COMPUTER PROGRAMMING
Unit 5 – Teaching guide
Linear Data Structures

Jon Ander Gómez Adrián, Marisa Llorens Agost
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
{jon,mllorens}@dsic.upv.es

January 25, 2018

1 Contents

2 Specific objectives

Upon completion of this subject the student should be able to ...

Objective 1 define the concept of data structure (DS).

Objective 2 distinguish between the internal implementation and the external use of a DS.

Objective 3 define the features of a linear data structure (LDS).

Objective 4 define the stack as a linear data structure.

Objective 5 enumerate possible uses of stacks.

Objective 6 define the operations for manipulating stacks.

Objective 7 use stacks in his programs for solving problems.

Objective 8 implement stacks with different ways of internal data storage using Object Oriented Programming (OOP). The data structures commonly used are arrays and linked structures.

Objective 9 compute the temporal cost function of each stack operation.

Objective 10 apply the knowledge about internal data storage of stacks for implementing new operations.

Objective 11 propose alternative implementations of stacks in order to fulfil complexity requirements in a particular problem.

Objective 12 define the queue as a linear data structure.

Objective 13 enumerate possible uses of queues.

Objective 14 define the operations for manipulating queues.

Objective 15 use queues in his programs for solving problems.

Objective 16 implement queues with different ways of internal data storage (arrays and linked structures) using OOP.

Objective 17 compute the temporal cost function of each queue operation.

Objective 18 apply the knowledge about internal data storage of queues for implementing new operations.

Objective 19 propose alternative implementations of queues in order to fulfil complexity requirements in a particular problem.

Objective 20 define the list as a linear data structure.

Objective 21 enumerate possible uses of lists.

Objective 22 define the operations for manipulating lists.

Objective 23 use lists in his programs for solving problems.

Objective 24 implement lists with different ways of internal data storage (arrays and linked structures) using OOP.

Objective 25 compute the temporal cost function of each operation over lists.

Objective 26 apply the knowledge about internal data storage of lists for implementing new operations.

Objective 27 propose alternative implementations of lists in order to fulfil complexity requirements in a particular problem.

References

[Eck 2006] "Introduction to Programming Using Java", fifth edition
Eck, D.J., 2006
`http://math.hws.edu/javanotes/`
Chapter 9, sections 9.2 and 9.3

[Garcia 2005] "Una introducción a la programación. Un enfoque algorítmico",
García, J.J. et al.
Ed. Thomson, 2005.
Chapter 8, section 8.4 except subsection 8.4.4

[Joyanes 2001] "Java 2: Manual de programación",
Joyanes, L. y Fernández, M.
Ed. McGraw-Hill, 2001.
Chapter 15

[Weiss 2006] "Estructuras de datos en Java: compatible con Java 2"
Mark Allen Weiss
Ed. Addison Wesley, 2000 - 2006
Chapters 6, 15 and 16

[Arnold 2001] "El lenguaje de programación Java"
Arnold, K. et al
Ed. Addison Wesley, 2001
Chapter 21

[Cormen 2007] "Introduction to Algorithms" 2nd edition
T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein
The MIT Press – Cambridge, Massachusetts, 2007
Chapters 6, 10 and 11

3 Planning of each session

Duration of activities

	Inside classroom	Outside classroom
Before	–	2h
Session 1	1.5h	2h
Session 2	1.5h	2h
Session 3	1.5h	2h
Session 4	1.5h	2h
Session 5	1.5h	2h
Session 6	1.5h	2h
Session 7	1.5h	2h
Session 8	1.5h	2h
Session 9	1.5h	–
	13.5h	18h

Before the first session

Outside classroom activities (up to 2h)

- Students must read before the first class some basic concepts concerning linear data structures, their usefulness in more complex algorithms, and the significance of knowing to decide which internal data storage is more appropriate depending on the problem to solve.
- We recommend a previous reading of at least two of the following references.
  - Section 9.2 from [Eck 2006].
  - Section 8.4 (except 8.4.4) from [Garcia 2005].
  - Section 6.1 from [Weiss 2006].

First session

Classroom activities (1h 30')

Minutes	Activity
25'	The teacher explains. Introduction to linear data structures. Students will have available a copy of the slides to take notes.
5'	Break.
25'	The teacher explains. Representing linked sequences.
5'	Break.
20'	The teacher explains. Representing linked sequences (II).
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of basic concepts explained in class by reading the sections referenced next. The concepts to review are: representation of linked sequences by means of linear data structures, where one object references the next one. Books for consulting:
  - Section 9.2 from [Eck 2006].
  - Sections 15.2 and 16.1 from [Weiss 2006].

Second session

Classroom activities (1h 30')

Minutes	Activity
20'	The teacher explains. Stacks as linear data structures. Stack operations.
5'	Break.
20'	Work groups implement the `Stack` class by using arrays for internal data storage.
5'	Break.
30'	Students and teacher discuss about the details of implementation of stacks with arrays, and about the temporal cost of each operation using stacks.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of the implementation of stacks with arrays and study of how to implement them by using dynamic memory. Besides reading the sections referenced above, we also recommend you reading the following ones:
  - Sections 9.2 and 9.3 from [Eck 2006].
  - Sections 15.6 from [Joyanes 2001].
  - Sections 6.2 and 15.1.1 from [Weiss 2006].

Third session

Classroom activities (1h 30')

Minutes	Activity
25'	Work groups implement the stack class by using dynamic memory.
5'	Break.
25'	Students and teacher discuss about the details of implementation of stacks with dynamic memory, and about the temporal cost of each operation using stacks.
5'	Break.
20'	Posing of problems whose solution needs the use of stacks.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of dealt concepts and study of new ones by reading the proposed references and the following ones:
  - Section 9.3 from [Eck 2006].
  - Section 15.2.1 from [Joyanes 2001].

- Deliverable #5.1 – To be done by work groups.

  It must be implemented a program which loads in a stack all the integer values stored in an ASCII file. Then this program separates the values into two new stacks, one stack with the positive values and the other stack with the negative values. Finally, it uses the new stacks to display the integers in two columns, the negatives values in the left column and the positives ones in the right column. For example, if the data file is:

  datos.in

  ```
  -756 -536 109 92 394 421 253 240 774 -761
  ```

  the output of executing `java Separa -input datos.in` must be

  stdout

  ```
      -756        109
      -536         92
      -761        394
                  421
                  253
                  240
                  774
  ```

  where if one stack gets empty before the other, the program must fill out with white spaces the place of the number in order to achieve that the values appear vertically aligned. Please, observe how the values appear in the same order that they are stored in the input file. You have not to perform any sorting operation, your program must hold the original order. You will find in resources of *PoliformaT* the following two compiled classes, `GeneraDatos.class` for generating your own samples of input data, and `Separa.class` to test if the output of your program matches the correct one.

  This work must be submitted by email before the fifth session. Students must send one or more Java files with the source code, including the comments on the attributes (instance variables) and the methods in the format required by the `javadoc` tool. All programs must compile without errors.

  This deliverable should include, by way of a logbook, a record of work group meetings, day, start time and end time, the members who attended, what was discussed, and which tasks were assigned to each group member. Besides working on the problem each student must play a different role in each task: (1) secretary, (2) task scheduler and time controller, and (3) responsible for finding and preparing the necessary documentation. Roles must be rotating.

Fourth session

Classroom activities (1h 30')

Minutes	Activity
5'	Resolution of doubts about deliverable #5.1
20'	The teacher explains. Queues as linear data structures. Operations on queues.
5'	Break.
20'	Implementation of a FIFO queue using arrays for the internal data storage by way of a circular buffer.
5'	Break.
25'	Students and teacher discuss about the details of implementation of queues with arrays, and about the temporal cost of each operation using queues.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of the implementation of queues with arrays and study of how to implement them by using dynamic memory. Besides reading the sections referenced above, we also recommend you reading the following ones:

  - Section 9.3 from [Eck 2006].
  - Section 15.7 from [Joyanes 2001].
  - Sections 6.3 and 15.1.2 from [Arnold 2001].
- Work groups meet for completing the deliverable #5.1

Fifth session

Classroom activities (1h 30')

Minutes	Activity
25'	Work groups implement the queue class by using dynamic memory.
5'	Break.
25'	Students and teacher discuss about the details of implementation of queues with dynamic memory, and about the temporal cost of each operation using queues.
5'	Break.
20'	Posing of problems whose solution needs the use of queues.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of dealt concepts and study of new ones by reading the detailed sections of proposed books:
  - Section 9.3 from [Eck 2006].
  - Section 15.2.2 from [Weiss 2006].
- Deliverable #5.2 – To be done by work groups.

  It must be solved the same problem posed in the deliverable 5.1, however, in this case the solution must use queues instead of stacks. Do you think that this version of the program will display the integer values in the same order they are showed in the deliverable 5.1? Remember, the values should be displayed in the same order they are in the input file. The implementation of the queue class must use linked structures, no arrays.

  This work must be submitted by email before the seventh session. Students must send one or more Java files with the source code, including the comments on the attributes (instance variables) and the methods in the format required by the `javadoc` tool. All programs must compile without errors.

  This deliverable should include, by way of a logbook, a record of work group meetings, day, start time and end time, the members who attended, what was discussed, and which tasks were assigned to each group member. Besides working on the problem each student must play a different role in each task: (1) secretary, (2) task scheduler and time controller, and (3) responsible for finding and preparing the necessary documentation. Roles must be rotating.

Sixth session

Classroom activities (1h 30')

Minutes	Activity
20'	The teacher explains. Lists as linear data structures. Operations on lists.
5'	Break.
20'	Implementation of lists by using arrays for internal data representation.
5'	Break.
25'	Students and teacher discuss about the details of implementation of lists with arrays, and about the temporal cost of each operation using lists.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Review of the implementation of lists with arrays and the operations on lists.
- Study and review of all the concepts dealt in this unit for preparing the last deliverable. We recommend you consulting the following references:
  - Sections 9.2 and 9.3 from [Eck 2006].
  - Sections 15.1, 15.2, 15.3, 15.4 and 15.5 from [Joyanes 2001].
  - Sections 6.4, 16.1, 16.2 and 16.3 from [Weiss 2006].

Seventh session

Classroom activities (1h 30')

Minutes	Activity
20'	Work groups implement the `List` class by using dynamic memory and specify the temporal cost of each operation.
5'	Break.
25'	Implementation of the `List` class doubly linked. Benefits it can provide as needed: traversing in reverse order. Temporal cost of each operation on lists.
5'	Break.
25'	Implementation of the *selection-sort* algorithm using a doubly linked list.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- We recommend you reading the following sections of books concerning heaps:
  - Sections 6.1 and 6.2 from [Cormen 2007].
- Deliverable #5.3 – To be done by work groups.

  Complete implementation of both sorting algorithms *insertion-sort* and *merge-sort* by using doubly linked lists. The implementation of the *merge-sort* algorithm implies implementing the *natural-merge* algorithm. You should prepare sample data files for testing purposes, for what you can use the program `GeneraDatos.class` that we provided to you in the deliverable #5.1.

  This work must be submitted by email before the last session. Students must send one or more Java files with the source code, including the comments on the attributes (instance variables) and the methods in the format required by the `javadoc` tool. All programs must compile without errors.

  This deliverable should include, by way of a logbook, a record of work group meetings, day, start time and end time, the members who attended, what was discussed, and which tasks were assigned to each group member. Besides working on the problem each student must play a different role in each task: (1) secretary, (2) task scheduler and time controller, and (3) responsible for finding and preparing the necessary documentation. Roles must be rotating.

Eighth session (additional contents for ARA group)

Classroom activities (1h 30')

Minutes	Activity
30'	The teacher explains. Heaps. The **(binary) heap** data structure, and the basic operations.
5'	Break.
20'	Maintaining the heap property thanks to the `maxHeapify()` subroutine.
5'	Break.
20'	Building a heap from an array by using the `buildMaxHeap()` subroutine.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Work groups are still working on tasks related to the deliverable #5.3
- We recommend you reading the following sections of books concerning hash tables:
  - Sections 11.1, 11.2 and 11.3 (avoiding the proof of theorems) and from [Cormen 2007].

Ninth session (additional contents for ARA group)

Classroom activities (1h 30')

Minutes	Activity
30'	The teacher explains. Hash tables. The data structure of hash tables.
5'	Break.
20'	Resolution of collisions.
5'	Break.
20'	Hash functions.
10'	Resolution of doubts.

Outside classroom activities

- No activities are planned after the last session of this unit, consult the teaching guide for the next unit, where you will find activities planned before the first session.

4 Issues and problems that, minimally, should be studied in this unit

- Implementation of stacks (LIFO LDS) with arrays and linked structures.
- Implementation of queues (LIFO LDS) with arrays and linked structures.
- Implementation of lists with arrays and linked structures, simply or doubly linked.
- Study of the behaviour of the basic operations on lists, stacks and queues, in either modality of internal data representation, arrays or linked structures.
- Implementation of at least two sorting algorithms on lists.

5 Deliverables

Id	Description	Modality	Limit date	Score
#5.1	Separation of positive and negative integers by using stacks (LIFO). It must be delivered the source code and a text file (or PDF) with the logbook.	Group	By electronic mail before the fifth session.	80
#5.2	Separation of positive and negative integers by using queues (FIFO). It must be delivered the source code and a text file (or PDF) with the logbook.	Group	By electronic mail before the seventh session.	80
#5.3	Two sorting algorithms with doubly linked lists. It must be delivered the source code and a text file (or PDF) with the logbook.	Group	By electronic mail before the ninth session.	100

6 Qualification

The grade for all activities performed in this unit will be computed as the sum of the score of each deliverable. The grade of activities performed by the work group will be the same for each member that has participated.

The knowledge of this subject will be assessed in the quizzes and in the lab practises, the grade obtained in the deliverables described herein is used to measure the work of formation, and it is part of the NAS (Grade of the Follow-up Activities, the Spanish acronym is used), which, in turn, as said in the evaluation rules, contributes to the final grade with a 20%.