#11: List: Implementation (Array)

February 11, 2022 · *G Carl Evans*

Array List Implementation:

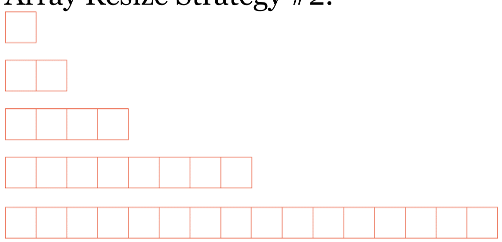
Array Resize Strategy #1:

...total copies across all resizes: ___________

...total number of insert operations: ___________

...average (amortized) cost of copies per insert: ___________

Array Resize Strategy #2:



...total copies across all resizes: ___________

...total number of insert operations: ___________

...average (amortized) cost of copies per insert: ___________

Running Time:

	Singly Linked List	Array
Insert/Remove at **front**		
Insert after a **given** element		
Remove after a **given** element		
Insert at **arbitrary** location		
Remove at **arbitrary** location		

Stack ADT

Queue ADT

Stack and Queue Implementations

Stack.h

```
#pragma once

#include <vector>

template <typename T>
class Stack {
  public:
    void push(const T & d);
    T pop();
    bool isEmpty();

  private:
    std::vector<T> list_;
};

#include "Stack.hpp"
```

Stack.hpp

```
template <typename T>
void Stack<T>::push(const T & d) {
  list_.push_back(d);
}

template <typename T>
T Stack<T>::pop() {
  T data = list_.back();
  list_.pop_back();
  return data;
}
```

Circler Queue

Example 1


```
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

Example 2


```
Queue<char> q;
q.enqueue('m');
q.enqueue('o');
q.enqueue('n');
…
q.enqueue('d');
q.enqueue('a');
q.enqueue('y');
q.enqueue('i');
q.enqueue('s');
q.dequeue();
q.enqueue('h');
q.enqueue('a');
```

Iterators

In C++, iterators provide an interface for client code access to data in a way that abstracts away the internals of the data structure.

An instance of an iterator is a current location in a pass through the data structure:

Type	Cur. Location	Current Data	Next
Linked List			
Array			
Hypercube			

The iterator minimally implements three member functions:
`operator*`, Returns the current data
`operator++`, Advance to the next data
`operator!=`, Determines if the iterator is at a different location

Implementing an Iterator

A class that implements an iterator must have two pieces:

1. [Implementing Class]: Must implement:

   -

   -

2. [Implementing Class' Iterator]:
   A separate class (usually an internal class) that extends **`std::iterator`** and implements an iterator. This requires:

   -

   -

   -

Locations of ::begin and ::end iterators:

Type	::begin()	::end()
Linked List		
Array		

CS 225 – Things To Be Doing:

1. lab_memory due Sunday
2. mp_list extra credit part1 due Monday
3. Daily POTDs