Stacks

Canberra Institute of Technology

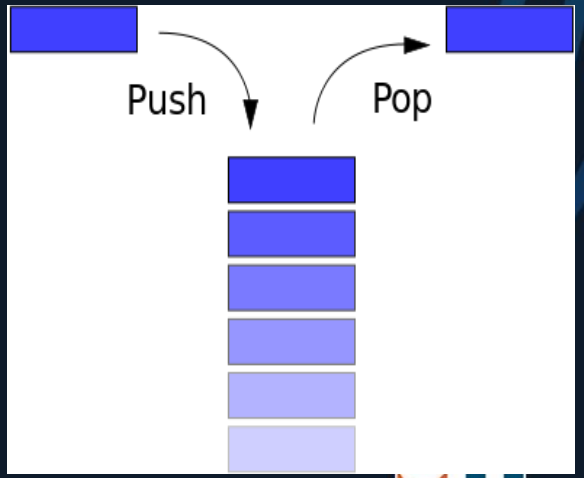
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Pancakes

- You can add pancakes to the top of the stack,
- But you can only (safely) eat the one on the top

Stacks

- Stacks are a container that work on the LIFO (Last In First Off) concept.
- This means that the last thing on the stack is the first thing to come back off.
- Elements can only be *pushed* onto the stack, and then *popped* off the stack.
- Is how your program runs and stores variables and function calls

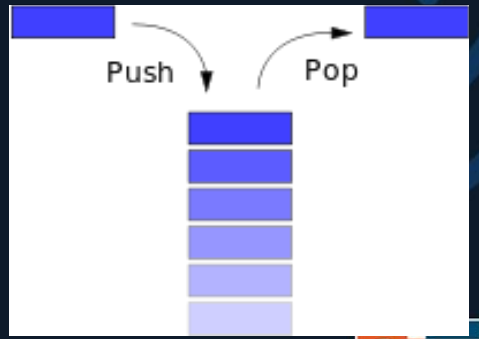


Properties of Stacks

- Stacks are dynamically resizable arrays (of sorts)
- As stacks are a LIFO container you cannot access any element within the container except for the last added element!
- To get to an element that is underneath it you have to pop elements off.

Operations of a Stack

- The stack has a very limited set of operations when compared to other container types:
  - Empty: Returns true if stack is empty
  - Size: Returns the number of elements
  - Push: Adds element to the end/top
  - Pop: Removes element at the end/top of the stack
  - Top: Returns the element at the end of the stack



Stack Pseudocode

- Let’s look at a simple stack class that will allow us to:
  - Push elements to the top of our stack
  - Pop elements off our stack
  - Check if our stack is empty
  - Check the size of our stack
  - Access the top element of our stack

Constructor

- The constructor should:
  - Take in a stack size (capacity)
  - Allocate an array of size: *capacity* (and initialize them to 0 or NULL)
  - Initialise array size (*m_size*) to capacity
  - Initialise top element (*m_top*) to -1

```
def Stack(capacity)
    SET m_data to array of capacity elements
    SET m_size to capacity
    SET m_top to -1
```

Pop

- Pop removes the top element from the stack
- We can do this by decrementing m_top by 1

```
function Pop()
      IF m_top >= 0
          SET m_data[ m_top ] to NULL
          DECREMENT m_top by 1
      END IF
```

Push

- Push adds an element to the end of the stack
- Resize the stack when you run out of space

```
function Push(value)
    INCREMENT m_top by 1
    IF m_top equals m_size-1
         resize m_data array
         SET m_size to new size
    ENDIF
    SET m_data[m_top] to value
```

Top

- Returns the top element of our stack
- If m_top < 0, return NULL or throw error

```
function Top()
      IF m_top < 0
          return null
      ELSE
          return m_data[ m_top ]
      END IF
```

Empty and Size

- Empty returns true if the stack is empty
- Size returns the number of elements on the stack

```
function Empty()
      IF m_top is equal to -1
          return true
      ELSE
          return false
      END IF
```

```
function Size()
      RETURN m_top + 1
```

Destructor

- As usual, our destructor should free any data that was created in our stack class

```
def ~Stack()
    deallocate m_data
```

References

- C++ Primer Plus, Chapter 10: Abstract Data Types, p553
- http://en.wikipedia.org/wiki/Stack_(abstract_data_type)

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX