|  |
| --- |
| MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE  NATIONAL TECHNICAL UNIVERSITY  «KHARKIV POLYTECHNIC INSTITUTE»  Department of Software Engineering and Management Information Technologies  List of laboratory reports  discipline «Algorithm and Data Structures»  Executed by: Chukwu Irele Omike  Student of group KH-201.8ei.1  Checked by:  Prof.M . O.Bilova  Kharkiv – 2019 |

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

«KHARKIV POLYTECHNIC INSTITUTE»

Department of Software Engineering and Management Information Technologies

Report from lab № 1

discipline «Algorithm and Data Structures»

Kharkiv

2018

Theme :Basic Data Structures:

Objective: explore basic data structures (list, queue, and stack) and get

programming skills of algorithms that process them.

Task:

Develop a program that reads from the keyboard sequence of N unique

integers N (1 <N <256), saves them to a data structure (according to task) and

displays the following characteristics:

number of elements;

average of saved items;

minimum and maximum element;

fourth element of the sequence;

element that is before the minimum element.

Progress of the lab:

Theory : Double linked list is a data structure that deals with Nodes .It has two pointers: previous (which houses the address of the previous node ) and Next (that houses the address of the next Node). The availability of this pointers make it easy to transverse the list. Moreover , a double linked list can be implemented as a XOR-linked list , thus use the same amount of memory as a single linked list. Its easy to perform basic data structure operations (searching ,adding node, deleting a node and traversing the list ) with the use of the pointers .

CODE:

using System;

using System.Text;

namespace lab1

{

/\*Node class represents the node of doubly linked list. It consists of the

information part and links to its succeeding and preceeding nodes

in terms of next and previous nodes.\*/

class Node

{

public int age;

public string name;

public Node next; //points to the next node

public Node prev; //points to the previous node;

}

class doubleLinkedList

{

Node START;

public doubleLinkedList()

{

START = null;

}

public void addNode()//adds a new node

{

int a;

string nm;

Console.Write("\nEnter the name of student ");

nm = Console.ReadLine();

Console.Write("\nEnter the age of student ");

a = Convert.ToInt32(Console.ReadLine());

Node NewNode = new Node();

NewNode.age = a;

NewNode.name = nm;

if(START == null)

{

NewNode.next = START;

NewNode.prev = null;

START = NewNode;

return;

}

Node current, previous;

for( current = previous = START;current != null;previous = current, current = current.next)

{

}

/\*On the execution of the above for loop, prev and current will point to those nodes

between which the new node is to be inserted.\*/

NewNode.next = current;

NewNode.prev = previous;

if (current == null)/\*If the node is to be inserted at the end of the list.\*/

{

NewNode.next = null;

previous.next = NewNode;

return;

}

//current.prev = newnode;

//previous.next = newnode;

Console.WriteLine("record added");

}

public bool delNode(string name )

{

Node previous, current;

previous = current = null;

if (Search(name, ref previous, ref current) == false)

return false;

if (current == START)

{

START = START.next;

if (START != null)

START.prev = null;

return true;

}

if (current.next == null)/\*If the last node is to be deleted\*/

{

previous.next = null;

return true;

}

/\*If the node to be deleted is in between the list \*/

previous.next = current.next;

current.next.prev = previous;

return true;

}

public void traverse()/\*Traverses the list\*/

{

if (listEmpty())

Console.WriteLine("\nList is empty");

else

{

Console.WriteLine("\nRecords in the ascending order of placement are:\n");

Node currentNode;

for (currentNode = START; currentNode != null; currentNode = currentNode.next)

Console.Write(currentNode.name + " " + currentNode.age + "\n");

}

}

public void revtraverse()/\*traverses the list in the reverse direction\*/

{

if (listEmpty())

Console.WriteLine("\nList is empty");

else

{

Console.WriteLine("\nRecords in the descending order of placement are:\n");

Node currentNode;

for (currentNode = START; currentNode.next != null; currentNode = currentNode.next)

{ }

while (currentNode != null)

{

Console.Write(currentNode.name + " " + currentNode.age + "\n");

currentNode = currentNode.prev;

}

}

}

public bool Search(string name , ref Node previous ,ref Node Current)/\*Checks wheteher the specified node is present\*/

{

for(previous = Current = START;Current != null && name != Current.name; previous = Current, Current = Current.next)

{

}

return(Current != null);

}

public bool listEmpty()

{

if (START == null)

return true;

else

return false;

}

static void Main(string[] args)

{

doubleLinkedList obj = new doubleLinkedList();

while (true)

{

try

{

Console.WriteLine("\nMenu");

Console.WriteLine("1. Add a record to the list");

Console.WriteLine("2. Delete a record from the list");

Console.WriteLine("3. View all records in the ascending order of placement");

Console.WriteLine("4. View all records in the descending order of placement");

Console.WriteLine("5. Search for a record in the list");

Console.WriteLine("6. get number of records");

Console.WriteLine("7. get average ages");

Console.WriteLine("8. find minimum and maximum age ");

Console.WriteLine("9. get the fourth record in the sequence");

Console.WriteLine("10. find element that is before the minimum");

Console.WriteLine("11. Exit\n");

Console.Write("Enter your choice (1-6): ");

string ch = Console.ReadLine();

switch (ch)

{

case "1":

{

obj.addNode();

}

break;

case "2":

{

if (obj.listEmpty())

{

Console.WriteLine("\nList is empty");

break;

}

Console.Write("\nEnter the name of the student whose record is to be deleted: ");

string name = Console.ReadLine();

Console.WriteLine();

if (obj.delNode(name) == false)

Console.WriteLine("Record not found");

else

Console.WriteLine("Record with name " + name + " deleted \n");

}

break;

case "3":

{

obj.traverse();

}

break;

case "4":

{

obj.revtraverse();

}

break;

case "5":

{

if (obj.listEmpty() == true)

{

Console.WriteLine("\nList is empty");

break;

}

Node prev, curr;

prev = curr = null;

Console.Write("\nEnter the name of the student whose record you want to search: ");

string name = Console.ReadLine();

if (obj.Search(name, ref prev, ref curr) == false)

Console.WriteLine("\nRecord not found");

else

{

Console.WriteLine("\nRecord found");

Console.WriteLine("\nName: " + curr.name);

Console.WriteLine("\nAge: " + curr.age);

}

}

break;

case "6":

{

int count = 0;

if (obj.listEmpty())

Console.WriteLine("\nList is empty");

else

{

Node currentNode;

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

count++;

Console.WriteLine("\nNumber of records saved is : " + count + "\n");

}

}

break;

case "7":

{

double Average = 0;

int sum = 0;

int count = 0;

if (obj.listEmpty())

Console.WriteLine("\nList is empty");

else

{

Node currentNode;

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

{

count++;

sum += currentNode.age;

}

Average = (double)sum / count;

Console.WriteLine("\nThe Average age is : " + Average + "\n");

}

}

break;

case "8":

{

Node min;

Node max;

if (obj.listEmpty())

Console.WriteLine("\nList is empty");

else

{

Node currentNode;

min = obj.START;

max = obj.START;

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

{

if (currentNode.age < min.age )

{

min = currentNode;

}

}

Console.WriteLine("\nStudent with the minimum age is : " + min.name + " with an age of :" + min.age + "\n");

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

{

if(currentNode.age > max.age)

{

max = currentNode;

}

}

Console.WriteLine("\nStudent with the Maximum age is : " + max.name + " with an age : " + max.age + "\n");

}

}

break;

case "9":

{

int count = 0;

if (obj.listEmpty())

Console.WriteLine("\nList is empty");

else

{

Node currentNode;

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

{

count++;

if( count == 4 )

{

Console.WriteLine("\nThe frouth record in the sequence is : " + currentNode.name +" " + currentNode.age + "\n");

break;

}

}

if(count < 4)

{

Console.WriteLine("\nThe sequence has less than four records \n");

}

}

}

break;

case "10":

{

Node min;

Node preElemnt;

if (obj.listEmpty())

Console.WriteLine("\nList is empty");

else

{

Node currentNode;

min = obj.START;

for (currentNode = obj.START; currentNode != null; currentNode = currentNode.next)

{

if (currentNode.age < min.age)

{

min = currentNode;

}

}

preElemnt = min.prev;

if(min == null)

{

Console.WriteLine("there is no element prior to the minimum");

}

Console.WriteLine("\nStudent previous to the student with minimum age is : " + preElemnt.name + " with an age of :" + preElemnt.age + "\n");

}

}

break;

case "11":

return;

default:

{

Console.WriteLine("\nInvalid option");

}

break;

}

}

catch (Exception e)

{

Console.WriteLine("Check for the values entered.");

}

}

}

}

}

:

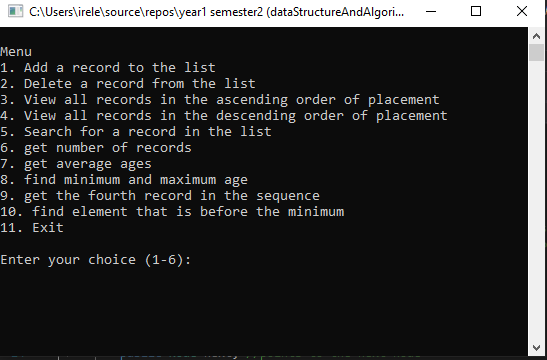


Figure.1- Menu of program

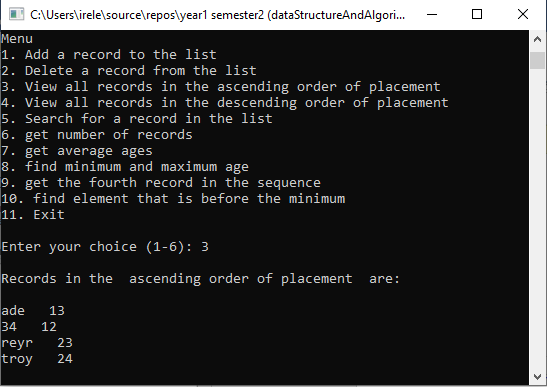
\

Figure.2- All saved records



Figure.3- Average age saved

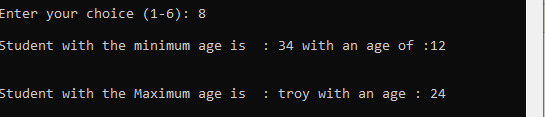


Figure.4- Minimum and maximum element

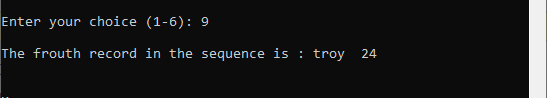


Figure.5- Fourth element of the squence

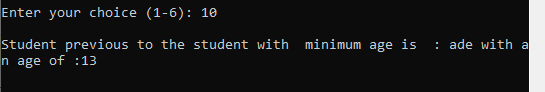


Figure.6- element that is before the minimum element;

Conclusion:

In this laboratory the study of data structures was considered and implementing various basic data structure operations. Double linked list was implemented in this lab. The following operations was carried out

* Transverse the list
* Finding minimum and maximum element saved
* Searching for a specific element in the list
* Finding the average of elements stored in the list.