Laboratory 2: Using dynamic memory, Arrays, Vectors

CSC205A Data structures and Algorithms Laboratory B. Tech. 2017

Vaishali R Kulkarni

Department of Computer Science and Engineering
Faculty of Engineering and Technology
M. S. Ramaiah University of Applied Sciences
Email: vaishali.cs.et@msruas.ac.in
Tel: +91-80-4906-5555 (2212) WWW: www.msruas.ac.in

Introduction and Purpose of Experiment

- This experiment also includes design and implementation of algorithms for creating and applying linked list and deques. The test cases are written to validate the design and implementation of linked list ADT.
- Testing improves the quality and reliability of a ‘C’ program. This experiment introduces test case design for a ‘C’ program to test the desired program for bugs and errors.

Aim and objectives

Aim:

- To design and develop linked list ADT and deque ADT and design test cases to test the both the programs.

Objectives:

At the end of this lab, the student will be able to

- Design and develop test cases for a given C program
- Design and develop linked list and deque ADT
- Application of ADT to illustrate the addition and deletion of the elements in the middle of the collection.

Theory Behind the Experiment

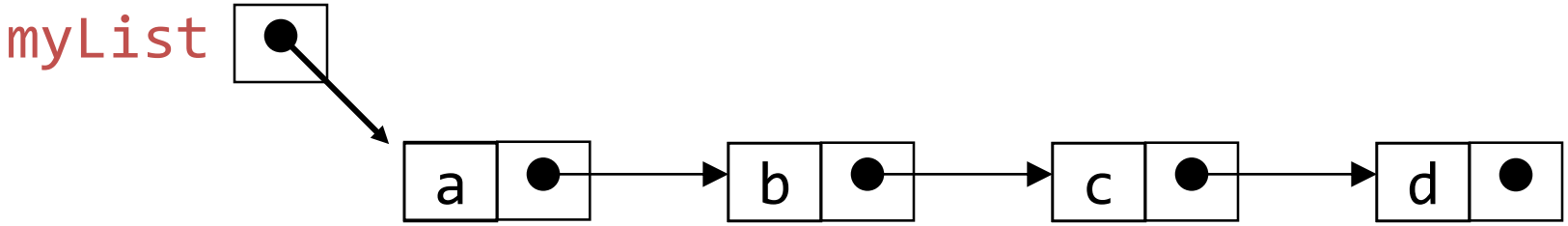
Lists:

A list or sequence is an Abstract Data Type that represents an ordered sequence of values, where the same value may occur more than once. Lists can be implemented using arrays or linked lists. In this experiment linked list implementation is considered.

- Insertion: Ordered lists are maintained in sequence according to the data or a key that identifies the data. Example: ssn#. In Random list or chronological lists, there is no sequential relationship between two elements. Generally found in data-gathering applications. Data Insertion is at the end.
- Deletion: Deletion requires the list be searched to locate data being deleted. The predecessor of the element to be deleted has its link member assigned the address of the successor to the deleted element.
- Retrieval: Requires data to be searched and located in a list and presented to the calling module without the change in contents of the list.
- Traversal: List traversal processes each element in a list in sequence. Looping algorithm used and each execution of the loop processes one element in the list.

Linked lists

- A linked list consists of:
  - A sequence of nodes

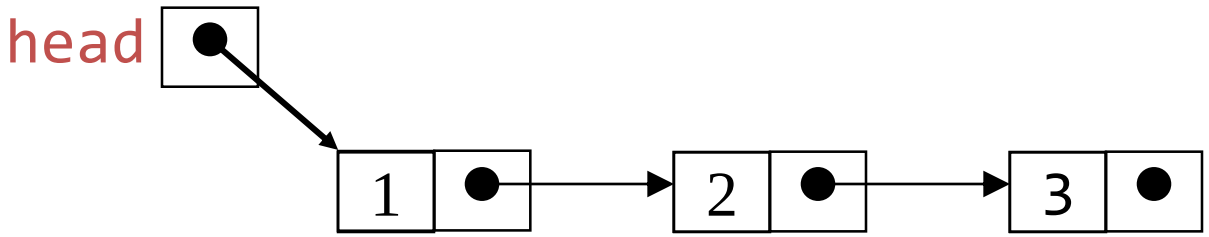


Each node contains a value
and a link (pointer or reference) to some other node

The last node contains a null link

The list may (or may not) have a header

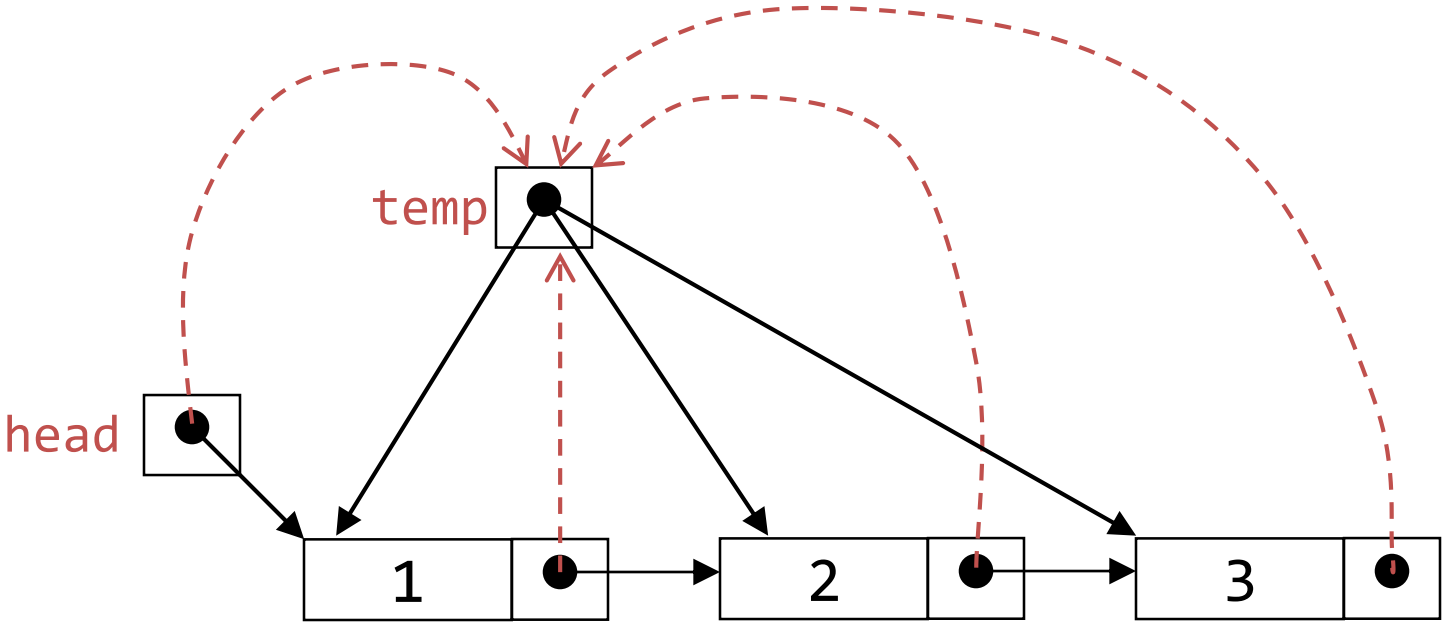
Creating links in C




```c
struct node
{
    int data;
    struct node *next;
};
```

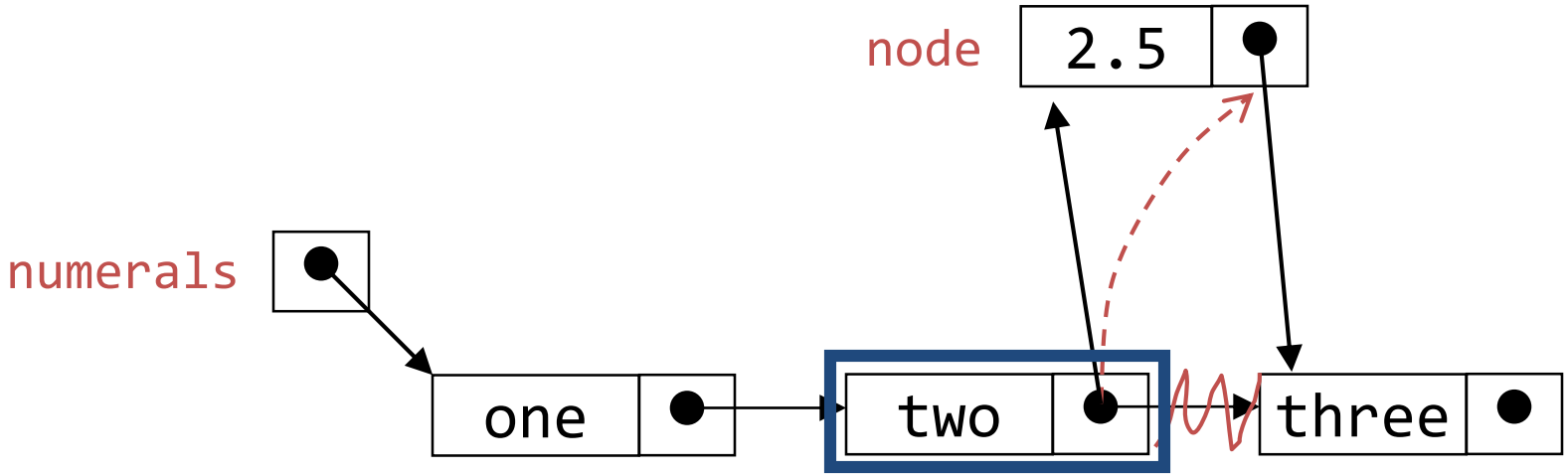
```c
int main()
{
    struct node *head=NULL;
    struct node *second=NULL;
    struct node *third=NULL;
    head=(struct node*)malloc(sizeof(struct node));
    second=(struct node*)malloc(sizeof(struct node));
    third=(struct node*)malloc(sizeof(struct node));
    head->data=1;
    head->next = second;
    second->data=2;
    second->next = third;
    third->data=3;
    third->next = NULL;
}
```

Traversing and Printing (animation)




```
void printList(struct node* head)  {
 struct node *temp=head;
 while(temp!=NULL)
 {
    printf("%d \n",temp->data);
    temp=temp->next;
 }
 }
```

Inserting after (animation)



Find the node you want to insert after

First, copy the link from the node that's already in the list

Then, change the link in the node that's already in the list

Code for Insert After

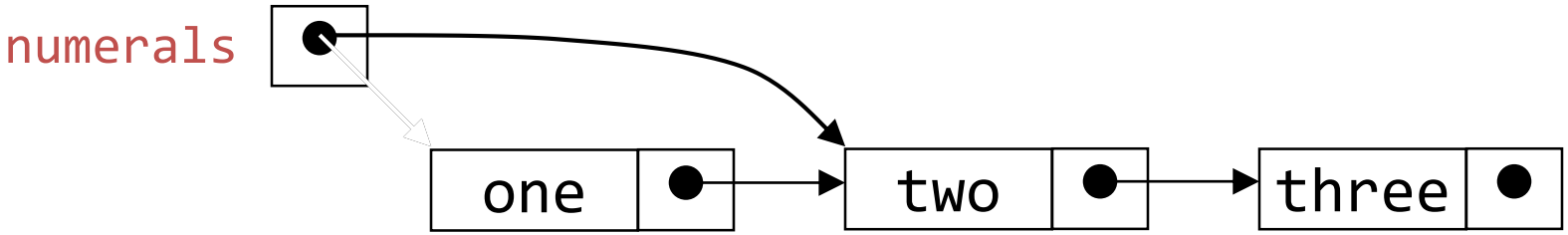
```
void insertAfter(struct node *prev_node, int newData)
 {
     if(prev_node!=NULL)
     {
      struct node *newNode=
                (struct node*)malloc(sizeof(struct node));
      newNode->data= newData;
      newNode->next=prev_node->next;
      prev_node->next= newNode;
     }
}
```

Deleting a node from a SLL

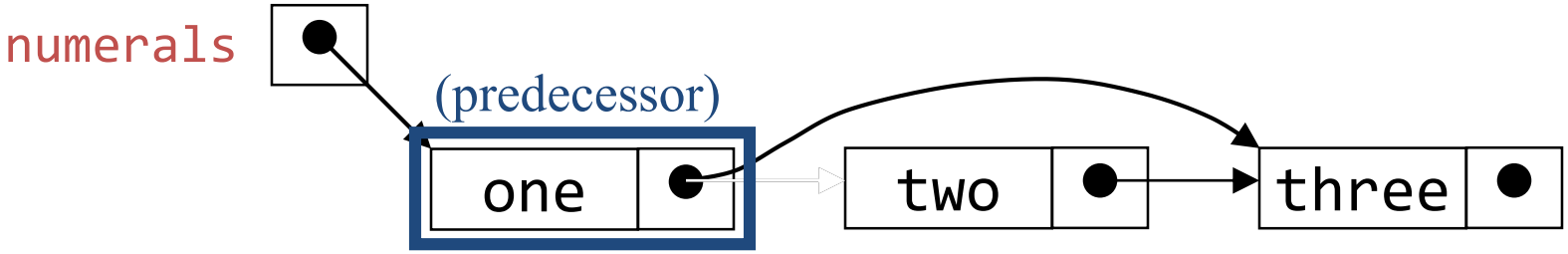
- In order to delete a node from a SLL, you have to change the link in its *predecessor*
- This is slightly tricky, because you can't follow a pointer backwards
- Deleting the first node in a list is a special case, because the node's predecessor is the list header

Deleting an element from a SLL

- To delete the first element, change the link in the header



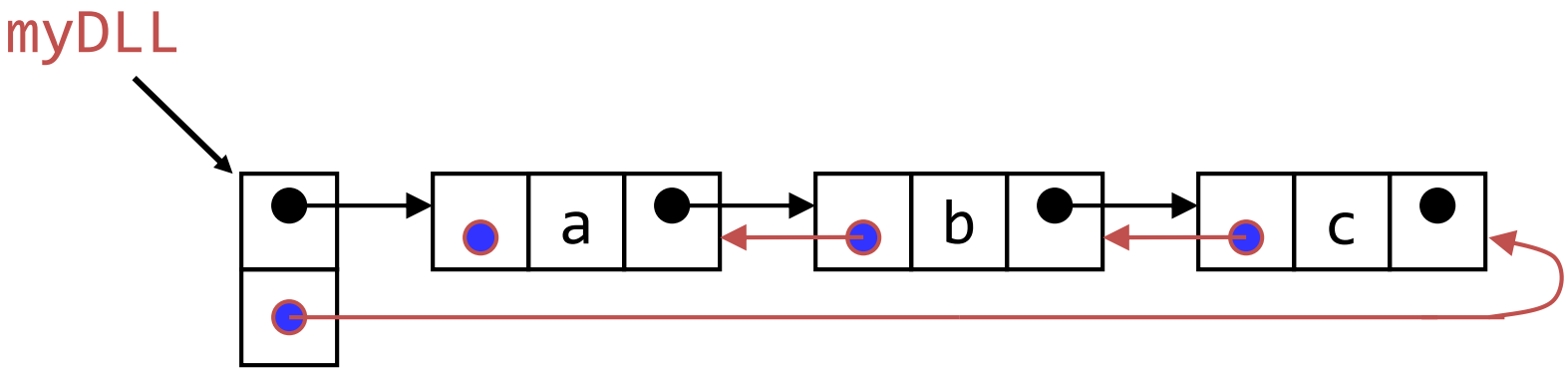
- To delete some other element, change the link in its predecessor



- Deleted nodes will eventually be garbage collected

Doubly-linked lists

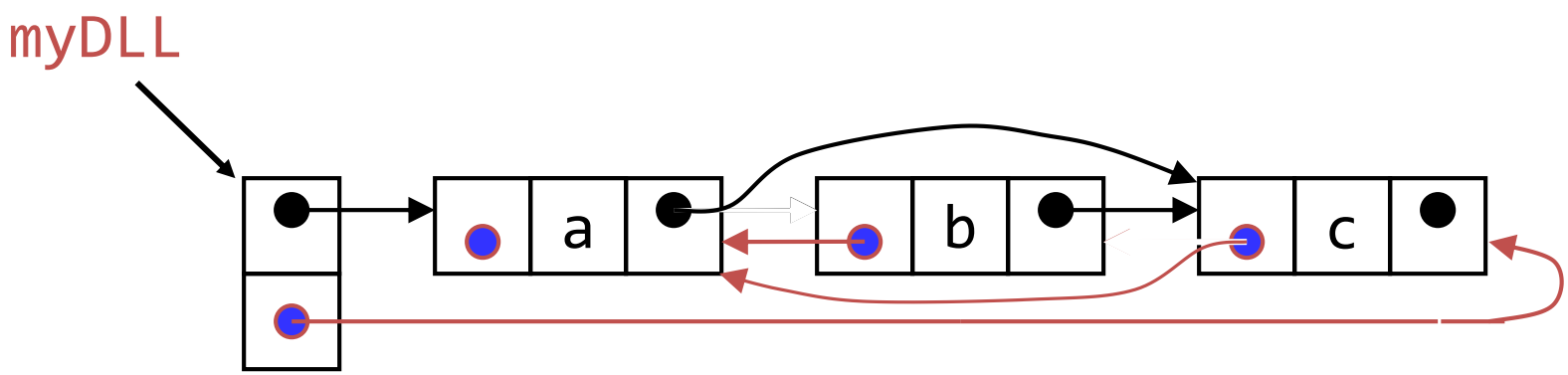
- Here is a doubly-linked list (DLL):



- Each node contains a value, a link to its successor (if any), *and* a link to its predecessor (if any)
- The header points to the first node in the list *and* to the last node in the list (or contains null links if the list is empty)

Deleting a node from a DLL

- Node deletion from a DLL involves changing *two* links
- In this example,we will delete node b



- We don't have to do anything about the links in node b
- Garbage collection will take care of deleted nodes
- Deletion of the first node or the last node is a special case

Experimental Procedure

- Analyse the problem statement
- Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
- Implement the algorithm in C language
- Compile the C program
- Design test cases and test the implemented program
- Document the Results
- Analyse and discuss the outcomes of your experiment

Exercise

- Design the data structures, algorithm and write the program to implement a linked list ADT and deque ADT. Apply the designed ADT to demonstrate its operations on integer data and perform addition and deletion, insertion operation. Design the test cases and validate the results. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

Key factors and discussion

- A Dequeue is implemented using lined list
  - Use the list ADT operations and implement the Dequeue
- Dequeue should not allow insertion/deletion in the middle
  - Program should confirm this fact
  - The ADT operations should be exercised on given data
- Test case design
  - A standard template available for test case design, which has the following details: Test Case ID, Description, Input, expected output and obtained output, and the result.
  - A test report should be generated which provides the details such as: 1) total number of test cases 2) total number of test cases pass and fail and so on.

Testing

Test case ID	Description	User Input	Expected output	Actual Output	Remarks Pass/fail
Unique id for identifying each test case	Brief description of what is being tested in the program	User input for this test case	Expected output for this input	Obtained output for this test case	If both expected output and actual output is same then test case is pass else it is fail.

Test Report

1. Total number of test cases:
2. Total number of test cases passed:
3. Total number of test cases failed:

Results and Presentations

- Calculations/Computations/Algorithms

  The calculations/computations/algorithms involved in each program has to be presented

- Presentation of Results

  The results for all the valid and invalid cases have to be presented

- Analysis and Discussions

  how the data is manipulated or transformed, what are the key operations involved. Errors encounters and how they are resolved.

- Conclusions

  Summary

Comments

- Limitations of Experiments

  Outline the loopholes in the program, data structures or solution approach.

- Limitations of Results

  Present the test cases; justify if the program is tested correctly considering all the outcomes. Mention what is not tested, if any.

- Learning happened

  What is the overall learning happened

- Conclusions

  Summary

References

- Gilberg, R. F., and Forouzan, B. A. (2007): A Pseudocode Approach With C, 2nd edn. Cengage Learning