CS 225

#12: Array Resize Analysis

September 25, 2017

List Implementation #2: Array

Stack.cpp

```
#include "Stack.h"

template <class T>
void Stack::push(T & t) {
  // If we are about to overflow, double the size of the array:
  if (count_ + 1 == size_) {
    size_ *= 2;
    T * newArray = new T[size_];
    for (unsigned i=0; i < count_; i++) { newArray[i] = arr_[i]; }
    delete arr_;
    arr_ = newArray;
  }

  // Insert (push) the element into the array-backed stack:
  arr[ count_++ ] = t;
}

template <class T>
T & Stack::pop() {
  return arr[ --count_ ];
}
```

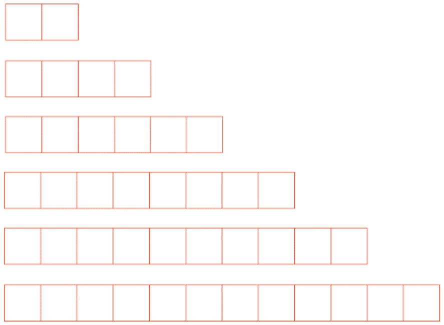
Stack.h

```
#ifndef STACK_H
#define STACK_H

template <class T>
class Stack {
  public:
    Stack();
    Stack(const Stack &other);
    ~Stack();
    Stack& operator=(const Stack &other);

    void push(T & t);
    T & pop();
    bool isEmpty() const;

  private:
    T * arr_;
    unsigned size_, count_;

};

#endif
```

T* arr:	C	S	2	2	5		
	[0]	[1]	[2]	[3]	[4]		

Resize Strategy – Details:

Strategy #1:



Strategy #2:

Three designs for data storage in data structures:

1. Not possible / T & data
2. T ** arr / T * data
3. T * arr / T data

Implication of Design

1. Who manages the lifecycle of the data?
2. Is it possible to store a NULL as the data?
3. If the data is manipulated by user code while stored in our data structure, are the changes reflected within our data structure?
4. Is it possible to store literals?
5. Speed

	Storage by Reference	Storage by Pointer	Storage by Value
Lifecycle management of data?			
Possible to insert NULL?			
External data manipulation?			
Literal storage?			
Speed			

Queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

template <class T>
class Queue {
  public:





  private:

};

#endif
```

A queue is a: _______________ data structure

...which stands for:

Why do we care about stacks and queues?

CS 225 – Things To Be Doing:

1. Exam #3 starts today ("Theory Exam", Advanced C++)
2. MP2 is due today; MP3 released on Tuesday
3. Lab Extra Credit → Attendance in your registered lab section!
4. Daily POTDs