Array Lists

Lecture 15
Section 6.2

Robb T. Koether

Hampden-Sydney College

Thu, Feb 19, 2009
1. List Implementations
2. Array Lists
3. Circular Array Lists
4. Assignment
List Implementation

- We will implement lists in a number of ways.
- As an array.
  - Fixed head.
  - Circular.
- As a linked list.
  - Singly linked.
  - Doubly linked.
  - Circularly linked.
  - Recursively linked.
An array list is an implementation of the List ADT that uses an array to store the list elements.

The **ArrayList** class is very similar to the **Vectr** class.

The difference is that the size is adjustable after the list has been constructed.
Data Members

**ArrayList Data Members**

- `int mSize` - The number of elements in the list.
- `int capacity` - The number of array positions allocated.
- `T* element` - A pointer to the first array element.
The list elements

\[ a_0, \ldots, a_{m\text{Size}-1} \]

are stored in array positions \texttt{element[0]} through \texttt{element[mSize - 1]}. 
Validity Requirements

- The object is structurally valid provided
  - capacity >= 0.
  - mSize >= 0 and mSize <= capacity.
  - If capacity == 0, then element == NULL.
  - If capacity > 0, then element != NULL.
The ArrayList Class

- Download `arraylist.h`
- Download and run `ListTest.cpp`
Inlining Functions

Definition (Inline Functions)

An **inline function** is copied and pasted, with appropriate modifications, into the calling function, thereby replacing a function call.

- A function may be inlined by using the `inline` keyword.

  ```c
  inline sqr(int n);
  ```
A function may be inlined by using the `inline` keyword.

Then, for example,

```c
int s = sqr(a);
```

is replaced with

```c
int s = a*a;
```
A class member function may be inlined by writing its definition within the class definition.
The `insert()` Function

- The `insert()` function must
  - Test that `pos` is valid.
  - Test that there is sufficient capacity to add one more element. If not, then call `setCapacity()` to double the capacity.
  - Shift the elements with indexes `pos` to `mSize - 1` to the right one position.
  - Then copy `value` to index `pos`.
  - Increment the size of the list.
The insert() Function

```cpp
template <class T>
void ArrayList<T>::insert(int pos, const T& value) {
    assert(pos >= 0 && pos <= mSize);

    if (mSize == capacity) {
        if (capacity == 0)
            setCapacity(1);
        else
            setCapacity(2 * capacity);
    }

    for (int i = mSize - 1; i >= pos; i--)
        element[i + 1] = element[i];
    element[pos] = value;

    mSize++;
    return;
}
```
The `remove()` Function

- What should the `remove()` function do?
Circular Array Lists

Definition (Circular Array List)

A circular array list is like an ordinary array list, except that the elements may wrap around the ends of the array.

- This makes it much more efficient to add and remove elements from the front end of the list.
Circular Array Lists Dynamics

- Begin with an empty list.
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).

```
10
```

`tail`

`head`
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).

```
tail
10
head
45
```
Add six elements at the tail (pushBack).
Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Delete three elements from the head (popFront).

```
head
45 25 90 15 40
tail
```

```
Delete three elements from the head (popFront).
Circular Array Lists Dynamics

- Delete three elements from the head (popFront).
Circular Array Lists Dynamics

- Add one element at the head (pushFront).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).

![Circular Array Lists Diagram](image)
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Circular Array Lists Dynamics

- Add six elements at the tail (pushBack).
Assignment

Homework

- Read Section 6.2, pages 257 - 269. (Static array-based list = \texttt{Vectr})
- Read Section 6.3, pages 269 - 287. (Dynamic array-based list = \texttt{ArrayList})