#include <iostream>

#include <string>

using namespace std;

class Node {

public:

explicit Node(string data = "");

string m\_data;

Node \*next{};

};

Node::Node(string data) {

m\_data = std::move(data);

}

class LinkList {

private:

Node \*pList;

int m\_length;

public:

LinkList();

~LinkList();

void ClearList();

bool InserHead(Node \*node);

bool InserTail(Node \*node);

bool InsertByIndex(int index, Node \*node);

void Traverse();

int CountMr();

};

LinkList::LinkList() {

pList = new Node();

pList->next = nullptr;

m\_length = 0;

}

LinkList::~LinkList() {

ClearList();

delete pList;

}

void LinkList::ClearList() {

Node \*currentNode = pList->next;

while (!currentNode->next) {

Node \*temp = currentNode->next;

delete currentNode;

currentNode = temp;

}

pList->next = nullptr;

m\_length = 0;

}

bool LinkList::InserHead(Node \*node) {

if (!node) {

return false;

}

Node \*temp = pList->next;

Node \*newNode = new Node;

newNode->m\_data = node->m\_data;

pList->next = newNode;

newNode->next = temp;

m\_length++;

return true;

}

bool LinkList::InserTail(Node \*node) {

Node \*currentnode = pList;

while (currentnode->next != nullptr) {

currentnode = currentnode->next;

}

Node \*newNode = new Node;

newNode->m\_data = node->m\_data;

currentnode->next = newNode;

newNode->next = nullptr;

m\_length++;

return false;

}

bool LinkList::InsertByIndex(int index, Node \*node) {

Node \*currentNode = pList->next;

for (int i = 0; i < index - 1; ++i) {

currentNode = currentNode->next;

}

if (currentNode == nullptr || currentNode->next == nullptr) {

InserTail(node);

return true;

}

Node \*temp = currentNode->next;

Node \*newNode = new Node;

newNode->m\_data = node->m\_data;

currentNode->next = newNode;

newNode->next = temp;

m\_length++;

return true;

}

void LinkList::Traverse() {

Node \*currentNode = pList->next;

while (currentNode != nullptr) {

cout << currentNode->m\_data << " ";

currentNode = currentNode->next;

}

}

int LinkList::CountMr() {

int count = 0;

Node \*currentNode = pList->next;

for (int i = 0; i < m\_length; ++i) {

string str;

for (int j = 0; j < 3; ++j) {

str += currentNode->m\_data[j];

}

if (str == "Mr.") count++;

currentNode = currentNode->next;

}

return count;

}

int main() {

Node \*node1 = new Node("Mr.Black");

Node \*node2 = new Node("Mr.Green");

Node \*node3 = new Node("Mrs.Blue");

auto \*pList = new LinkList;

pList->InserTail(node1);

pList->InserTail(node2);

pList->InserTail(node3);

cout << "初始化链表" << endl;

pList->Traverse();

cout << endl;

cout << node1->m\_data[2] << endl;

Node \*node4 = new Node("Mr.Wrown");

pList->InserHead(node4);

cout << "插入链表头" << endl;

pList->Traverse();

cout << endl;

Node \*node5 = new Node("Mrs.White");

pList->InserTail(node5);

cout << "插入链表尾" << endl;

pList->Traverse();

cout << endl;

Node \*node6 = new Node("Dr.Martin");

pList->InsertByIndex(2, node6);

cout << "插入链表第2个元素后" << endl;

pList->Traverse();

cout << endl;

cout << "以Mr.为称谓的人员数量为" << pList->CountMr() << endl;

delete pList;

delete node1;

delete node2;

delete node3;

delete node4;

delete node5;

delete node6;

return 0;

}