Kahlil Bello

Data Structures

Oct 17 2018

Project 6

#include <iostream>

#include "STACKPAC.h"

#include <iomanip>

using namespace std;

int main()

{

//given 3 arrays

int a[7] = {3,6,9,1,8,5,2};

int b[5] = {10, 50, 70, 20, 40};

int c[6] = { 100, 500, 800, 300, 200, 900 };

//----------------------QUESTION 1------------------------------

//1. insert array a in a linked list to behave like a stack FILO

//-->[2| ]-->[5| ]--> --> -->[3|/]

STACK\_LIST <int> theStack;

cout << "Displaying array a: ";

for (int i = 0; i < 7; ++i)

{

cout << a[i] << " ";

//-----pushing array values into stack to behave as LIFO

theStack.pushs(a[i]);

}//endfor

cout << endl;

int sum = 0;

int totalNodes = 0;

//---------displaying array a in reversed order

//i.) DISPLAY THE CONTENTS OF ALL THE NODES

cout << "Displaying array a REVERSED: ";

while (!theStack.emptys())

{

int d = theStack.pops();

//ii.) COMPUTE THE TOTAL OF ALL THE NODES

sum += d;

//iii.) DETERMINE THOW MANY NODES ARE IN THE LINKED LIST

totalNodes++;

cout << d << "-->";

}//endwhile

cout << "NULL\n";

cout << "Sum: " << sum << endl;

cout << "Total Nodes: " << totalNodes << endl;

cout << "############################################################\n\n";

//----------------------QUESTION 2------------------------------

//2. insert array b in a linked list to behave like a stack FIFO

//-->[10| ]-->[50| ]--> --> -->[40|/]

QUEUE\_LIST <int> theQueue;

cout << "Displaying array b: ";

for (int i = 0; i < 5; ++i)

{

cout << b[i] << " ";

//-----pushing array values into queue to behave as FIFO

theQueue.pushq(b[i]);

}//endfor

cout << endl;

//i.) DISPLAY THE CONTENTS OF ALL THE NODES

int max=0;

int min=b[4]; //since b[0] is already the lowest so put diff num to if in whileloop can sort

cout << "Displaying array b in SAME ORDER: ";

while (!theQueue.emptyq())

{

int d = theQueue.popq();

//ii. FIND THE MAX DATA IN THE QUEUE

if (d > max)

{

max = d;

}

// FIND THE MIN DATA IN THE QUEUE

if (d < min)

{

min = d;

}

cout << d << "-->";

}//endwhile

cout << "NULL\n";

cout << "The Maximum is: " << max << endl;

cout << "The Minimum is: " << min << endl;

cout << "############################################################\n\n";

//----------------------QUESTION 3------------------------------

//3. Write a program to insert data in array c in an ordered linked list

ORDEREDLINKEDLIST <int> theOList;

QUEUE\_LIST <int> theEVEN; //for even numbers added to list

int isEven;

cout << "Displaying array c: ";

for (int i = 0; i < 6; ++i)

{

cout << c[i] << " ";

//-----pushing array values into orderedlist

theOList.insert(c[i]);

//ii.) compute average of all even numbers, collect even numbers into queue or stack

if ((c[i] % 2) == 0)

{

theEVEN.pushq(c[i]);

}

}//endfor

cout << endl;

//i.) DISPLAY THE CONTENTS OF ALL THE NODES

cout << "Displaying array c in ASCENDING: ";

theOList.display();

//ii.) compute and display the average of all even numbers in array c

int totalEven=0;

double theAverage;

while (!theEVEN.emptyq())

{

int d = theEVEN.popq();

totalEven += d;

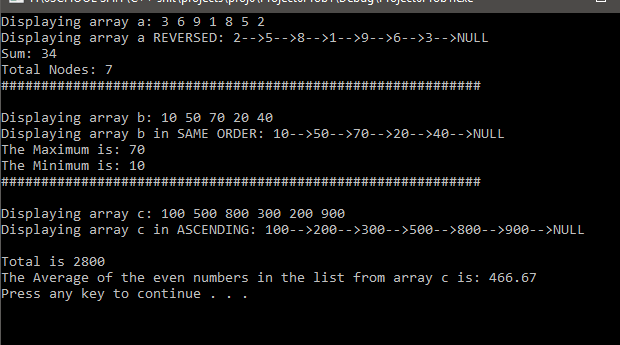
}

cout << endl;

cout << "Total is " << totalEven << endl;

theAverage = (totalEven / 6.0);

cout << "The Average of the even numbers in the list from array c is: " << setprecision(5) << theAverage << endl;

 system("pause");

return 0;

}//end main

OUTPUT:

/\*

OUTPUT:

Displaying array a: 3 6 9 1 8 5 2

Displaying array a REVERSED: 2-->5-->8-->1-->9-->6-->3-->NULL

Sum: 34

Total Nodes: 7

############################################################

Displaying array b: 10 50 70 20 40

Displaying array b in SAME ORDER: 10-->50-->70-->20-->40-->NULL

The Maximum is: 70

The Minimum is: 10

############################################################

Displaying array c: 100 500 800 300 200 900

Displaying array c in ASCENDING: 100-->200-->300-->500-->800-->900-->NULL

Total is 2800

The Average of the even numbers in the list from array c is: 466.67

Press any key to continue . . .

\*/

Header file

#pragma once

#ifndef STACKPAC

#define STACKPAC

using namespace std;

template <class T>

class QUEUE\_LIST

{

private:

struct node

{

T info;

node \*next;

};//end struct node

node \*front, \*rear;

public:

QUEUE\_LIST() { front = NULL; rear = NULL; }//CONSTRUCTOR

//-------ADD A NODE AT THE REAR OF THE QUEUE

void pushq(T x)

{

node \*p;

p = new(node);

p->info = x;

p->next = NULL;

if (front == NULL)

{

front = p;

rear = p;

}//endif

else

{

rear->next = p;

rear = p;

}//endelse

}//end pushq

//-------TEST WHETHER QUEUE IS EMPTY OR NOT

bool emptyq()

{

return(front == NULL) ? true : false;

}//end emptyq

//---------POP FIRST NODE

T popq()

{

node \*p = front;

T x;

x = front->info;

front = front->next;

delete(p);

return x;

}//end popq

};//end QUEUE\_LIST

//-----CLASS STACK

template<class G>

class STACK\_LIST

{

private:

struct node

{

G info;

node \*next;

};//end struct node

node \*stack;

public:

STACK\_LIST() { stack = NULL; }//constructor

//----------ADD NODE AT THE FRONT OF THE LIST

void pushs(G x)

{

node \*p;

p = new(node);

p->info = x;

p->next = NULL;

if (stack == NULL)

{

stack = p;

}//endif

else

{

p->next = stack;

stack = p;

}//endelse

}//end pushs

//-------------test WHETHER STACK IS EMPTY OR NOT

bool emptys()

{

return(stack == NULL) ? true : false;

}//end emptys

//---------------pop the first node

G pops()

{

node \*p = stack;

G x;

x = stack->info;

stack = stack->next;

delete(p);

return x;

}//end pops

};//end STACK\_LIST

//------CLASS ORDEREDLINKEDLIST

template<class F>

class ORDEREDLINKEDLIST

{

private:

struct NODE

{

F info;

NODE \*next;

};//end struct NODE

NODE \*list;

public:

ORDEREDLINKEDLIST() { list = NULL; }

void insert(F x)

{

NODE \*p = list, \*q = list, \*r;

//create new node

r = new(NODE);

r->info = x;

r->next = NULL;

//find the insertion place

while (p != NULL && x > p->info)

{

q = p;

p = p->next;

}//endwhile

if (p == list) //x is the first info

{

list = r;

r->next = p;

}//endif

else if (p == NULL) //x is the last info

{

q->next = r;

}//end elseif

else //x is neither first nor last info

{

r->next = p;

q->next = r;

}//endelse

}//end insert

void display()

{

NODE \*p = list;

while (p != NULL)

{

cout << p->info << "-->";

p = p->next;

}//endwhile

cout << "NULL\n";

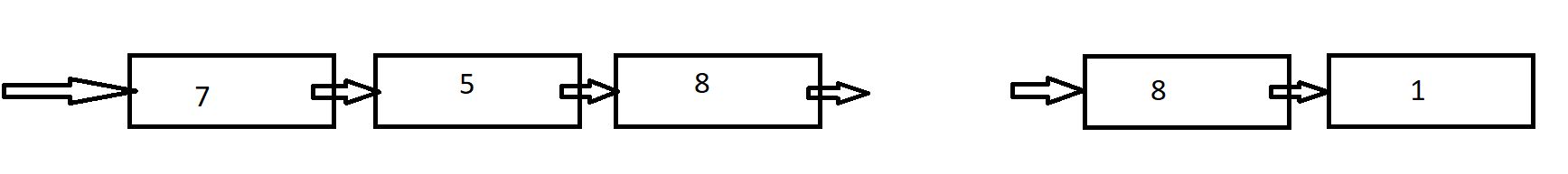
}//end display

};//end class ORDEREDLINKEDLIST

#endif

WRITTEN PART

LIST



4-i.) INSERT 10 in front of the list

node \*p;

p = new (node);

p->info = 10;

p->next = list;

list=p;

//-------------------------------------------------------------------

4-ii.) insert 50 at the rear of the list

node \*p;

p->info = 50;

p->next=NULL;

node \*q=list;

while(q->next!=NULL)

{

q=q->next;

}//endwhile

q->next=p;

//-------------------------------------------------------------------

4-iii.) insert 30 before the node whose info is 40

node \*p = new (node);

p->info = 30; p->next=NULL;

node \*r;

node \*q;

r=list; q=list;

wihle(q->info!=40)

{

r=q;

q=q->next;

}//endwhile

r->next=p;

p->next=q;

//-------------------------------------------------------------------

4-iv.) insert 66 after the first node

node \*p;

p=new(node);

p->info=66;

p->next=NULL;

node \*r, \*q;

r = list; q = list;

r=r->next;

q->next=p;

p->next=r;

/\* OR

node \*p;

p=new(node);

p->info=66;

p->next=list->next->next;

list->next=p;

\*/