Divyajitsinh Mahida

Data Structures and Algorithms

Project 1

Main.cpp

#include <iostream>

using namespace std;

#include "doublylinkedlist.h"

int main()

{

// test case 1

// Create a doubly linked list

DoublyLinkedList Dll\_1;

// Each node contains only one integer of a string

Node n11(10);

Node n12(20);

Node n13(30);

Node n14(40);

Node n15(50);

// a doubly linked list of integers by inserting nodes

Dll\_1.addNewNodeToFront(&n11);

Dll\_1.addNewNodeToBack(&n12);

Dll\_1.addNewNodeToFront(&n13);

Dll\_1.addNewNodeToBack(&n14);

Dll\_1.addNewNodeToFront(&n15);

Dll\_1.drawDoublyLinkedList();

cout << endl;

system("PAUSE");

return 0;

}

#include <iostream>

using namespace std;

#include "doublylinkedlist.h"

int main()

{

// test case 1

// Create a doubly linked list

DoublyLinkedList Dll\_1;

// Each node contains only one integer of a string

Node n11(10);

Node n12(20);

Node n13(30);

Node n14(40);

Node n15(50);

// a doubly linked list of integers by inserting nodes

Dll\_1.addNewNodeToFront(&n11);

Dll\_1.addNewNodeToFront(&n12);

Dll\_1.addNewNodeToFront(&n13);

Dll\_1.addNewNodeToFront(&n14);

Dll\_1.addNewNodeToFront(&n15);

cout << Dll\_1.isPalindrome() << endl;

// test case 2

// Create a doubly linked list

DoublyLinkedList Dll\_2;

// Each node contains only one integer of a string

Node n21(10);

Node n22(20);

Node n23(20);

Node n24(10);

// a doubly linked list of integers by inserting nodes

Dll\_2.addNewNodeToFront(&n21);

Dll\_2.addNewNodeToFront(&n22);

Dll\_2.addNewNodeToFront(&n23);

Dll\_2.addNewNodeToFront(&n24);

cout << Dll\_2.isPalindrome() << endl;

system("PAUSE");

return 0;

}

// The main.cpp

#include <iostream>

using namespace std;

#include "Node.h"

#include "doublylinkedlist.h"

int main()

{

// test case 1

// Create a doubly linked list

DoublyLinkedList\* Dll\_1;

Dll\_1 = new DoublyLinkedList;

// Each node contains only one integer of a string

Node n11(1);

Node n12(2);

Node n13(3);

Node n14(4);

Node n15(5);

Node n16(6);

Node n17(7);

// a doubly linked list of integers by inserting nodes

Dll\_1->addNewNodeToBack(&n11);

Dll\_1->addNewNodeToBack(&n12);

Dll\_1->addNewNodeToBack(&n13);

Dll\_1->addNewNodeToBack(&n14);

Dll\_1->addNewNodeToBack(&n15);

Dll\_1->addNewNodeToBack(&n16);

cout << "The size of the doubly linked list is " << Dll\_1->countSize() << endl;

system("PAUSE");

return 0;

}

// The main.cpp

#include <iostream>

using namespace std;

#include "Node.h"

#include "doublylinkedlist.h"

int main()

{

// test case 1

// Create a doubly linked list

DoublyLinkedList\* Dll\_1;

Dll\_1 = new DoublyLinkedList;

// Each node contains only one integer of a string

Node n11(1);

Node n12(2);

Node n13(3);

Node n14(4);

Node n15(5);

Node n16(6);

// a doubly linked list of integers by inserting nodes

Dll\_1->addNewNodeToBack(&n11);

Dll\_1->addNewNodeToBack(&n12);

Dll\_1->addNewNodeToBack(&n13);

Dll\_1->addNewNodeToBack(&n14);

Dll\_1->addNewNodeToBack(&n15);

Dll\_1->addNewNodeToBack(&n16);

Dll\_1->split(3);

system("PAUSE");

return 0;

}

// The main.cpp

#include <iostream>

using namespace std;

#include "Node.h"

#include "doublylinkedlist.h"

int main()

{

// test case 2

// Create a doubly linked list

DoublyLinkedList\* Dll\_1;

Dll\_1 = new DoublyLinkedList;

// Each node contains only one character of a string

Node n11(1);

Node n12(2);

Node n13(3);

Node n14(4);

Node n15(5);

Node n16(6);

Node n17(7);

// a doubly linked list of characters by inserting nodes

Dll\_1->addNewNodeToBack(&n11);

Dll\_1->addNewNodeToBack(&n12);

Dll\_1->addNewNodeToBack(&n13);

Dll\_1->addNewNodeToBack(&n14);

Dll\_1->addNewNodeToBack(&n15);

Dll\_1->addNewNodeToBack(&n16);

Dll\_1->addNewNodeToBack(&n17);

Dll\_1->split(3);

system("PAUSE");

return 0;

}

Doublylinkedlist.cpp

// University of Arkansas at Little Rock

// Department of Computer Science

// CPSC 2380: Data Structures and Algorithms

// Spring 2020

// Project 1: Doubly Linked List

// Due Date: March 10, 2020

// Name: Divyajitsinh Mahida

// T-number (Last 4 Digits): T00631929

// Description of the Program (2-3 sentences): In this progarm we are compalying the code using the doublylinkedlist.

// Date Written: 03/10/2020

// Date Revised: 03/10/2020

#ifndef DOUBLYLINKEDLIST\_CPP

#define DOUBLYLINKEDLIST\_CPP

#include "doublylinkedlist.h"

DoublyLinkedList::doublyLinkedList()

{

head = NULL;

tail = NULL;

}

doublyLinkedList::~DoublyLinkedList()

{

}

void doublyLinkedList::addNewNodeToFront(Node\* NewNode) {

if (head == NULL && tail == NULL) {

head = tail = newNode;

}

else {

head->prev = newNode;

newNode->next = head;

head = newNode;

}

}

void DoublyLinkedList::addNewNodeToBack(Node\* newNode) {

if (head == NULL && tail == NULL) {

head = tail = newNode;

}

else {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

Node\* DoublyLinkedList::removeNodeFromFront()

{

Node\* tempNode;

tempNode = head;

head = head->next;

return tempNode;

}

void DoublyLinkedList::displayDoublyLinkedList()

{

Node\* tempNode;

tempNode = head;

while (tempNode != NULL)

{

cout << tempNode->value << " ";

tempNode = tempNode->next;

}

}

void DoublyLinkedList::drawDoublyLinkedList()

{

Node\* tempNode;

tempNode = head;

while (tempNode != NULL)

{

printf("[address:%X, ", tempNode);

printf("prev:%X, ", tempNode->prev);

printf("value:%d , ", tempNode->value);

printf("next:%X]", tempNode->next);

cout << endl;

tempNode = tempNode->next;

}

}

bool DoublyLinkedList::isPalindrome()

{

Node\* tempNode1 = head;

Node\* tempNode2 = tail;

bool check;

if (tempNode1->value == tempNode2->value)

{

check = true;

}

else

{

check = false;

return check;

}

while (tempnode1 = NULL) {

if ((tempNode1->value)) == (tempNode2->value)) {

check = true

}

}

else {

check = flase

return check;

}

tempNode1 = tempNode1->Nect;

tempNode2 = tempNode2->prev;

}

return check;

int DoublyLinkedList::countSize()

{

int count = 0;

Node\* tempNode;

tempNode = head;

while (head != NULL)

{

count = count + 1;

head = head->next;

}

return count;

}

Void DoublyLinkedList::split(int n) // n is the number of even partitions

{

Node\* tempNode = head;

int size = this->countsize();

DoublyLinkedList\* arr;

arr = new DoublyLinkedList[n];

if (!(size % n == 0) || (n == 0) || (tempnode == NULL)) {

cout << "cannot be processed.\n";

}

else {

int numofElement = size / n

int numInserted = 0;

for (int i = 0; i < n; i++) {

for (int k = 0; k < numofElement; k++) {

arr{ i }.addNewNodeToBack(new Node(tempNode->value));

tempNode = tempNode->next;

}

}

cout << n << "evenly split subdoubly linked lists.\n";

for (int k = 0; k < n k++) {

arr{ k }.displayDoublyLinkedList();

cout << endl;

}

}

delete[] arr;

}

void DoublyLinkedList::addNewNodeBeforeTargetNode(Node\* newNode, int c)

{

ary[i].displayDoublyLinkedList();

cout << " " << endl;

}

#endif

Doublylinkedlist.h

#pragma once

#ifndef DOUBLYLINKEDLIST\_H

#define DOUBLYLINKEDLIST\_H

#include <iostream>

#include "Node.h"

class DoublyLinkedList {

public:

DoublyLinkedList();

~DoublyLinkedList();

void addNewNodeToFront(Node\* newNode);

void addNewNodeToBack(Node\* newnNde);

Node\* removeNodeFromFront();

void removeNodeFromBack();

void addNewNodeBeforeTargetNode(Node\* newNode, int v);

void displayDoublyLinkedList();

bool isPalindrome();

void split(int n);

void drawDoublyLinkedList();

private:

Node\* head;

Node\* tail;

};

#endif

Node.cpp

#ifndef NODE\_CPP

#define NODE\_CPP

#include "Node.h"

Node::Node()

{

prev = NULL;

value = 0;

next = NULL;

}

Node::Node(int v)

{

prev = NULL;

value = v;

next = NULL;

}

Node::~Node()

{

}

Node.h

#pragma once

#ifndef NODE\_H

#define NODE\_H

#include <iostream>

using namespace std;

class Node {

friend class DoublyLinkedList;

public:

Node();

Node(int v);

~Node();

private:

Node\* prev;

int value;

Node\* next;

};

#endif

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated