CS 32 Spring 2021

TA: Manoj Reddy

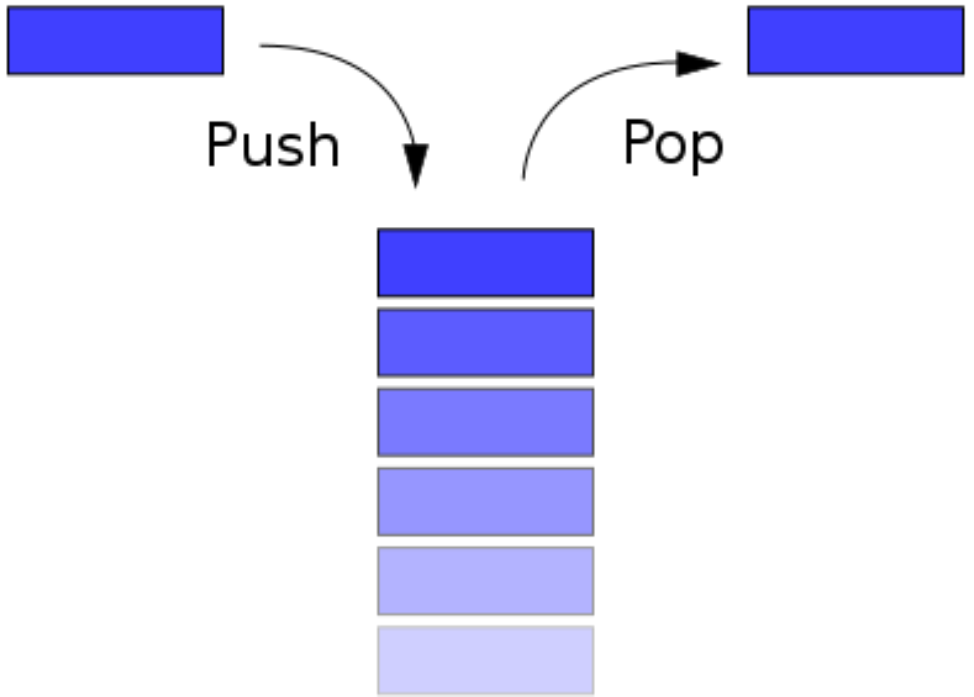
LA: Katherine Zhang

Outline

- Stacks
- Queues

Stacks

- Important Data Structure in CS
- Last In First Out
- Applications:
  - Find path in maze
  - Infix to Postfix expression evaluation
  - “Undo” in Microsoft Word
- Operations:
  - Push: Add item on top of stack
  - Pop: Remove the top item
  - Top: Access top most item
  - Size: Number of items
- Item can be any type: primitives + objects



Stack

Implementations

- Arrays
  - Array of Items
  - Variable to store index of top element
    - Initially set to 0
    - Stores 1 + (index of top-most element)
- LinkedList
  - Head points to top of the stack
- Tradeoff ???

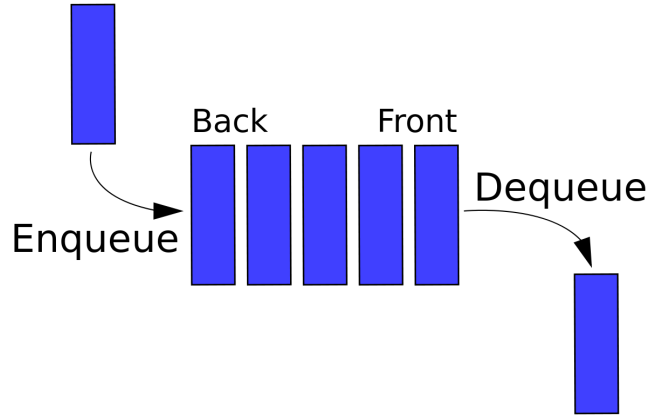
Stacks

Stacks are so popular that the C++ people actually wrote one for you. It's in the Standard Template Library (STL)!

```
#include <iostream>
#include <stack>              // required!

int main()
{
  std::stack<int>  istack; // stack of ints

  istack.push(10);         // add item to top
  istack.push(20);

  cout << istack.top();  // get top value
  istack.pop();          // kill top value

  if (istack.empty() == false)
    cout << istack.size();
}
```

Credit: Prof. Nachenberg

Queues



- Data structure that represents its literal meaning
- First-in-First-Out
- Applications:
  - Streaming video buffering
  - Process Context Switching
  - Optimal route navigation
- Operations:
  - Enqueue: Inserts item
  - Dequeue: Removes and returns item
  - GetFront(): Retrieves the item that will be removed next
  - Size: Number of items

Implementations

- Arrays
  - Array + variable storing index of the end of queue
  - Enqueue by adding item at end
  - Dequeue by removing item at index 0 and shifting all elements left
- LinkedList
  - Enqueue at end using tail pointer
  - Dequeue from front using head pointer
- Circular Queue
  - Array based implementation
  - **3** variables to track: start index, end index and **size (why do we need this?)**

A Queue in the STL!

The people who wrote the Standard Template Library also built a queue class for you:

```
#include <iostream>
#include <queue>

int main()
{
        std::queue<int>  iqueue;   // queue of ints

        iqueue.push(10);           // add item to rear
        iqueue.push(20);
        cout << iqueue.front();    // view front item
        iqueue.pop();         // discard front item
        if (iqueue.empty() == false)
          cout << iqueue.size();
}
```

Credit: Prof. Nachenberg

LA Worksheet