CSC212
Data Structure
- Section FG

Lecture 10

The Bag and Sequence Classes with Linked Lists

Instructor: Feng HU
Department of Computer Science
City College of New York

Reviews: Node and Linked List

- Node
  - a class with a pointer to an object of the node class
  - core structure for the linked list
  - two versions of the “link” functions
    - why and how?
    -

The Complete **node** Class Definition

- The node class is fundamental to linked lists
- The private member variables
  - data_field
  - link_field
- The member functions include:
  - A constructor
  - Set data and set link
  - Retrieve data and retrieve link

```
class node
 {
 public:
         // TYPEDEF
         typedef double value_type;

         // CONSTRUCTOR
         node(
           const value_type& init_data = value_type( ),
           node* init_link = NULL
         )
         { data = init_data; link = init_link; }

         // Member functions to set the data and link fields:
         void set_data(const value_type& new_data) { data = new_data; }
         void set_link(node* new_link)         { link = new_link; }

         // Constant member function to retrieve the current data:
         value_type data( ) const { return data; }

         // Two slightly different member functions to retrieve
         // the current link:
         const node* link( ) const { return link; }
         node* link( )          { return link;}

 private:
         value_type data;
         node* link;
 };
```

default argument given by the value_type default constructor

Why TWO? p. 213-4

Reviews: Node and Linked List

- Linked Lists Traverse
  - How to access the next node by using link pointer of the current node
  - the special for loop

```
size_t list_length(const node* head_ptr)
{
    const node *cursor;
    size_t count = 0;
    for (cursor = head_ptr; cursor != NULL; cursor = cursor->link())
            count++;
    return count;
}
```

Reviews: Node and Linked List

- Insert
  - Insert at the head
    - set the head_ptr and the link of the new node correctly
  - Insert at any location
    - cursor pointing to the current node
    - need a pre-cursor to point to the node before the current node (two approaches)
    - the third approach: **doubly linked list**

Reviews: Node and Linked List

- Delete
  - Delete at the head
    - set the head_ptr correctly
    - release the memory of the deleted node
  - Delete at any location
    - cursor pointing to the current node
    - need a pre-cursor to point to the node before the current node (two approaches)
    - the third approach: **doubly linked list**

Key points you need to know

Toolkit Code

- Linked List Toolkit uses the node class which has
  - set and retrieve functions
- The functions in the Toolkit are not member functions of the node class
  - length, insert(2), remove(2), search, locate, copy,...
  - compare their Big-Os with similar functions for an array
- They can be used in various container classes, such as bag, sequence, etc.

Container Classes using Linked Lists

- Bag Class with a Linked List
  - Specification
  - Class definition
  - Implementation
  - Testing and Debugging
- Sequence Class with a Linked List
  - Design suggestion – difference from bag
- Arrays or Linked Lists: which approach is better?
  - Dynamic Arrays
  - Linked Lists
  - Doubly Linked Lists

Our Third Bag - Specification

- The documentation
  - nearly identical to our previous bag
  - The programmer uses the bag do not need to know know about linked lists.
- The difference
  - No worries about capacity therefore
    - no default capacity
    - no reserve function
  - because our new bag with linked list can grow or shrink easily!

Our Third Bag – Class Definition

- The invariant of the 3rd bag class
  - the items in the bag are stored in a linked list (which is dynamically allocated)
  - the head pointer of the list is stored in the member variable head_ptr of the class bag
  - The total number of items in the list is stored in the member variable many_nodes.
- The Header File (code)

Our Third Bag – Class Definition

- How to match bag::value_type with node::value_type

```
class bag
{
 public:
             typedef node::value_type value type;
             ......
}
```

- Following the rules for dynamic memory usage
  - Allocate and release dynamic memory
  - The law of the Big-Three

Our Third Bag - Implementation

- The Constructors
  - default constructor
  - copy constructor
- Overloading the Assignment Operator
  - release and re-allocate dynamic memory
  - self-assignment check
- The Destructor
  - return all the dynamic memory to the heap
- Other functions and the code

Sequence Class with Linked List

- Compare three implementations
  - using a fixed size array (assignment 2)
  - using a dynamic array (assignment 3)
  - using a linked list (assignment 4)
- What are the differences?
  - member variables
  - value semantics
  - Performance (time and space)

Sequence – Design Suggestions

- Five private member variables
  - many_nodes: number of nodes in the list
  - head_ptr and tail_ptr : the head and tail pointers of the linked list
    - why tail_ptr - for attach when no current item
  - cursor : pointer to the current item (or NULL)
  - precursor: pointer to the item before the current item
    - for an easy insert (WHY)
- Don't forget
  - the dynamic allocation/release
  - the value semantics and
  - the Law of the Big-Three

Sequence – Value Semantics

- Goal of assignment and copy constructor
  - make one sequence equals to a new copy of another
- Can we just use list_copy in the Toolkit?
  - list_copy(source.head_ptr, head_ptr, tail_ptr);
- Problems (deep copy – new memory allocation)
  - many_nodes OKAY
  - head_ptr and tail_ptr OKAY
  - How to set cursor and precursor ?

Dynamic Arrays vs Linked Lists

- Arrays are better at random access
  - O (1) vs. O(n)
- Linked lists are better at insertions/ deletions at a cursor
  - O(1) vs O(n)
- Doubly linked lists are better for a two-way cursor
  - for example for insert O(1) vs. O(n)
- Resizing can be Inefficient for a Dynamic Array
  - re-allocation, copy, release

Reading and Programming Assignments

- Reading after Class
  - Chapter 6

- Programming Assignment 4
  - Detailed guidelines online!