



# **The implication and taxonomy of linear and nonlinear data structure**

By

**Hafiza Aqsa Ali (13022050024)**

A Dissertation Submitted For the Degree of  
Master in computer science in the Department of Computer Science, Faculty of  
Science, University of Management science and technology

**October 2015**

**Supervised by:**

Dr. Muhammad Shoaib Farooq

Copyright © 2015 Aqsa Ali

## **Declaration**

I declare that this dissertation is my own original work. Where collaborations with other researchers are involved, or materials generated by other researchers are included, the parties and/or materials are acknowledged or are explicitly referenced as appropriate.

This work is being submitted for the degree of Master in Computer Science at the University of Management Science and technology. This thesis has not been submitted to any other university or institution for any other degree or examination.

---

Date

---

Signature

**Aqsa Ali**

## Publications

Some ideas, figures and tables of this dissertation have previously appeared in the following six publications:

Muhammad Shoaib Farooq, Aqsa Ali, Adnan Abid (2014) What Should Be Taught About Arrays In Cs2? 2nd International Conference on Computational and Social Sciences (ICCS-14), Vol (3), pp. 2904-2913.

Farooq, M. S., Ali, A., Abid, K., & Abid, A. (2015). Learning Stack: Structured, Object Oriented, Generic, and Design Pattern Approach. *JOURNAL OF FACULTY OF ENGINEERING & TECHNOLOGY*, 21(2), 191-203.

Muhammad Shoaib Farooq, Aqsa Ali, Adnan Abid (2014) What Should Be Taught About Arrays In Cs2? J. Appl. Environ, Biol. Sci Vol(4) issue 7s, pp. 22-32.

Farooq, M. S., Ali, A., Abid, (2015). Learning Queue: Structured, Object Oriented, Generic, and Design Pattern Approach. *JOURNAL OF FACULTY OF ENGINEERING & TECHNOLOGY*. (Submitted).

Farooq, M. S., Ali, A., Abid, (2015). Learning Linked List for the Course of Data Structures. *JOURNAL OF RESEARCH AND REFLECTION IN EDUCATION* (Submitted).

Farooq, M. S., Ali, A., Abid, (2015). Learning Tree for the Course of Data Structures. *JOURNAL OF RESEARCH AND REFLECTION IN EDUCATION* (in progress).

## **Dedication**

*This thesis is dedicated to my parents*

*And*

*My husband, their proficient*

*Guidance and splendid inspiration had*

*Positively influenced my life and made me capable*

*Of attaining this degree*

*Successfully.*

## **ACKNOWLEDGEMENT**

I am highly grateful to Allah who blessed me with strength, wisdom and proper light in order to complete my research work.

I am thankful to my kind supervisor, Dr. Muhammad Shoaib Farooq for his precious guidance, which paved the way to steer me a right. He not only guided me throughout this work but also provided me with his unprecedented and continuous support, encouragement and motivation. This work never has been completed without his enthusiastic supervision. I am highly thankful to Dr. Adnan Abid, who gives me ideas for research.

The prayer of my lovely parents, sisters and husband always remained with me in this difficult work. Thanks for their selflessness support. My friends and my fellows, who remembered me in their prayers and extended their moral support to me are also worthy for my complements.

# Table of Contents

Declaration.....	ii
Publications.....	iii
Dedication.....	iv
ACKNOWLEDGEMENT.....	v
Table of Contents.....	vi
List of Codes.....	x
List of Figures.....	xii
List of Tables.....	xv
Abstract.....	xvi
Chapter 1: Introduction.....	1
1.1    Scope of the Study.....	2
1.2    Problem Statement.....	2
1.3    Research Questions.....	3
1.4    Research Methodology.....	3
1.5    Research Contributions.....	4
1.6    Thesis Outline.....	4
Chapter 2: Related work.....	7
2.1    Arrays: Basic Data structure, Variants and Applications.....	8
2.2    Linked lists: Basic Data structure, Variants and Applications.....	8
2.3    Stacks: ADT, Design patterns and Applications.....	8
2.4    Queues: ADT, Design patterns and Applications.....	9
2.5    Trees: Structure, Variants and Applications.....	9
Chapter 3 Arrays: Basic Data structure, Variants and Applications.....	10
3.1    Classification of arrays.....	11
3.2    Memory.....	11
3.2.1    Static arrays.....	11
3.2.2    Fixed Stack Dynamic arrays.....	12
3.2.3    Stack Dynamic arrays.....	12
3.2.4    Fixed heap dynamic arrays.....	12
3.2.5    Heap dynamic arrays.....	13

3.3	Mapping Functions.....	13
3.3.1	One dimensional.....	14
3.3.2	Two dimensional:.....	14
3.3.3	Three-to-N dimensional array.....	15
3.3.4	Two Dimensional representations.....	16
3.4	Classification using Subscript type.....	17
3.4.1	Scalar Type.....	17
3.4.2	Associative arrays.....	18
3.5	ADT (Abstract data types).....	19
3.5.1	Matrix.....	19
3.5.2	Vector.....	19
3.5.3	ArrayList.....	19
3.6	Considerations for curriculum.....	21
Chapter 4 Linked Lists: Basic Data structure, Variants and Applications .....		23
4.1	Taxonomy of Linked List.....	24
4.2	Structure of a linked list .....	24
4.2.1	Singly linked list.....	25
4.2.2	Doubly linked list.....	25
4.3	Implementation Variants .....	26
4.3.1	Imperative (naive) linked list .....	26
4.3.2	Circular linked list.....	27
4.3.3	ADT singly linked list.....	28
4.3.4	ADT Doubly Linked List:.....	30
4.3.5	Generic singly linked list.....	30
4.3.6	Generic doubly linked list.....	32
4.3.7	Sentinel singly linked list.....	32
4.3.8	Sentinel doubly linked list.....	32
4.3.9	Array based linked list.....	35
4.4	Storage variants .....	37
4.4.1	Volatile linked list.....	37
4.4.2	Persistent linked list.....	37
4.4.3	Persistent doubly linked list.....	37

4.5	Advance linked list.....	38
4.5.1	Array of linked list.....	38
4.5.2	Ordered list.....	39
4.6	Considerations of curriculums .....	42
Chapter 5: Stacks: ADT, Design patterns and Applications.....		44
5.1	Formal Definition of a Stack.....	45
5.2	Taxonomy of Stack using Different Implementations .....	46
5.2.1	Structured Implementation of Stack.....	47
5.2.1.1	Array based (fixed size) stack:.....	47
5.2.1.2	Linked list based (dynamic size) stack:.....	49
5.2.2	Object Oriented Implementation of Stack.....	50
5.2.3	Generic stack.....	51
5.2.4	Adapter design pattern. ....	53
5.3	Relative Importance of the Implementation Variants .....	56
Chapter 6: Queues: ADT, Design patterns and Applications .....		58
6.1	Formal Definition of a queue .....	59
6.2	Taxonomy of Queue using Different Implementations.....	60
6.2.1	Structured Implementation of Queue .....	62
6.2.1.1	Array based (fixed size) queue:.....	62
6.2.1.2	Linked list based (dynamic size) queue: .....	63
6.2.2	Object Oriented Implementation of Queue .....	65
6.2.2.2	Adapter design pattern. ....	68
6.2.3	Variants of Queue.....	70
6.2.3.1	Circular Queue: .....	70
6.2.3.2	Priority Queue: .....	72
6.3	Relative Importance of the Implementation Variants. ....	73
Chapter 7: Trees: Structure, Variants and Applications .....		75
7.1	Binary tree.....	77
7.1.1	Structure binary tree .....	78
7.1.1.1	Complete binary tree.....	78
7.1.1.1.1	Heap.....	80
7.1.1.2	Binary Search tree.....	80

7.1.1.3	ADT binary search tree.....	80
7.1.1.4	Balance Techniques.....	86
7.1.1.4.1	Red black tree.....	86
7.1.1.4.2	AVL tree.....	92
7.2	Variants.....	96
7.2.1	Rooted binary tree.....	96
7.2.2	Full binary tree.....	96
7.2.3	Perfect binary tree.....	97
7.2.4	Infinite binary tree.....	97
7.2.5	Strictly binary tree.....	97
7.2.6	Degenerated binary tree.....	97
7.2.7	Extended binary tree.....	97
7.3	Applications.....	98
7.3.1	Sorting.....	98
7.3.2	Searching.....	98
7.3.3	Priority Queue.....	98
7.4	Memory.....	99
7.4.1	Dynamic.....	99
7.4.2	Static (fixed size).....	99
7.5	Consideration of curriculum.....	99
Chapter 8: Conclusion and Future work.....		100
REFERENCES .....		102
APPENDICES .....		106

## List of Codes

Code listing 3.1 Fixed Stack Dynamic arrays in C++ .....	12
Code listing 3.2 Stack Dynamic arrays in ADA .....	12
Code listing 3.3 Fixed heap dynamic arrays in C++.....	13
Code listing 3.4 Fixed heap dynamic arrays in JAVA.....	13
Code listing 3.5 Heap Dynamic arrays in Perl.....	13
Code listing 3.6 Jagged Array.....	16
Code listing 3.7 Jagged Array in C++ .....	17
Code listing 3.8 Charter type array subscript in Pascal .....	18
Code listing 3.9 Charter type array subscript in Pascal .....	18
Code listing 3.10 Enumeration in C++ .....	19
Code listing 3.11 Matrix ADT in C++.....	20
Code listing 3.12 Vectors in C++ .....	20
Code listing 3.13 ArrayList in C++ .....	21
Code listing 3.14 Generic ArrayList in C++.....	21
Code listing 4.1 Singly linked list in C++ .....	25
Code listing 4.2 Doubly linked list in C++ .....	26
Code listing 4.3 Circular linked list in C++.....	27
Code listing 4.4 ADT-insert node to end of linked list.....	29
Code listing 4.5 Generic linked list in C++ .....	31
Code listing 4.6 ADT-insert node to end of linked list.....	31
Code listing 4.7 Sentinel singly linked list in C++ .....	33
Code listing 4.8 Sentinel doubly linked list C++.....	34
Code listing 4.9 Persistent singly linked list in C++.....	38
Code listing 4.10 Array of linked list in C++ .....	39
Code listing 4.11 Levelled Skip lists in C++ .....	41
Code listing 5.1 Array based stack in C++ .....	48
Code listing 5.2 Linked list based stack in C++ .....	50
Code listing 5.3 ADT stack (array based) in C++ .....	51
Code listing 5.4 ADT stack (Linked list based) in C++ .....	52
Code listing 5.5 Generic stack (array based) in C++ .....	52
Code listing 5.6 Generic stack (linked list based) in C++ .....	53
Code listing 5.7 Adapter through inheritance in C++.....	54
Code listing 5.8 Adapter through composition in C++.....	55
Code listing 6.1 Array based (Structured) queue in C++ .....	63
Code listing 6.2 Linked list based (Structured) queue in C++.....	64
Code listing 6.3 ADT queue (array based) in C++ .....	65
Code listing 6.4 ADT queue (linked list based) in C++ .....	66
Code listing 6.5 Generic queue (array based) in C++.....	67
Code listing 6.6 Generic Queue (Linked list based) in C++.....	68
Code listing 6.7 Adapter through inheritance in C++.....	69

Code listing 6.8 adapter through composition in C++ .....	70
Code listing 6.9 Circular queue (array based) in C++ .....	72
Code listing 6.10 Priority queue (array based) in C++ .....	73
Code listing 7.1 Complete Binary tree in C++ .....	80
Code listing 7.2 Binary search Tree in C++ .....	85
Code listing 7.3 A slightly modified BST in C++ .....	90
Code listing 7.4 RED and Black in C++.....	91
Code listing 7.5 A slightly modified BST in C++ .....	93
Code listing 7.6 AVL tree in C++.....	95

## List of Figures

Figure 1.1: Flowchart showing research design methods and research approach followed in the study.....	4
Figure 3.1: Array topics covered in CS2 .....	11
Figure 3.2 Logical view .....	14
Figure 3.3 Physical view.....	14
Figure 3.4 Logical view .....	15
Figure 3.5 Physical view.....	15
Figure 3.6 Jagged Matrix .....	17
Figure 3.7 Symmetric Matrix.....	17
Figure 4.1 Linked list topics .....	24
Figure 4.2 (a) Node of singly linked list. (b) Linked List generated by Code Listing 1 .....	25
Figure 4.3 (a) Node of doubly linked list. (b) Linked List generated by Code Listing 2 .....	26
Figure 4.4 Circular linked list.....	27
Figure 4.5 Circular doubly linked list.....	27
Figure 4.6 All possible states of link list by calling <i>addonTail(10)</i> on empty linked list (code listing 4.3). (a) Head and tail are null. (b) Assign head to new node (line no 15). (c) Assign data 10 to head of new node (line no 16). (d) Assign tail to head (line no 17). .....	28
Figure 4.7 All possible states of linked list by calling <i>addonTail(30)</i> on non-empty linked list (code listing 4.3). (a) Initial linked list and assign tail to head. (b) Add new node at tail (line no 20). (c) Move tail to tail of next (line no 21). (d) Assign 30 as data and tail to head (line no 22-23). .....	28
Figure 4.8 UML of ADT singly linked list.....	29
Figure 4.9 All possible states of linked list by calling <i>addonTail(10)</i> on empty linked list (code listing 4.4). (a) Head and tail are null. (b) Assign head to new node (line no 13). (c) Assign data 10 to head of new node (line no 14). (d) Assign tail to head (line no 15). (e) Assign null at tail (line no 22). .....	29
Figure 4.10 All possible states of link list by calling <i>addonTail(30)</i> on non-empty linked list (code listing 4.4). (a) Initial linked list. (b) Add new node at tail of next (line no 18). (c) Move tail to tail of next (line no 19). (d) Assign 30 as data and null at tail (line no 20-22). .....	30
Figure 4.11 UML of ADT Doubly linked list.....	30
Figure 4.12 All possible states of link list by calling <i>addonTail(10)</i> on empty linked list (code listing 4.6). (a) Initial linked list. (b) Add new node on <i>tail</i> of next (line no 14). (c) Assign 10 as data to <i>head</i> node (line no 15). (d) Assign <i>tail</i> to <i>head</i> (line no 16). (e) Assign null to head of <i>prev</i> (line no 18). (f) Assign null at <i>tail</i> (line no 25). .....	31
Figure 4.13 UML of generic singly linked list node .....	31
Figure 4.14 UML of generic singly linked list.....	31
Figure 4.15 All possible states of link list by calling <i>addonTail(30)</i> on non-empty linked list (code listing 4.6). (a) Initial link list. (b) Add new node at tail of next (line no 20). (c) Move tail to tail of next of <i>prev</i> (line no 21). (d) Move tail to tail of next (line no 22). (e) Assign data 30 to tail (line no 23). (f) Assign null at tail (line no 25). .....	32

Figure 4.16 All possible states of linked list by calling <i>addonTail(10)</i> on empty linked list(code listing 4.7). (a) Initial linked list. (b) Add new node at tail (line no 16). (c) Assign 10 at tail (line no 17). (d) Move tail to tail of next (line no 18). (e) Assign null at tail (line no 19). .....	33
Figure 4.17 All possible states of linked list by calling <i>addonTail(30)</i> on non-empty linked list (code listing 4.7). (a) Initial linked list. (b) Add new node at tail of next. (c) Assign 30 as data at tail. (d) Move tail to tail of next. (d) Assign null to tail of next.....	34
Figure 4.18 All possible states of linked list by calling <i>addonTail(10)</i> on empty linked list(code listing 4.8). (a) Initial linked list. (b) Add new node at tail of next (line no 21). (c) Assign 10 at tail (line no 22). (d) Move tail to tail of next of prev (line no 23). (e) Move tail to tail of next and assign null at tail (line no 24-25). .....	34
Figure 4.19 All possible states of linked list by calling <i>addonTail(30)</i> on non-empty linked list. (a) Initial linked list. (b) Add new node after the tail. (c) Assign 30 as data at tail. (d) Move tail to tail of next. (d) Assign null to tail of next.....	35
Figure 4.20 Array based linked list with different add and delete operations .....	36
Figure 4.21 Persistent linked list states (a) linked list in memory (b) Data of linked list store in disk.....	38
Figure 4.22 Ordered list .....	40
Figure 4.23 Array of Linked list .....	40
Figure 4.24 Unlevelled skip lists .....	41
Figure 4.25 Levelled Skip lists (searching for 35).....	42
Figure 5.1: Taxonomy of stack based on its implementation .....	46
Figure 5.2 Array based (fixed stack) size (code listing 5.1). (a) Initial empty stack (line no 3). (b) Push 10 as element onto stack with <i>tos=0</i> (line no 35). (c) Push 20 as element onto stack with <i>tos=1</i> (line no 36). (d) Pop 20 as an element from stack with <i>tos=0</i> (line no 38) .....	48
Figure 5.3: Linked list based (dynamic stack) size (code listing 5.2). (a) Initial stack (b) push value of 'a' on stack (c) push value of 'b' on stack (d) pop value of 'a' from stack (e) pop value of 'b' from stack.....	49
Figure 5.4 UML diagram of Stack Class .....	50
Figure 5.5: UML diagram of Stack Generic class .....	51
Figure 5.6: UML diagram of implementing stack using inheritance with <i>LinkedList</i> .....	54
Figure 5.7: UML diagram for implementation of Stack using composition of <i>LinkedList</i> .....	55
Figure 6.1: Taxonomy of queue based on its implementation.....	61
Figure 6.2 Array based (fixed size queue) (Code listing 6.1, line no 32-36) (a) Initial empty queue(b) <i>enQueue</i> 10 as an element and increment 1 in tail (c) <i>enQueue</i> 20 as an element and increment 1 in tail (d) <i>enQueue</i> 30 as an element and increment 1 in tail.....	62
Figure 6.3 : Array based (fixed size queue) (Code listing 6.1, line no 37-41)(a)initial queue (b)First <i>deQueue</i> operation returns 10 in variable 't' and left shift all elements(c)Second <i>deQueue</i> operation returns 20 in variable 't' and left shift all elements(d) Third <i>deQueue</i> operation returns 30 in variable 't' and left shift all elements .....	63
Figure 6.4 Linked list based (dynamic sizequeue) (Code listing 6.2, line no 29-37) (a)Initial queue (b) <i>enQueue</i> 10 inserts an element before tail (c) <i>enQueue</i> 20 inserts an element before tail (d) <i>enQueue</i> 30 inserts an element before tail .....	64

Figure 6.5 Linked list based (dynamic sizequeue) (Code listing 6.2, line no 38-42) (a) Initial queue (b) First <i>deQueue</i> operation returns 10 in variable ‘t’ and remove second node (c) Second <i>deQueue</i> operation returns 20 in variable ‘t’ and remove second node (d) Third <i>deQueue</i> operation returns 30 in variable ‘t’ and remove second node.....	64
Figure 6.6 UML diagram of Queue Class.....	65
Figure 6.7 UML diagram of Queue Generic class.....	67
Figure 6.8 UML diagram of implementing queue using inheritance with LinkedList.....	68
Figure 6.9 UML diagram for implementation of Queue using composition of LinkedList .....	69
Figure 6.10 Circular queue (array based) (Code listing 6.9, line no 49-53)(a) Initial queue (b) <i>enQueue</i> 10 as an element and increment 1 in tail and size (c) <i>enQueue</i> 20 as an element and increment 1 in tail and size (d) <i>enQueue</i> 30 as an element and increment 1 in tail and size...	71
Figure 6.11 Circular queue (array based) (Code listing 6.9, line no 55) <i>deQueue</i> operation returns 10 in variable ‘t’ .....	71
Figure 6.12 Circular queue (array based) (Code listing 6.9, line no 57) <i>enQueue</i> 40 as an element and increment 1 in size .....	71
Figure 6.13 UML diagram of Priority Queue Class .....	72
Figure 7.1 Taxonomy of binary tree .....	76
Figure 7.2: Number of nodes at depth $i^{\text{th}}$ of Binary tree .....	78
Figure 7.3: Complete binary tree for Code Listing 7.1 .....	80
Figure 7.4: (a) Node of Binary Search tree (b) Binary Search tree .....	80
Figure 7.5: UML diagram of binary search tree class .....	81
Figure 7.6: Insert node ‘17’ in BST .....	81
Figure 7.7: Memory diagram of insertion in Binary search tree generated by Code listing 7.2...	82
Figure 7.8: (a) Delete node (if no child) in binary Search tree (b): Delete node (if one child) in binary Search tree (c): Delete node (if two children) in binary Search tree .....	83
Figure 7.9: Search node in binary Search tree .....	84
Figure 7.10 Predecessor and successor of node 5 in binary tree .....	84
Figure 7.11: Red Black Tree .....	86
Figure 7.12 Left rotation in RB tree.....	87
Figure 7.13 Right rotation in RB tree .....	87
Figure 7.14: Insertion in RB tree .....	89
Figure 7.15: Insertion in AVL tree .....	96
Figure 7.16: Rooted binary tree .....	97
Figure 7.17: Full Binary tree.....	97
Figure 7.18: Perfect binary tree .....	97
Figure 7.19: Strictly binary tree .....	97
Figure 7.20: Degenerated tree.....	98
Figure 7.21: Extended binary tree.....	98

## List of Tables

Table 3.1: Suggested Considerations of curriculum for teaching the topics related to arrays.....	21
Table 4.1: Suggested importance of the topics under each category .....	43
Table 5.1: Formal Specification of Stack in VDM.....	45
Table 5.2: Signatures of major services/methods implemented in the Stack .....	47
Table 5.3 : Recommendation Level of Each Variant .....	56
Table 6.1 : Formal Specification of Queue in VDM .....	59
Table 6.2: Signatures of major services/methods implemented in the Queue.....	61
Table 6.3: Recommendation Level of Each Variant.....	74
Table 7.1 Signatures of major services/methods implemented in the Binary tree.....	76
Table 7.2 : Suggested considerations for teaching the topics related to binary tree.....	99

## **Abstract**

The Linear and nonlinear Data Structures are core part of computer science or curriculum. Linear data structure covered all types of consecutive indexed and reference wise ordinal data structures. For example, array, linked list and linear collection of abstract data types. We cover binary tree in nonlinear data structure. All these topics were covered in most of CS related core courses and included in data structure text book. Major issue is that what topic is to be covered and what subtopics is to be taught by undergraduate students. This study propose classification of topics and subtopics in linear and nonlinear data structure. Major contribution of this research is to figure out all variants of linear and nonlinear data structure topics and provide standard taxonomies. Each dimension has explained in terms of structured, object oriented, memory diagrams, applications, implementation and complexity and have also discussed its application using design patterns. It will facilitate students to learn and visualize more about linear and nonlinear data structure concepts and for teachers to revise their course outlines, implement standard code and visualize memory diagrams.