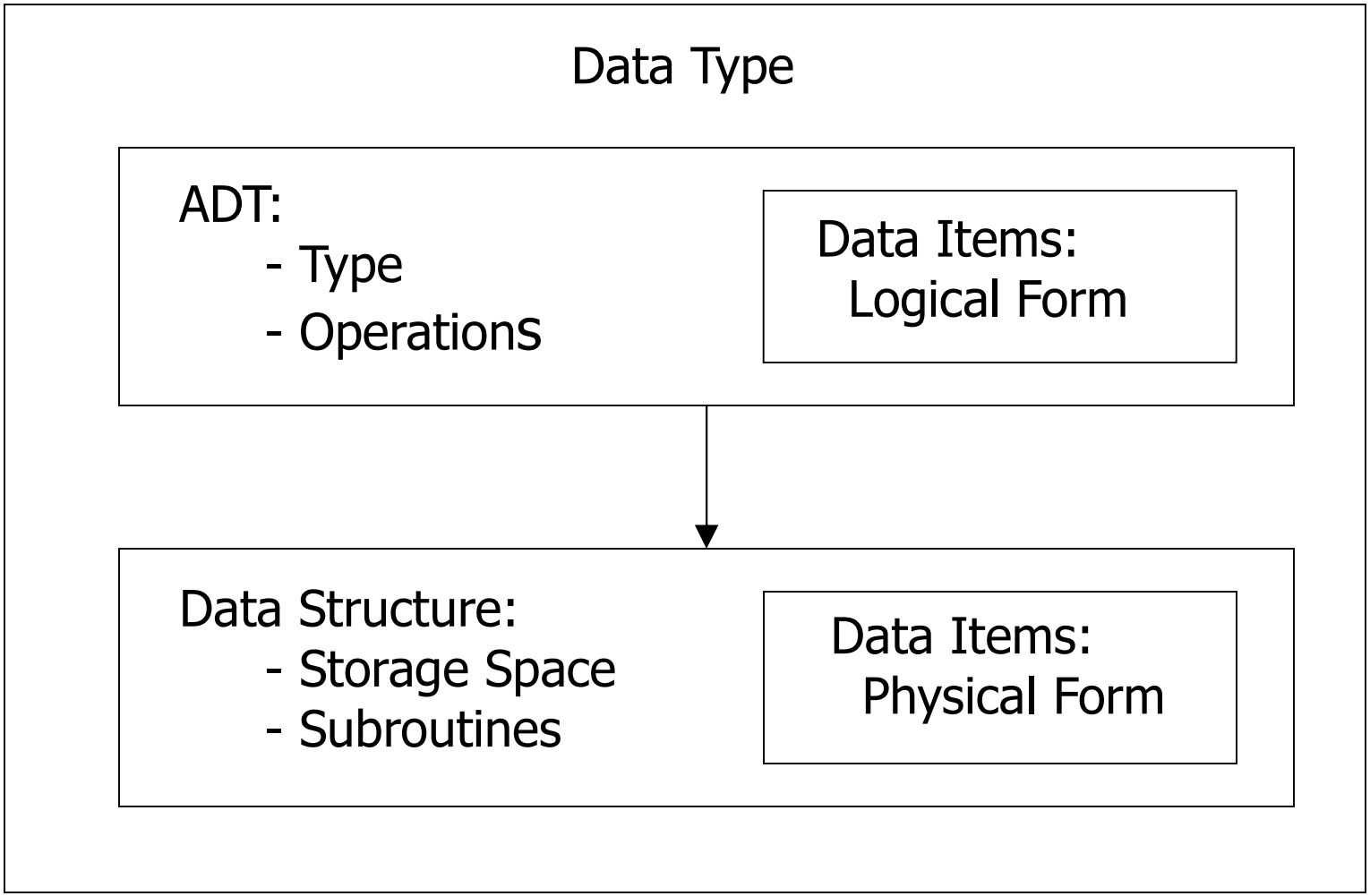
Data Structures

- A data structure is physical implementation of an ADT
  - Each operation associated with ADT is implemented by one or more subroutines in the implementation

- Data structure usually refer to an organization of data in main memory

- File structure is an organization of data on peripheral storage such as disk drive

ADT vs. Data Structures



Example: Airplane Flight Reservation (1)

- Consider example of an airplane flight with 10 seats to be assigned

- Operations
  - List available seats
  - Reserve a seat

- Implementation: How to store, access data?
  - 10 individual variables

Implementation: 10 Individual Variables

List available seats:

```
1. if seat1 == ‘ ’;
        display 1
2. if seat2 == ‘ ’;
        display 2
.
.
.

10. if seat10 == ‘ ’;
        display 10
```

Reserve a seat:

```
1. Set DONE to false
2. if seat1 ==‘ ’;
   print “do you want seat #1??”
   Get answer
   if answer==‘Y’;
        set seat1 to ‘X’
        set Done to True
3. if seat2 ==‘ ’ and Done == false;
   print “do you want seat #2??”
   Get answer
   if answer==‘Y’;
        set seat2 to ‘X’
        set Done to True
.
.
.
```

Example: Airplane Flight Reservation (2)

- Consider example of an airplane flight with 10 seats to be assigned

- Operations
  - List available seats
  - Reserve a seat

- Implementation: How to store, access data?
  - 10 individual variables
  - An array of variables

Implementation: An array of variables

List available seats:

```
for number ranging from 0 to max_seats-1, do:
        if seat[number] == ‘ ’;
                Display number
```

Reserve a seat:

```
Reading number of seat to be reserved
if seat[number] is equal to ‘ ’;
        set seat[number] to ‘X’
else
        Display a message that the seat having this number is
        occupied
```

Example: Airplane Flight Reservation (2)

- This simple example illustrate the concept of an Abstract Data Type
- ADT consists of
  - Collection of data items
  - Basic operations that must be performed on them
- In the example, a collection of data is a list of seats
- Basic operations are
  - List available seats
  - Reserve a seat

ADTs for Primitive Data Types

Boolean Data

- Data values: `{false, true}`

- Operations:
  - And `&&`
  - Or `||`
  - Not `!`

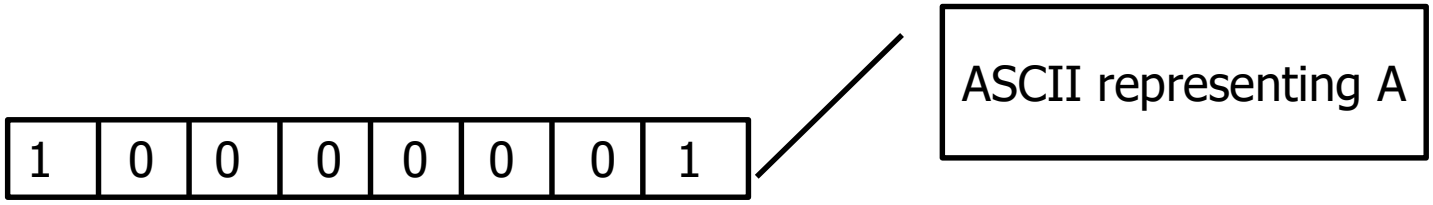
&&	0	1
0	0	0
1	0	1

||	0	1
0	0	1
1	1	1

x	!*x*
0	1
1	0

Character Data

- Data values
  - Numeric codes (ASCII, Unicode)
    - 1 byte for ASCII
    - 2 bytes for Unicode (UTF-16)



- Basic Operations
  - Comparison (equal, less than, greater)
  - Use of numeric codes
  - ….

Integer Data

- Data values
  - Non-negative (unsinged) integer
    - Base-two representation in a fixed number of bits (e.g., 32 bits)
    - 88 : 0000 0000 0000 0000 0000 0000 0101 1000
  - Signed integer
    - Ones' or Two's complement representation in fixed number of bits
    - 127 : 01111111, -127: 10000000 (Ones' complement)
    - 127: 01111111, -127: 10000001 (Two's complement)
- Operations
  - Arithmetic operations
    - Addition, subtraction, multiplication, division

Any Question So Far?