CMPSC 122 Lab 4 Report

Code

// Programmer: Yizhou Wang

// Section: 2

// Lab: 4

// Date: Jan. 30, 2014

// Description: An exercise for hardcoded linked list.

#include <iostream>

using namespace std;

struct NodeType; //Define a new data type NodeType

typedef NodeType\* NodePtr; //Give the type an alias

//The fields of the linked list

struct NodeType

{

int data;

NodePtr next;

};

void traverseList(NodePtr head)

//PRE: head is initialized

//POST: A linked list starting with head will be printed. There is a space between every

// two neighboring elements

{

NodePtr cur; //A temporary pointer to hold the node

// we are working on

cur = head; //Get cur started at the head of the linked list

while (cur != NULL) //Go through the list from the head to the tail.

// Hit every node and print their key

{

cout << (\*cur).data << " ";

cur = (\*cur).next; //Move the cur point points to the next node

}

};

int main()

{

//STEP 1: Create a head node and store 5 in the first location.

NodePtr head; //Create the head

NodePtr temp; //Temporary pointer pointing to the new node

head = new NodeType; //Allocate a pointee to the head

(\*head).data = 5; //Store 5 as the key of the first node

(\*head).next = NULL; //Initialize successor of the first node

cout << "STEP 1:" << endl; //Print out all the keys in the list

traverseList(head);

cout << "\n" << endl; //Space between steps

//STEP 2: Now insert 8 at the beginning of the list.

temp = new NodeType; //Allocate the temp pointee

(\*temp).data = 8; //Store 8 as key in the new node

(\*temp).next = head; //Point the new node to the 5 node

head = temp; //Point head to where temp points

cout << "STEP 2:" << endl; //Print out all the keys in the list

traverseList(head);

cout << "\n" << endl; //Space between steps

//STEP 3: Insert 12 at the end of the list

temp = new NodeType; //Allocate the temp pointee

(\*temp).data = 12; //Store 12 as key in the new node

(\*temp).next = NULL; //Point the new node to NULL as it is the last

// node of the list

(\*((\*head).next)).next = temp; //Point 5 node to the 12 node

cout << "STEP 3:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

//STEP 4: Insert 7 between 5 and 12

temp = new NodeType; //Allocate the temp pointee

(\*temp).data = 7; //Store 7 as key in the new node

(\*temp).next = //point the 7 node to the 12 node

(\*((\*head).next)).next;

(\*((\*head).next)).next = temp; //Point node 5 to node 7

cout << "STEP 4:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

//STEP 5: Delete the node 5

temp = (\*head).next; //Allocate temp to point to node 5

(\*head).next = (\*temp).next; //Point node 8 to node 7

delete temp; //Deallocate the 5 node

cout << "STEP 4:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

return 0;

};

Sample Runs

C:\Users\yxw186\Desktop>a

STEP 1:

5

STEP 2:

8 5

STEP 3:

8 5 12

STEP 4:

8 5 7 12

STEP 4:

8 7 12

Discussion

Temporary pointer is an important tool when manipulating the linked list.

It seems that nested “next” field is the only way to determine the absolute location of a node in one linked list. It can be tedious to do so when the number of elements grows bigger.

Revised Code with New Operator

// Programmer: Yizhou Wang

// Section: 2

// Lab: 4

// Date: Feb. 05, 2014

// Description: An exercise for hardcoded linked list.

#include <iostream>

using namespace std;

struct NodeType; //Define a new data type NodeType

typedef NodeType\* NodePtr; //Give the type an alias

//The fields of the linked list

struct NodeType

{

int data;

NodePtr next;

};

void traverseList(NodePtr head)

//PRE: head is initialized

//POST: A linked list starting with head will be printed. There is a space between every

// two neighboring elements

{

NodePtr cur; //A temporary pointer to hold the node

// we are working on

cur = head; //Get cur started at the head of the linked list

while (cur != NULL) //Go through the list from the head to the tail.

// Hit every node and print their key

{

cout << cur -> data << " ";

cur = cur -> next; //Move the cur point points to the next node

}

};

int main()

{

//STEP 1: Create a head node and store 5 in the first location.

NodePtr head; //Create the head

NodePtr temp; //Temporary pointer pointing to the new node

head = new NodeType; //Allocate a pointee to the head

head -> data = 5; //Store 5 as the key of the first node

head -> next = NULL; //Initialize successor of the first node

cout << "STEP 1:" << endl; //Print out all the keys in the list

traverseList(head);

cout << "\n" << endl; //Space between steps

//STEP 2: Now insert 8 at the beginning of the list.

temp = new NodeType; //Allocate the temp pointee

temp -> data = 8; //Store 8 as key in the new node

temp -> next = head; //Point the new node to the 5 node

head = temp; //Point head to where temp points

cout << "STEP 2:" << endl; //Print out all the keys in the list

traverseList(head);

cout << "\n" << endl; //Space between steps

//STEP 3: Insert 12 at the end of the list

temp = new NodeType; //Allocate the temp pointee

temp -> data = 12; //Store 12 as key in the new node

temp -> next = NULL; //Point the new node to NULL as it is the last

// node of the list

(head -> next) -> next = temp; //Point 5 node to the 12 node

cout << "STEP 3:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

//STEP 4: Insert 7 between 5 and 12

temp = new NodeType; //Allocate the temp pointee

temp -> data = 7; //Store 7 as key in the new node

temp -> next = //point the 7 node to the 12 node

(head -> next) -> next;

(head -> next) -> next = temp; //Point node 5 to node 7

cout << "STEP 4:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

//STEP 5: Delete the node 5

temp = head -> next; //Allocate temp to point to node 5

head -> next = temp -> next; //Point node 8 to node 7

delete temp; //Deallocate the 5 node

cout << "STEP 4:" << endl; //Print out all keys in the linked list

traverseList(head);

cout << "\n" << endl; //white space

return 0;

};

New Sample Run

C:\Users\yxw186\Desktop>a

STEP 1:

5

STEP 2:

8 5

STEP 3:

8 5 12

STEP 4:

8 5 7 12

STEP 4:

8 7 12