AN E-LEARNING PACKAGE FOR DATA STRUCTURE USING VISUAL BASIC

Yasmin M. Kassim
Ass. Lec. In Nahrain University

ABSTRACT

E-learning comprises all forms of electronically supported teaching and learning. The information and communication system, whether networked or not, serve as specific media to implement the learning process; Computer based learning is one of the applications and processes of E-learning, sometimes abbreviated to CBL, refers to the use of computers as a key component of the educational environment [2].
Through my teaching, I found that there is a need for software learning packages at the lab to help students understand their courses especially in the subjects that need imagination like data structure. This research presents a data structure learning package for array and linear structure. Linear structure covers non-linked structure (stack and queue) and linked structure with their kinds.Through out this project the students can obtain a demonstration of how data structure is represented in memory. Most of the windows consist of three tabs, the first one is the information tab which contains all the information about the structures. The second tab contains all the operations performed on it and there is an ability to display all the operations in one executable program (full program), which receives inputs from the students, the students can enter different inputs according to the choices displayed and produce output after implementing by C++ program. The third tab contains exercises for the students to test their ability in understanding and after completing all the questions the package gives a score depending upon solutions. The package includes pictures and demos (movies) that represent the data structure implemented with 3D Max, finally all the windows are supplied with voice (my voice) to teach the students how to use the package.

الحقيبة التعليمية الالكترونية لمادة هياكل البيانات باستخدام الفيجوال بيسك

الخلاصة

التعليم الالكتروني يشمل جميع نماذج التعليم و التدريس الالكتروني. نظام المعلومات و الاتصالات سواء كان متصلا بشبكة أو غير متصل فهو يخدم كواسطة لتنفيذ عملية التعليم. أحد أنواع التعليم الالكتروني هو التعليم بواسطة الحاسبة، أحيانا يسمى بال (CBL) وهو يشير إلى استعمال الحاسبات كعنصر أساسي في عملية التعليم.
من خلال تدريسي لاحظت وجود حاجه لحقيبة تعليمية في المختبر لمساعدة الطلاب لفهم المقررات التعليمية و خاصة في المواضيع التي تحتاج إلى تخيل كمادة هيكلة البيانات. لذلك هذا البحث يقدم حقيبة تعليمية خاصة بهيكلة البيانات و التي تغطي المصفوفة و الهياكل الخطية التي تضم الهياكل الغير متصلة (المكدس و الطابور) والهياكل المتصلة بأنواعها. من خلال هذه الحقيبة يتمكن الطلاب من الحصول على شرح مفصل عن كيفية تمثيل هياكل البيانات في الذاكرة وان معظم الواجهات في الحقيبة تتكون من ثلاث واجهات متغيرة، الأولى هي واجهة المعلومات و التي تتضمن جميع المعلومات عن الهيكل، والثانية تعرض جميع العمليات الخاصة بالهيكل مع وجود إمكانية لعرضها جميعها في برنامج واحد قابل للتنفيذ من خلال الحقيبة بواسطة برنامج ال (C++) حيث يتمكن الطالب من وضع المدخلات الخاصة بتنفيذ البرنامج و المعروضة من خلال اختيارات في الواجهة وبعد التنفيذ تظهر النتائج أيضا في الواجهة و الواجهة الثالثة تحتوي على تمارين للطالب لمعرفة مدى استيعابه للمادة و بعد إكمال جميع الأسئلة يعرض البرنامج الدرجة التي حصل عليها بالاعتماد على إجابته. كذلك تحتوي الحقيبة على صور و فلم يوضح آلية خزن الهياكل للبيانات مصممة بال(3D MAX)، و أخيرا جميع الواجهات مزودة بالية الصوت (صوت معد هذا البحث) لتعليم الطلبة كيفية استخدام الحقيبة.

Keywords: Learning Package (LP), Learning Package Platform (LPP), Electronic Learning (E-learning).

1. INTRODUCTION

During my teaching in computer and software engineering department, I found that there is a need for a software package that facilitates learning and understanding.

One of the fundamentals in software engineering study is data structure. Data Structure is any programming construction for representing and managing a collection of data. Data management in turns means storing and accessing data in a consistent way for a particular problem. The study of data structures involves two complementary goals. The first goal is to identify and develop useful mathematical entities and operation to identify what classes of problems using these entities and operations can solve. The second goal is to determine representation for those abstract entities and to implement the abstract operation on these concrete representations [9-11].

The learning package is implemented using Visual Basic because of the tools that permit the programmer to design graphical applications [5].

The package covers array (1D, 2D and ND), linear structure (non-linked structure and linked-structure), non-linked structure covered stack and queue (circular queue and double ended queue), linked structure covered linked list, circular linked list, double linked list, linked stack and linked queue. All of these are represented in a beautiful screens contain information, examples and exercises in addition to a movie to represent each of them.

2. LEARNING PACKAGE (LP) FRAMEWORK

This section presents a model of how LP can be integrated with a certain course.

2.1 Design model

The research has developed a design model of how to apply LP in a data structure through five steps as shown in fig. 1.

A. The platform is chosen according to the course content (here visual basic is chosen).

B. The design of exercises must reflect the limitations and constraints of the chosen LPP.

C. Start design and implementation according to the constraints within the course and LPP.

D. Documentation.

E. Analysis and Evaluation to learn from success and mistakes.

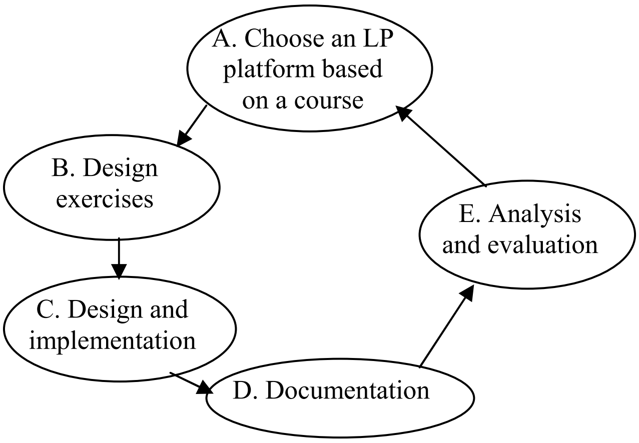

A. Choose an LP platform based on a course
B. Design exercises
C. Design and implementation
D. Documentation
E. Analysis and evaluation

Fig. 1
Model of designing an LP

3. IMPLEMENTATION OF DATA STRUCTURE LEARNING PACKAGE

This section contains a brief illustration of how to use the package, the main screen, and the function of each object. The complete design consists of six (6) forms:

1. Main Window.
2. Chapter 1 (Array).
3. Chapter 2 (Linear Structures).
4. Stack.
5. Queue.
6. Linked Structures.

Each form is supplied with "Menu Bar", "Tool Bar", and a "Status Bar". Voice comments are provided to facilitate the package usage.

3.1 Main Window

It contains the abstract information of the package as shown in fig. 2.

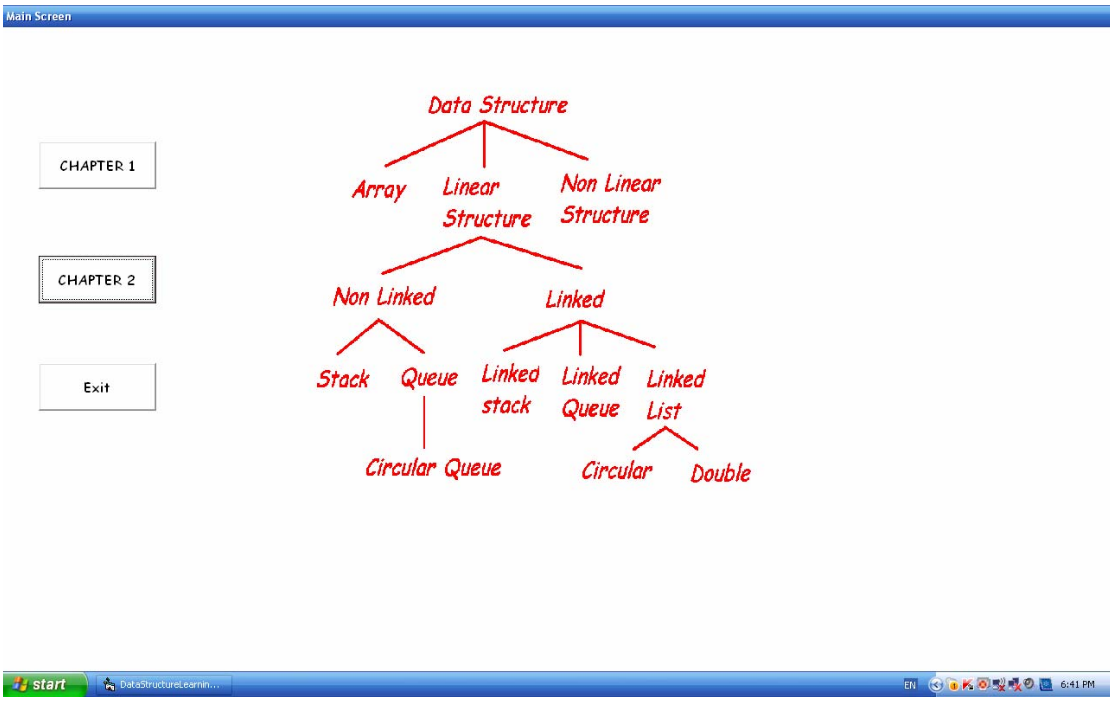


Fig. 2
Main Window

Here the package will begin to branch in to two chapters: Chapter 1 and Chapter 2. When you click on chapter1 a window will appear which contains all the information about the array.

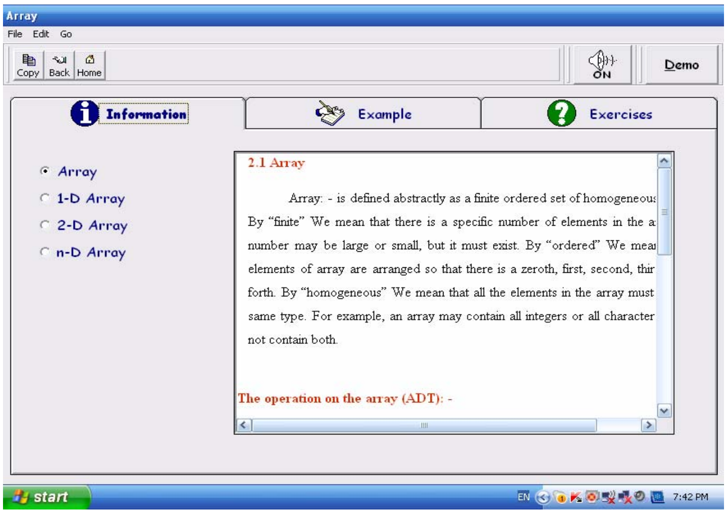
3.2 Chapter1 (Array)

The window contains three tabs and each of them is shown below:

1. Information tab: all the information about array includes one-dimensional array, two-dimensional array and n-dimensional array are presented in the rich text box and the chosen subject can be selected by an option buttons beside it as shown in fig. 3.
2. Example tab: which contains all the functions implemented on array (Create the array, destroy it, checking if it is empty, return it's length, return the location of a specific element, insert an element, remove an element and print the array), each of these operations is represented as a button and when you click on this button the corresponding program will appear in the rich text box beside the abstract data type ; also to test all the preceding operations a special button is placed at the bottom of the ADT field , when clicking on it a part of the form will change to make implementing the full program possible, the user may choose a function by writing his choice in the "choice box". If it's required to give input data, it must be written in the

"input box" and when finishing the user must enter 0 to exit then press the run button to execute the program, a message is displayed asking the user to wait for few seconds, the output is given in a textbox after pressing the show result as shown in fig. 4; The choices above include all the operations done on the array and the inputs of the user will be written on a file and this file will be the input for another C++ program; this program will be executed depending upon user input. The output is written to another file displayed by a text box in our package by clicking the show result button. An example is shown in fig. 4, entering an input in the choice textbox such as 55260 which means that the user first chooses a choice number 5 that represents the element insertion, when typing 5, the right textbox beside the choice box will be activated to receive the input to be inserted to the array (here the user inserts number 8), after that the textbox will be deactivated and the curser returns to the preceding textbox to receive the next choice which is also 5 (here the user inputs number 9), after inserting number 8 and 9 the user now wants to know the length of the array, so he chooses number 2 in the choice textbox then enters number 6 to print the array and 0 for exit after that the user must run the program by clicking on the run button and the output (execution result) will appear on the output textbox below show result.

3. Exercises tab: there are four questions with choices are given to test the user understandability, if the user fails in the current question, he can't return to it, the next button at the right lower corner will move him to the next question, a score is computed after attempting all questions as shown in fig. 5 here for example, the selected question is about the declaration of data structure, the student succeeds in this question by choosing B button, the program displays "correct answer" message to proceed in receiving questions.

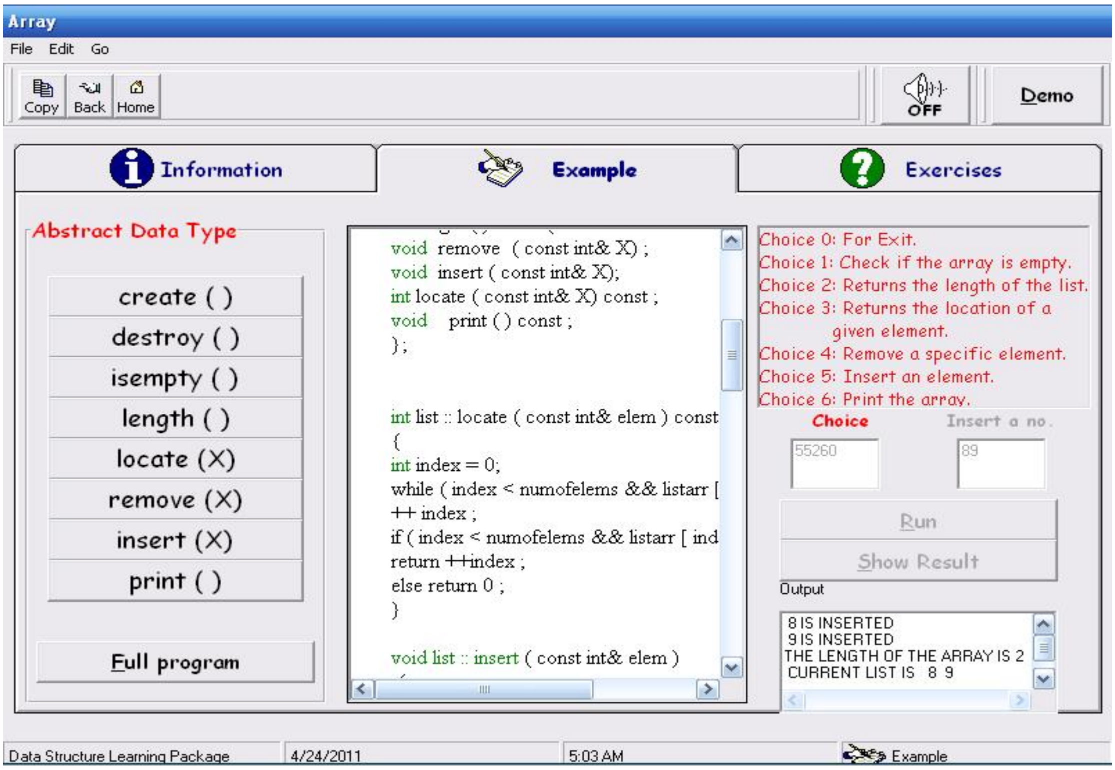


2.1 Array

Array: - is defined abstractly as a finite ordered set of homogeneous
By "finite" We mean that there is a specific number of elements in the a
number may be large or small, but it must exist. By "ordered" We mea
elements of array are arranged so that there is a zeroth, first, second, thir
forth. By "homogeneous" We mean that all the elements in the array must
same type. For example, an array may contain all integers or all character
not contain both.

The operation on the array (ADT): -

Fig. 3
Array information tab



```
void  remove  ( const int& X) ;
void  insert ( const int& X);
int locate ( const int& X) const ;
void    print ( ) const ;
};


int list :: locate ( const int& elem ) const
{
int index = 0;
while ( index < numofelems && listarr [
++ index ;
if ( index < numofelems && listarr [ ind
return ++index ;
else return 0 ;
}

void list :: insert ( const int& elem )
```

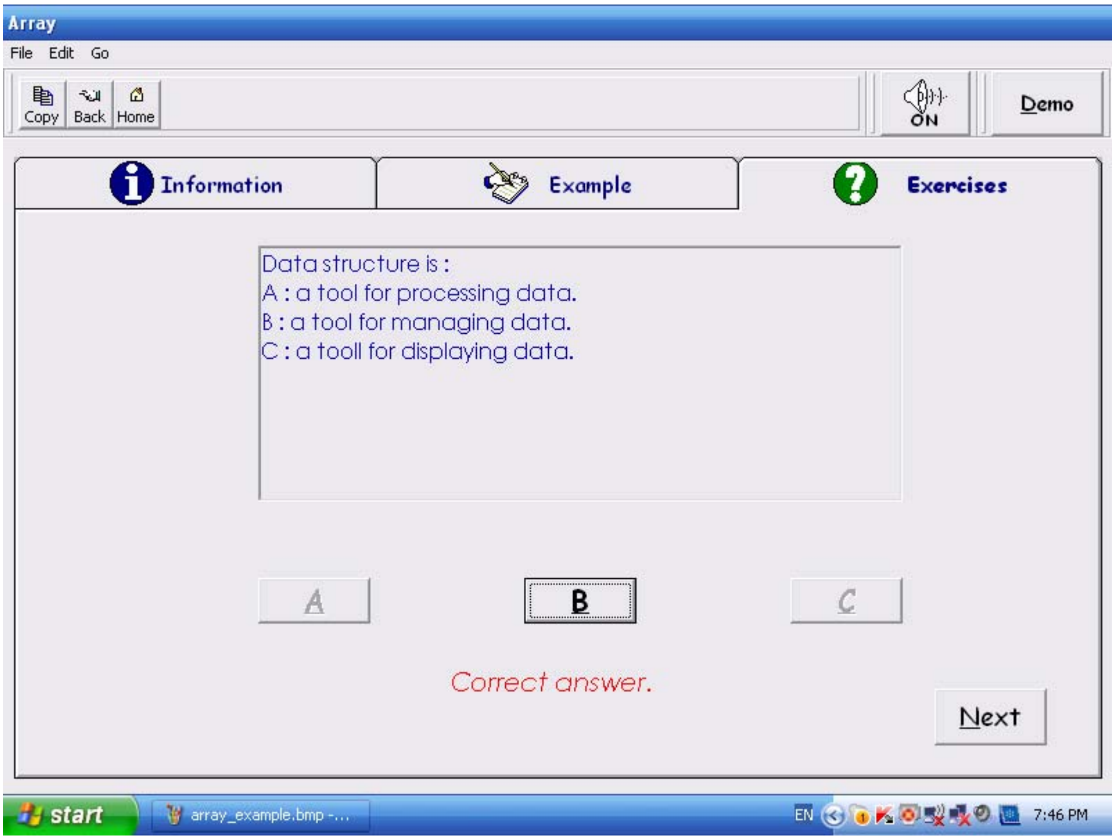
Choice 0: For Exit.
Choice 1: Check if the array is empty.
Choice 2: Returns the length of the list.
Choice 3: Returns the location of a given element.
Choice 4: Remove a specific element.
Choice 5: Insert an element.
Choice 6: Print the array.

Choice: 55260

Insert a no.: 89

```
 8 IS INSERTED
 9 IS INSERTED
THE LENGTH OF THE ARRAY IS 2
 CURRENT LIST IS   8  9
```

Fig. 4
Array Example tab



Data structure is :
A : a tool for processing data.
B : a tool for managing data.
C : a tooll for displaying data.

Correct answer.

Fig. 5
Array Exercise tab

Also there is a button called Demo, when the user clicks on it, a movie will appear to represent the array types as shown in fig. 6, the first part of the figure shown in blue color is to tell student that 1D array takes a linear place in memory, as for the second part of the same figure below it representing 2D in gray color shows the storage as a set of cells having a rectangular configuration and the third one with yellow color is for representing N-D storage.

The window also has a button showing a speaker icon for controlling the voice related to the package; also the widow contains the following buttons:

1. Copy: to copy any desired information to another document.
2. Back: to facilitate translation from screen to another screen .
3. Home: to return to home page.

Also the window contains a menu which has three elements:

1. File: includes the term "about" by clicking it, shows programmer's information and exit to exit from the package.
2. Edit: contains copy command.
3. Go: contains back and home page commands.

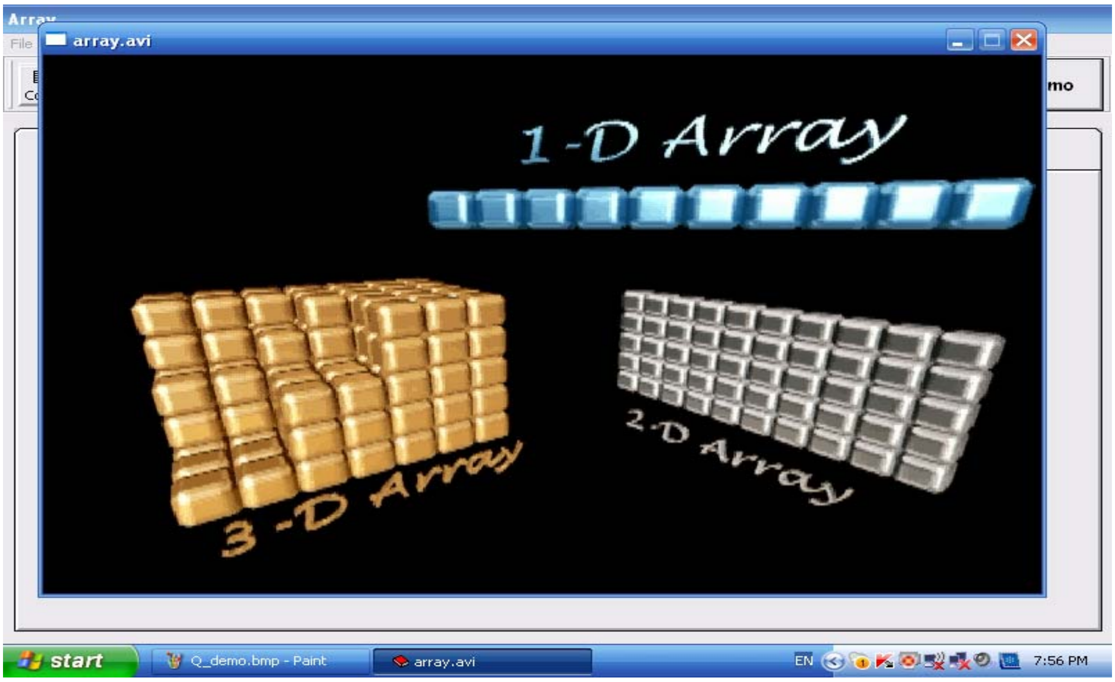
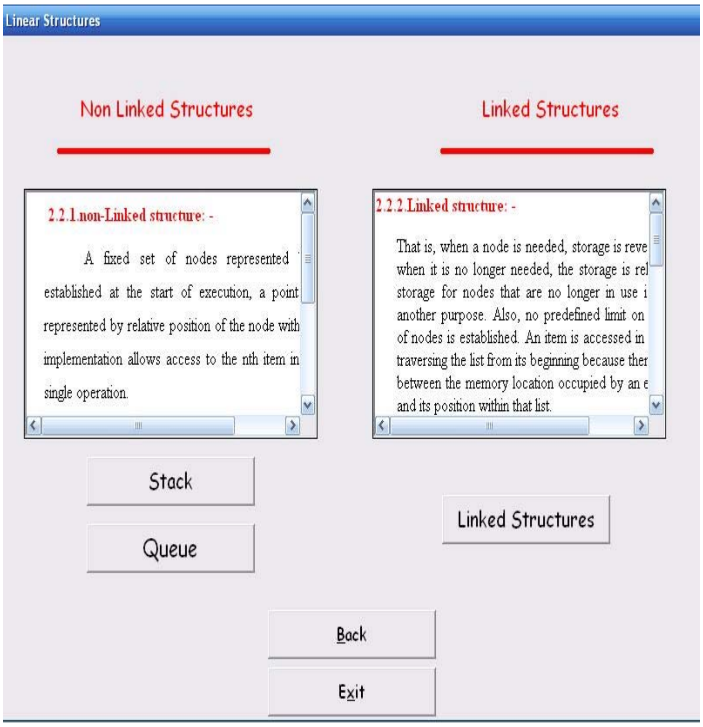


Fig. 6
Array picture from demo

Here we have completed chapter1 which includes the array only, after clicking on back, the package returns to the main window; here there is another button called chapter 2 which contains all the information about linear structure as listed below.

3.3 Chapter2 (Linear Structure)

This screen contains information about linear structure shown in fig. 7. Linear structure consists of non linked structure (Stack and Queue) and linked structure. There is a detailed explanation for each of them in the preceding window. It contains buttons called Stack, Queue and linked structure, when the user click on each of them, the package will go to the window attached to it. Also there is back and exit button to exit from the package.



2.2.1.non-Linked structure: -

A fixed set of nodes represented
established at the start of execution, a point
represented by relative position of the node with
implementation allows access to the nth item in
single operation.

2.2.2.Linked structure: -

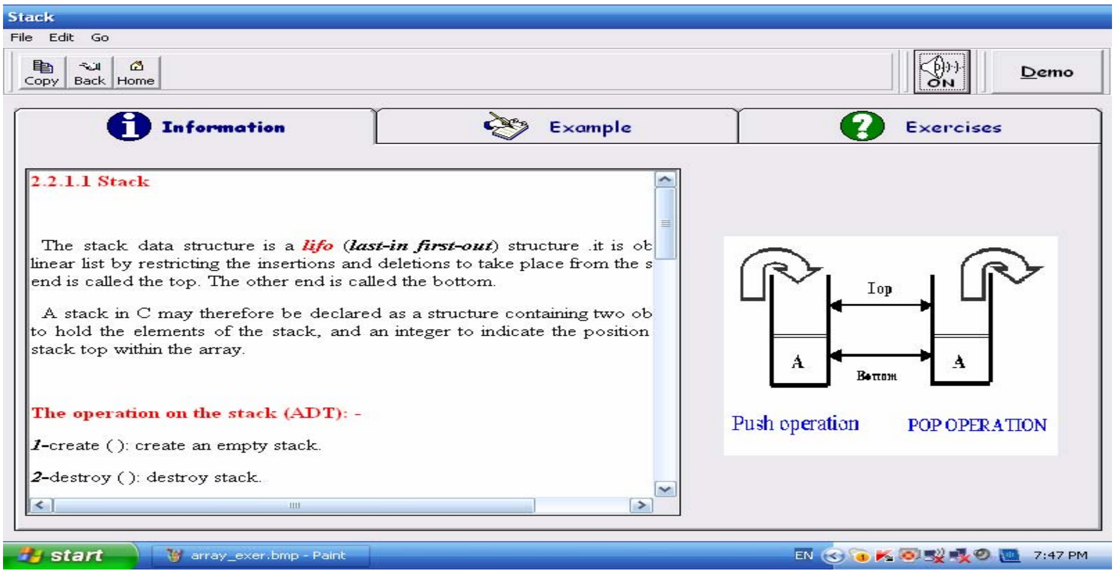
That is, when a node is needed, storage is reve
when it is no longer needed, the storage is rel
storage for nodes that are no longer in use i
another purpose. Also, no predefined limit on
of nodes is established. An item is accessed in
traversing the list from its beginning because ther
between the memory location occupied by an e
and its position within that list.

Fig. 7
Linear structure

3.4 Stack

The Stack data structure is a lifo (last-in first-out)structure. It is obtained from a linear list by restricting the insertions and deletions to take place from the same end. This end is called the top. The other end is called a bottom. This declaration and many other information about the stack are displayed in the stack screen in the package as shown in fig. 8.
In fig. 9 the user chooses the full program button so the full program appears in the large text box beside buttons, Here the user for example wants to add two elements into stack, to know the top element of it, he writes 4420 in the choice textbox, 4 (represents push), the second 4 (represents another push), 2 (represents retrieving top element) and 0 (for exit), then after execution the results will appear in the output textbox which is "THE UPPER ELEMENT IS : 2" this is to represent the second element he has inserted which is located on the top of the stack.

Fig. 10 shows the solving of 4 questions, 2 of them are true and the other two are false, so his score shows 50% from 100% as shown in exercises result window. Finally in all stack widows there is a demo button (to show movie) when the student clicks on it the demo begins as shown in fig. 11, the demo starts with entering "stack" word characters into the stack one after the other, beginning with S and ending with k accompanied with stack pointer "SP" updating for each character entered , after completing all characters, the characters begin to pull out from the stack in the same way but with the inverse order and this procedure will teach the student that any word stored in the stack will exit in the inverse order and many applications require that form, but if any one wants the same word to show in the same order, he must use the queue to store it.



2.2.1.1 Stack

The stack data structure is a *lifo* (*last-in first-out*) structure .it is ob
linear list by restricting the insertions and deletions to take place from the s
end is called the top. The other end is called the bottom.

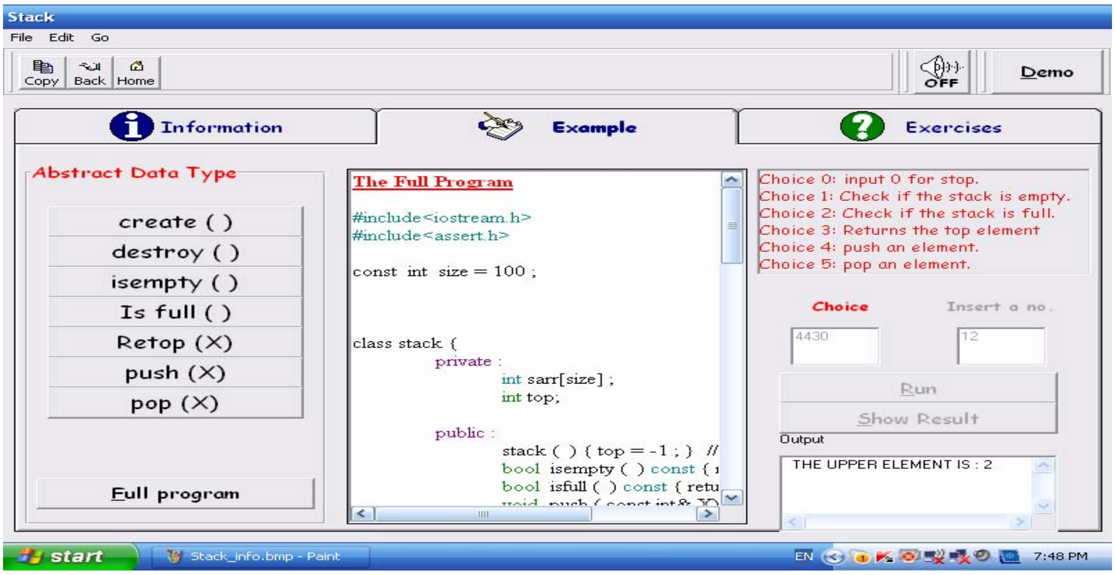
A stack in C may therefore be declared as a structure containing two ob
to hold the elements of the stack, and an integer to indicate the position
stack top within the array.

The operation on the stack (ADT): -

1-create ( ): create an empty stack.

2-destroy ( ): destroy stack.

Fig. 8
Stack information tab



The Full Program

```
#include<iostream.h>
#include<assert.h>

const  int  size = 100 ;


class stack {
          private :
                    int sarr[size] ;
                    int top;

          public :
                    stack ( ) { top = -1 ; }  //
                    bool  isempty ( ) const { ı
                    bool  isfull ( ) const { retu
```

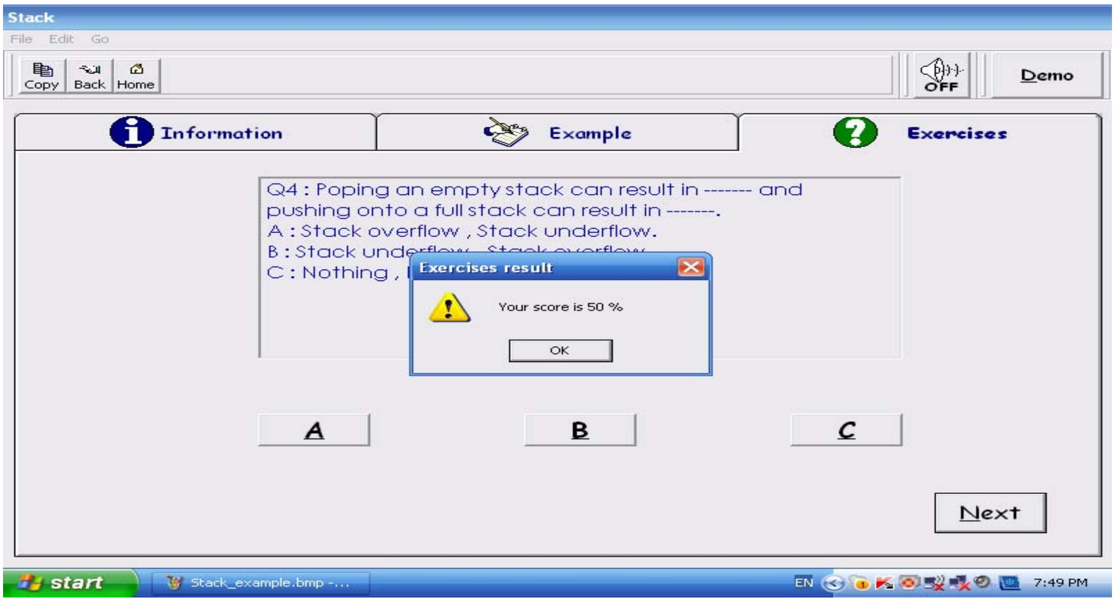
Choice 0: input 0 for stop.
Choice 1: Check if the stack is empty.
Choice 2: Check if the stack is full.
Choice 3: Returns the top element
Choice 4: push an element.
Choice 5: pop an element.

Choice: 4430

Insert a no.: 12

```
THE UPPER ELEMENT IS : 2
```

Fig. 9
Stack example tab



Q4 : Poping an empty stack can result in ------- and pushing onto a full stack can result in -------.
A : Stack overflow , Stack underflow.
B : Stack underflow , Stack overflow.
C : Nothing ,

Your score is 50 %

Fig. 10
Stack exercise tab

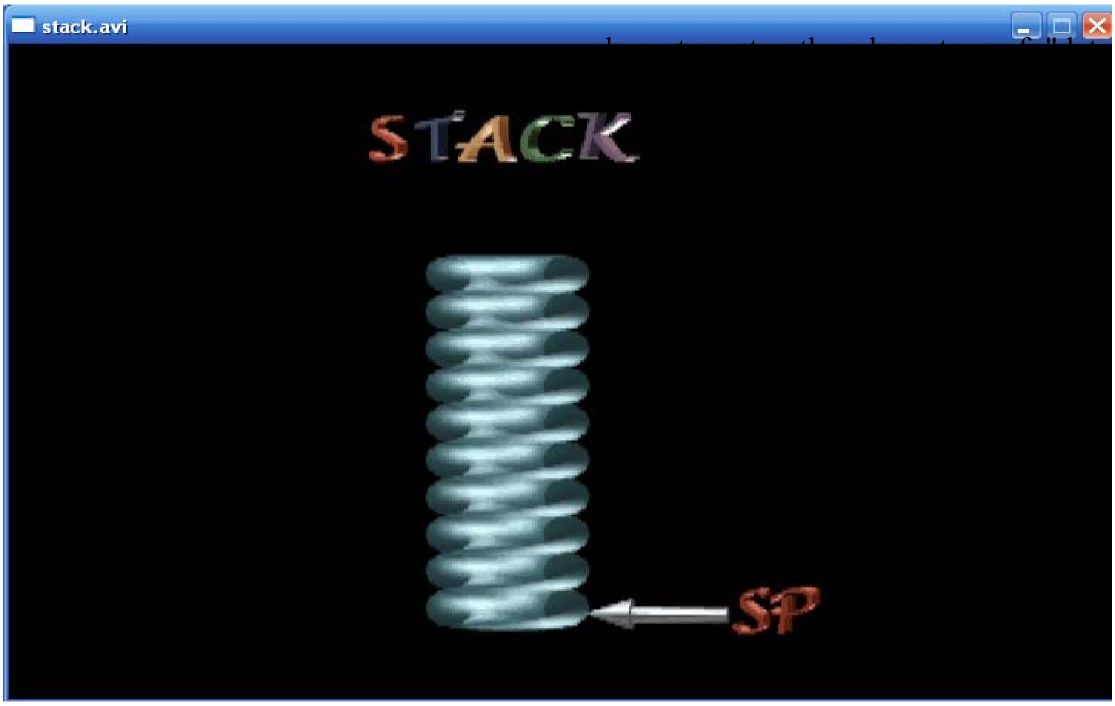


Fig. 11
Stack's Picture from movie

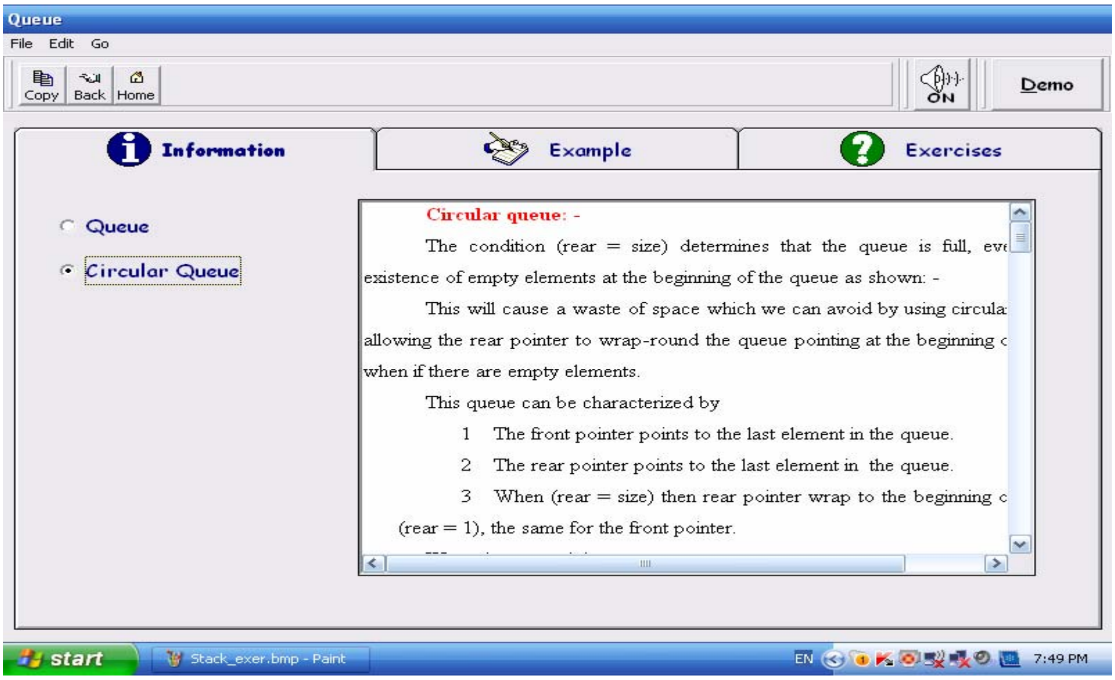
3.5Queue

The same window with alternative information is appeared in fig.12, where Queue's information are displayed, this widow will tell the student that there are two types of queues, the linear queue and the circular queue including simple description for each of them, functions implemented on the queue are displayed in the example tab as shown in fig. 13. Here the student input 22140 in the text box to implement five operations which are shown below:

2 ⟶ Insert an element to the queue which is 6.
2 ⟶ Insert another element to the queue which is 5.
1 ⟶ Check if the queue is empty.
4 ⟶ Print the content of the queue
0 ⟶ End.

After running the program the result is shown in the output textbox.

Also a movie will appear when the user clicks on demo as shown in fig. 14, the demo here shows the students word into the queue from the left and how the pointers (front and rear) will change from cell to cell after entering each character and when the whole operation is done, the characters begin to exit from the right side of the queue resulting the same word in the same order to make the student understand that any word entered the queue exit in the same order.



Circular queue: -

The condition (rear = size) determines that the queue is full, eve
existence of empty elements at the beginning of the queue as shown: -

This will cause a waste of space which we can avoid by using circula
allowing the rear pointer to wrap-round the queue pointing at the beginning c
when if there are empty elements.

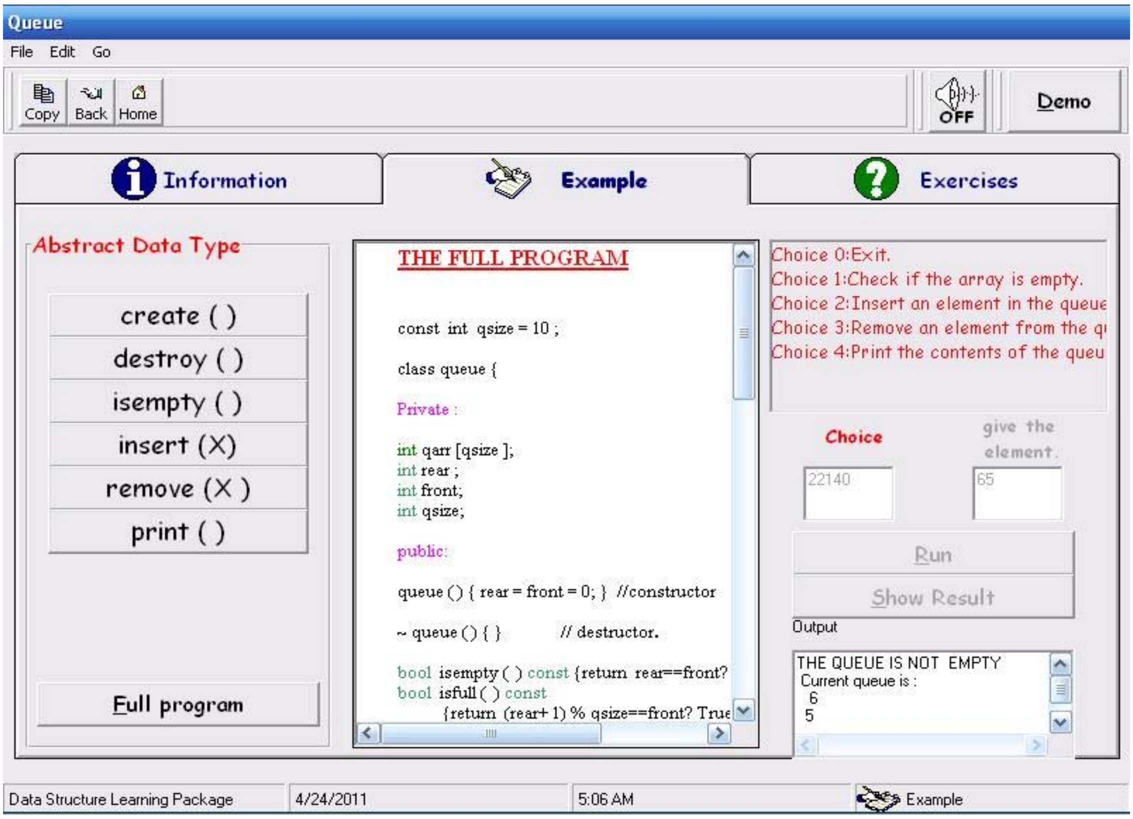
This queue can be characterized by

1 The front pointer points to the last element in the queue.

2 The rear pointer points to the last element in the queue.

3 When (rear = size) then rear pointer wrap to the beginning c
(rear = 1), the same for the front pointer.

Fig. 12
Queue information tab



THE FULL PROGRAM

```
const int qsize = 10 ;

class queue {

Private :

int qarr [qsize ];
int rear ;
int front;
int qsize;

public:

queue () { rear = front = 0; } //constructor

~ queue () { }          // destructor.

bool isempty ( ) const {return rear==front?
bool isfull ( ) const
        {return (rear+ 1) % qsize==front? True
```

Choice 0:Exit.
Choice 1:Check if the array is empty.
Choice 2:Insert an element in the queue
Choice 3:Remove an element from the qu
Choice 4:Print the contents of the queu

Choice: 22140

give the element.: 65

```
THE QUEUE IS NOT  EMPTY
 Current queue is :
  6
 5
```

Fig. 13
Queue example tab

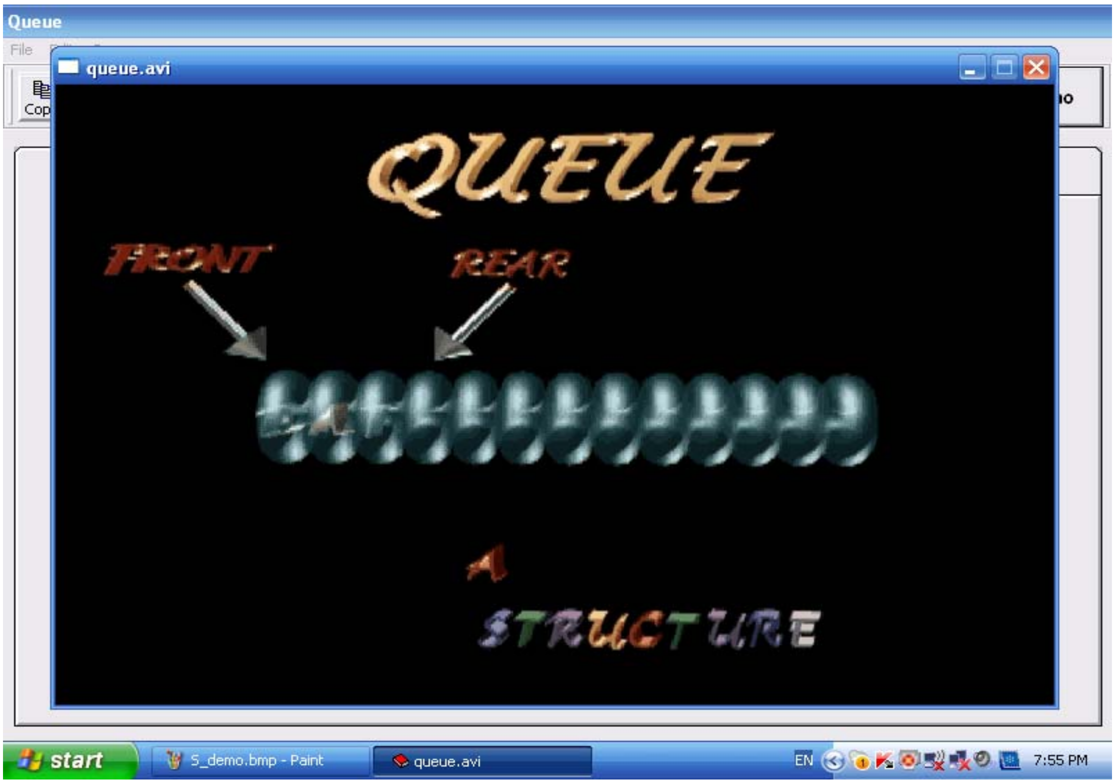
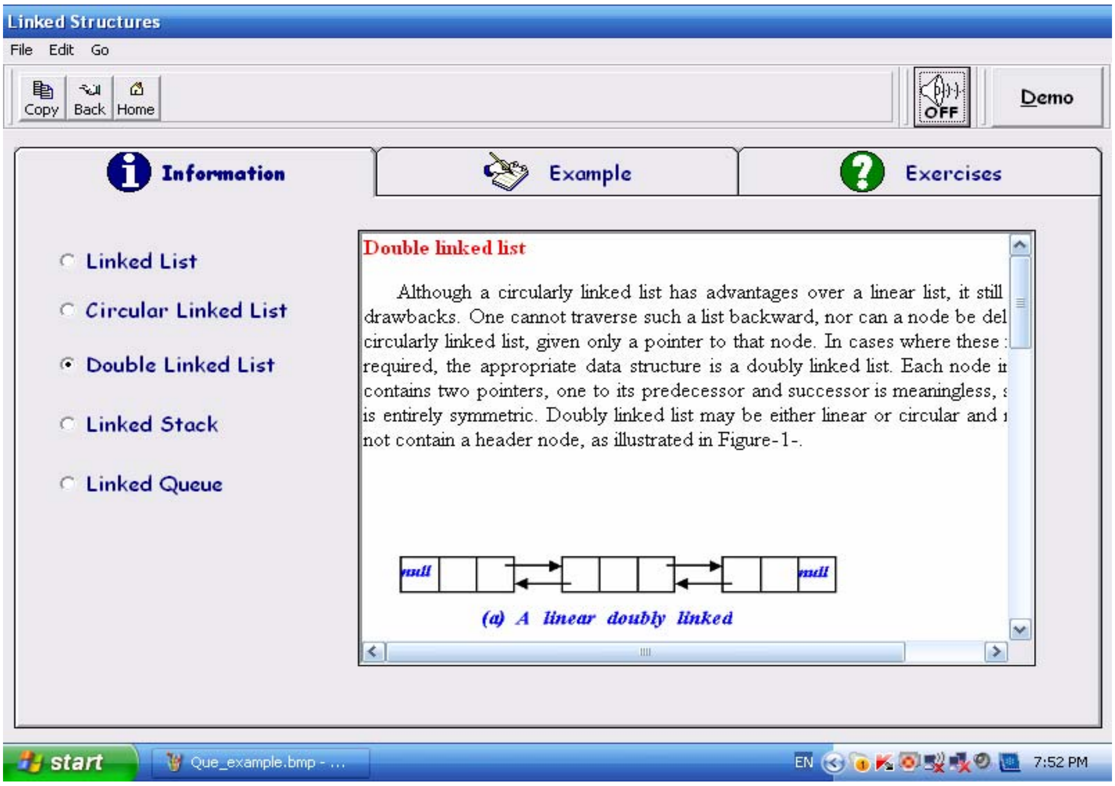


Fig. 14
Queue movie picture

3.6 Linked structure

All the details about linked data structure presented in this window include the same as the tabs shown above with alternative information as shown in fig.15 & 16. Fig. 15 shows the students that there are 5 types of linked structure. In fig. 16 there is a picture of train, it is not only a picture but it is a demo which runs after clicking on demo button, the train is chosen because it is very resembling linked structure, each vehicle represents a node and the nodes are linked with each other by pointers like the connection of train vehicles, these vehicles are numbered to show to the students that each node has a pointer, also the demo showing a feature in the linked list which is the head and tail pointers in the first and end vehicles, the demo showing also the addition and deletion operations are represented similar to exiting one vehicle form the train and leaving it in the desert (representing the deletion of a node) and then after taking a tour the same vehicle will return to the same or another location in the train (representing the addition of a node), this is to show the student that the deletion and addition exist in any index in the linked list and their pointers will be updated according to it's place, this is in contrary to the insertion and deletion of the stack and queue where updating only takes place at the beginning and their end.

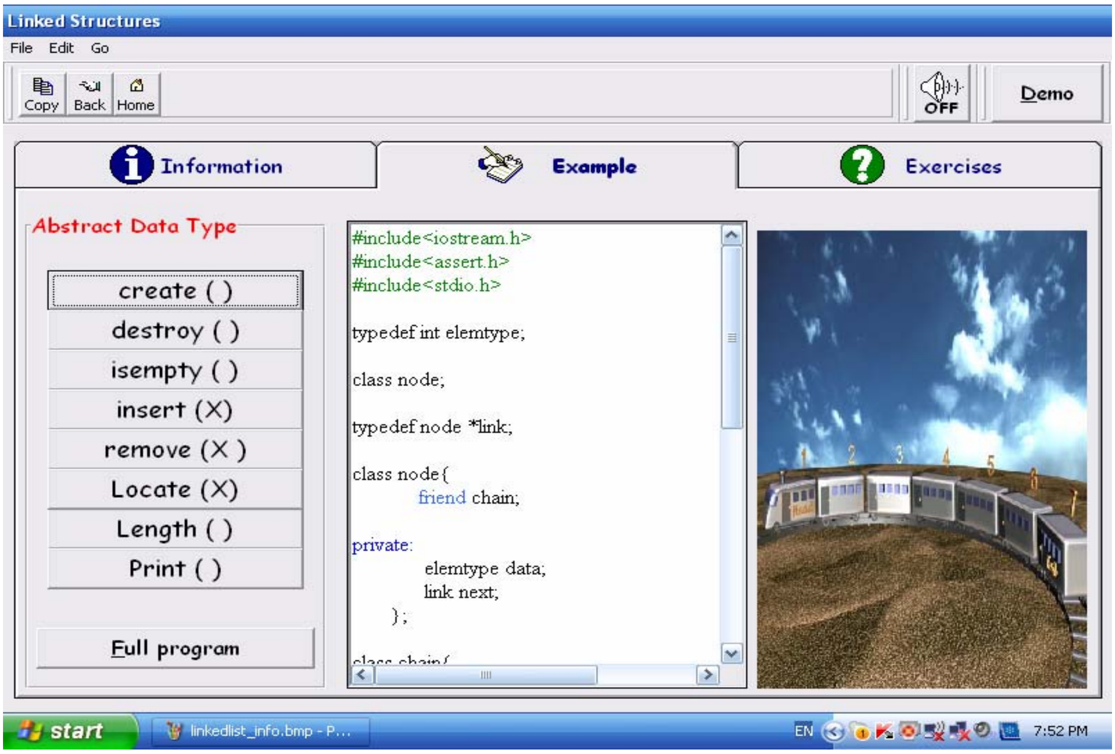
Examples and exercises of the linked list are not mentioned because they are in the same form as in array, stack and queue.



Double linked list

Although a circularly linked list has advantages over a linear list, it still
drawbacks. One cannot traverse such a list backward, nor can a node be del
circularly linked list, given only a pointer to that node. In cases where these :
required, the appropriate data structure is a doubly linked list. Each node ir
contains two pointers, one to its predecessor and successor is meaningless, s
is entirely symmetric. Doubly linked list may be either linear or circular and i
not contain a header node, as illustrated in Figure-1-.

Fig. 15
Linked structure information tab



```
#include<iostream.h>
#include<assert.h>
#include<stdio.h>

typedef int elemtype;

class node;

typedef node *link;

class node{
          friend chain;

private:
          elemtype data;
          link next;
     };

class chain(
```

Fig. 16
Linked structure example tab with picture from movie

At this point, the research covered most of the windows. The chosen information in information tap in all the windows is not a copy from one book but it is abstracts from multiple of books further to lectures. These abstracts and lectures were collected , assorted and written by me in word documents and were retrieved by the learning package program. If any student want these information, he/she can take it by copying them by the copy button or by taking the word documents which are available with the package. The examples tap first display each operation alone to help the student to focus in each operation then see all operations in one executable program, also the exercises are chosen to cover all package subjects to ensure that the student completely understands. The reason for focusing on demos and pictures, for the fact that pictures are stored in the brain much more than text.

4. USER ACCEPTANCE TESTING

User Acceptance testing is one of the important test activities in software development. It requires the participation of the end-users. An acceptance test plan was developed to guide the testing process. In order to conduct this testing, 31 users were selected at random to participate in the test. These participants were first given the LP to learn data structure using the system. They were each given a number of questionnaires to answer to indicate their level of satisfaction on LP, fig. 17 and fig. 18 show the user acceptance test results.

Table 1 and figure 17 show that 5 (0.16) and 21 (0.67) students, totally agree and agree that data structure LP is very sufficient for learning in the course. However, 4 (0.12) students are not sure. This could possibly be due to the fact that there were not given enough time to use LP. Only 1 student (0.03) disagrees with the other students' opinion.

Table 1
Student's test1

	Totally agree	agree	Not sure	disagree	Totally not agree
students	5	21	4	1	0

Table 2 and Figure 18 show that 7 (0.22) and 20 (0.64) students, totally agree and agree that they are satisfied with the functions and features provided in our LP, respectively. Only 4 (0.12) students indicated that they are not sure if LP can help to learn data structure. Overall, 27 (0.87) of the students agree that DSLP can assist them in the learning of data structure in an easy and interesting way.

Table 2
Student's test2

	Totally agree	agree	Not sure	disagree	Totally not agree
students	7	20	4	0	0

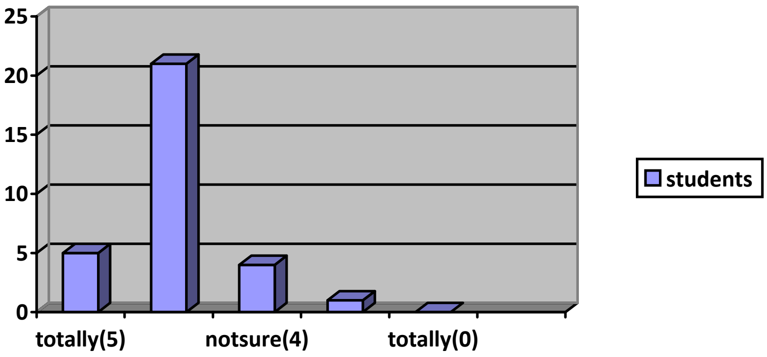


Fig. 17
Student's test1

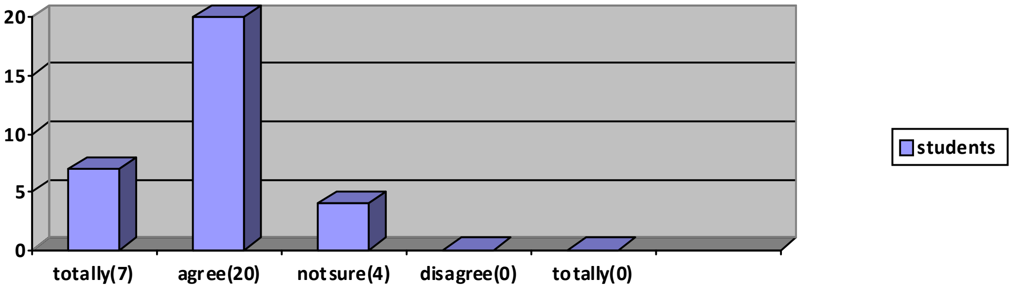


Fig. 18
Student's test2

5. CONCLUSIONS AND FUTURE WORK

This research involves a data structure learning package concerning arrays and linear structure. The package teaches the student through illustrations of how to represent data structure in memory. The package uses many windows showing three taps. The first is the information tap covering the demonstration structure, the second tap includes all operations performed on it which can be displayed in one executable program receiving inputs from students who can enter different inputs according to the choices displayed and output is displayed after C++ program is implemented. The third tap includes exercises for testing the ability of the students in understanding. The package will give a score for the student after completion of all questions depending on the solutions. Teaching aids are used in the package such as pictures, movies and voices that represent the data structure implemented with 3D max and to teach the students for using the package.

As a future plan:

1. Expanding this project to include non linear structures represented as graphs.
2. Using front page to design it can provide much better facility and flexibility.
3. By using large database, exercises can be selected randomly which gives the package higher quality.
4. More movies that show the mechanism of the ADT in a simple manner.
5. New sections can be used for writing C++ programs by students. This program then will be connected to the C++ compiler for checking it.
6. Some other languages (Pascal, Turbo C, Basic...) can be used to implement data structure.

REFRENCES

[1] Allen, "Michael Allen's Guide into E-Learning", admin, 2010.

[2] Bruno R., "Data Structure and algorithms with object oriented design patterns in C++", John Wiley & Sons, 1998.

[3] Clifford A., "Practical information to data structure and algorithm analysis third edition (C++ version)", Freeteckbooks.com, 2010.

[4] Daviniahl,"Investigations of E-Learning Patterns",2011.

[5] Granville B. and Luca T., "Data structures and algorithms: Annotated Reference with examples", Dotnetslackers, 2008.

[6] Jane S.,"Copyright and e-learning : a guide for practitioners", London Facet, 2010.

[7] Nigel P. and Eliom M., " E-Learning Perspectives", The MASIL Center, 2010.

[8] Som N., "E-learning A Guidebook of Principles, Procedures and Practices", CEMCA, 2006.

[9] Tavangarian D., Leybold M., Nolting K. and Roser M., "Is e_learningthe solution for individual learning?", Journal of e_learning, 2004.

[10] Yedidyah L., Moshe J. and Aaron M.," Data Structures using C and C++",1998.

[11] Yahya H. and Mohamed B., "Visual Basic 6", Dar Wael, Jordan, 2001.