CMPT 125

Introduction to Computing Science and Programming II

October 27, 2020

Assignment 3

- Assignment 3 is due to November 5, 23:59

  https://www.cs.sfu.ca/~ishinkar/teaching/fall21/cmpt125/assignments.html

- You need to submit one file to Canvas – *assignment3.c*
- Please make sure it compiles with the provided makefile

```
>> make
>> ./run_test3
```

Topics:

- Sorting algorithms
- Function pointers
- Standard operations on array (find, map, reduce)

Missing semester - MIT

https://missing.csail.mit.edu/

A series of lecture on random topics that are

- usually not covered in any course, but

- every CS student should know

✓ Command-line Environment

✓ Editors (Vim)

✓ Version Control (Git)

✓ Debugging

Seriously, this is something you all should know.
This will make your studies so much easier

- Link is on piazza-> resources

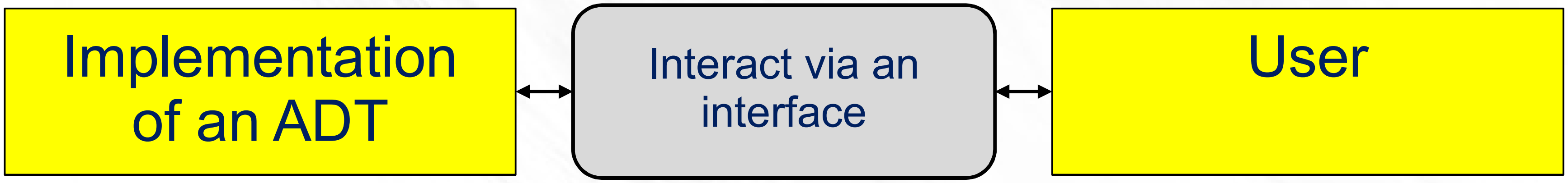
Today

- Abstract data types
    - Stack
    - Queue
    - Linked List

Abstract data types

Abstract Data Type

- Abstract data type (ADT):
  - A collection of data and a set of allowed operations on that data.
  - ***<u>Describes data + operations</u>***
  - Does not specify how the data is stored
  - Does not specify how the operations are carried out

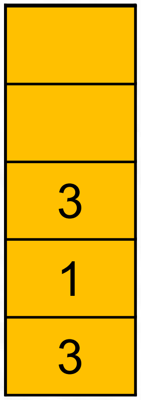


Example: Stack

- A stack: an ordered collection of items with the following operations:
  - push(item): add an item to the stack
  - pop(): remove an item from the stack, and return its value
  - isEmpty(): checks if the stack is empty
- Removal follows a last-in-first-out order (LIFO)

Example: Stack

- A stack: an ordered collection of items with the following operations:
  - push(item): add an item to the stack
  - pop(): remove an item from the stack, and return its value
  - isEmpty(): checks if the stack is empty
- Removal follows a last-in-first-out order (LIFO)
  - *init()*
  - *push(3)*
  - *push(5)*
  - *pop() -- returns 5*
  - *push(1)*
  - *push(3)*



3

1

3

Example: Stack

- A stack: an ordered collection of items with the following operations:
    - push(item): add an item to the stack
    - pop(): remove an item from the stack, and return its value
    - isEmpty(): checks if the stack is empty
- Removal follows a last-in-first-out order (LIFO)
- **Usage:**
    - Stacks in execution of a program
    - Undo operation in paint/notepad

The definition is independent from the implementation

Implementing Stack

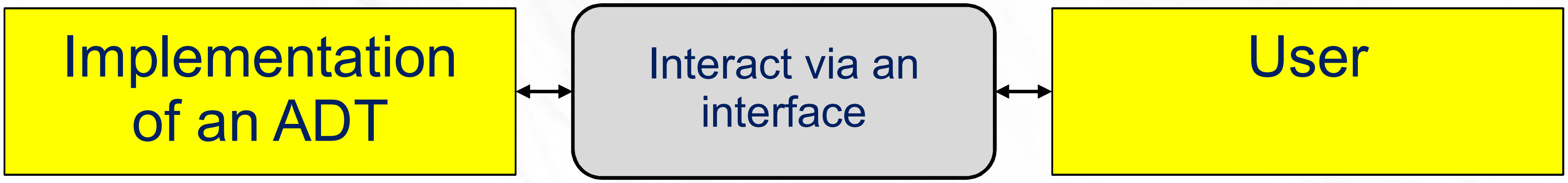
- Question: How would you implement Stack?

Back to abstract data types

Abstract Data Type

- Abstract data type (ADT):
  - A collection of data and a set of allowed operations on that data.
  - ***<u>Describes data + operations</u>***
  - Does not specify how the data is stored
  - Does not specify how the operations are carried out

The definition is independent from the implementation



Software Engineering Principles

- Encapsulation
  - Bundle related data and operations together
- Modularity
  - break up a problem into smaller, manageable programming tasks
- Information Hiding
  - keep the implementation details private
  - keep the interface stable
- Finding a good selection of interfaces is the foundation for writing large scale software

Interfaces

- An interface refers to data and a set of operations of the data.
  - Parametrized by inputs
  - Serves as a contract
- Why use interfaces?
  - Code re-usage
  - Code independence
  - Allows modifying parts of the implementations without the need to change the entire program

More examples of ADTs

Queue

- A queue: an ordered collection of items with the following operations:
  - enqueue(item): add an item to the queue
  - dequeue(): remove an item from the queue
  - isEmpty(): checks if the queue is queue
- Removal follows a first-in-first-out order (FIFO)

- **Usage:**
  - Waitlist
  - Documents sent to printer
  - Line at a supermarket

Queue

- A queue: an ordered collection of items with the following operations:
    - enqueue(item): add an item to the queue
    - dequeue(): remove an item from the queue
    - isEmpty(): checks if the queue is queue
- Removal follows a first-in-first-out order (FIFO)
- There is no bound on the number of element in the set
- Question: How would you implement Queue?

Dynamic array

- A dynamic array: is an array with the following operations:
  - init(): create an empty array
  - set_value (index, item): set array[index]=item
  - get_value (index): get array[index]

- There is no bound on the number of element in the array

- Question: How would you implement Dynamic array?

Set

- Set: is a bag of element with the following operations:
  - init(): create an empty set
  - add(item): add an item to the set
  - contains?(item): checks if the set contain item
  - remove(item): removes an item from the set
  - is_empty?: checks if the set is empty
- There is no bound on the number of element in the set
- Question: How would you implement Set?

Implementing a stack

Implementing Stack

- A stack: an ordered collection of items with the following operations:
  - create(): creates an empty stack
  - push(item): add an item to the top of the stack
  - pop(): remove an item from the top of the stack
  - isEmpty(): checks if the stack is empty
  - peek(): return the top element (without removing it)

- Removal follows a last-in-first-out order (LIFO)

Implementing Stack

- Question: How would you implement Stack?

```
typedef struct {
    ???
} stack_t;
```

- Caveat: Stack does not have a bound on its capacity.
- Want the capacity to be bounded by the limitations of your computer

Questions?

Comments?