CS 008
Lecture notes
3/14/24

1 Outline

- Announcement
- Templates
- Linked-lists and Double Linked-lists
- Big-O discussion
- Assignment 2 (Sequences part 2)

2 Announcement

Announcement:

There will be Office Hours held from Monday, Wednesday and Friday from 4-5 pm on Zoom. You can find the link on the Canvas homepage.

3 Templates

When you have multiple classes that perform the same task but for different datatypes, you can create a template class instead. You can declare a template with `template <class Template_parameter_name>` on the line before the class definition of the header file.

When converted to a template class, various changes to member and non-member functions will be needed. Any functions that use the template class will need modifications. The implementation of templated class methods must be in the header file. Instead, put a include statement at the bottom of the file.

Note:

See Section 6.2 of the textbook (Main and Savitch) for full syntax. **(Particularly the textbox on Page 304).**

Some examples of templates can be seen in the figures below.

Figure: bag_templated.h

```
#ifndef MAIN_SAVITCH_BAG5_H
#define MAIN_SAVITCH_BAG5_H

#include <cstdlib> // Provides NULL and size_t and NULL
#include "node2.h" // Provides node class

template <class Item>

class bag
{
    public:
        // TYPEDEFS
        typedef std::size_t size_type;
        typedef Item value_type;
        // CONSTRUCTORS and DESTRUCTOR
        bag( );
        bag(const bag& source);
        ~bag( );
        // MODIFICATION MEMBER FUNCTIONS
        size_type erase(const Item& target);
        bool erase_one(const Item& target);
        void insert(const Item& entry);
        void operator +=(const bag& addend);
        void operator =(const bag& source);
        // CONST MEMBER FUNCTIONS
        size_type count(const Item& target) const;
        Item grab( ) const;
        size_type size( ) const { return many_nodes; }

    private:
        node<Item> *head_ptr; // Head pointer for the
↪   list of items
```

```
        size_type many_nodes; // Number of nodes on the
↪  list
};

// NONMEMBER FUNCTIONS for the bag<Item> template class
template <class Item>
bag<Item> operator +(const bag<Item>& b1, const
↪  bag<Item>& b2);

// The implementation of a template class must be
↪  included in its header file:
#include "bag_templated.template"

#endif
```

4 Linked lists and Doubly-Linked lists

A linked list is a type of data structure that is comprised of list of nodes. Each node, except for the **head**, points to another node regardless of where it is in memory. The nodes contain data and a pointer to the next node.

The fundamental unit of both types is called a **node**. A node for a **linked list** consists of data and a pointer which points to the next node in the list. For a **doubly-linked list**, the nodes consist of data, a pointer which points to the next node in the list and the pointer which points to the previous node in the list.

Example:

For a **linked list**, an example node can be seen here:

```
struct node {

  node* next;
  (data type) data;

};
```

Both **linked lists** and **doubly-linked lists** use nodes. However, both nodes are different in small ways. An example of a doubly-linked node can be:

```
struct node {

  node* next;
  node* previous;
  (data type) data;

}
```

Question:

How would you know where to find the last node in a linked list?

Answer: To get the last node in a linked list, you would need to iterate through the linked list and find a node that points to *null*. Therefore, the node that points to *null* will be the last.

What is the time complexity (O(..)) of finding the last node in a linked list?

Answer: The time complexity of finding the last node in a linked list is $O(n)$ where n represents the number of nodes.

5 O(n) discussion

Consider a function: $f(x) = a_n x^n + x^{n-1}$

What would be the time complexity of this function?

To figure out the time complexity of the function given, we must look for the highest order of the function: $a_n x^n$. The coefficient is then dropped to reveal x^n which is now the function's time complexity, $O(x^n)$.

Question:

If we had an array of integers with a size of 10 and we have 5 elements filled up in the array, what would be the time complexity of adding a 6th element to the array?

Answer: It would be $O(1)$ since it would take 2 operations: Getting the 6th index in the array and setting a value for the 6th element. The coefficient is dropped from the largest power and is reduced to 1. Therefore, the result is $O(1)$.

6 Sequence part 2

There will be a new assignment that will revisit Sequences and also templates.

The new assignment is designed to:

- Get you more familiar with templates
- To understand the emphasis on the header (.h) file and the implementation (.cpp)
- Get more comfortable analyzing the run time complexity of real functions.