**Lecture 4 - Linked list using a class**

4 Files: Student.h - struct Student specification

StudentList.h - class StudentList specification

StudentList.cpp - StudentList functions

main.cpp

struct Student

Note: The only private member of the class is a

id : integer

name : string

\*next : Student

pointer (*head*) that points to the head of

a linked list of Student struct objects.

class StudentList

- \*head : Student

+ StudentList() :

+ ~StudentList() :

+ appendStudent(idNum : integer, name : string): void

+ insertStudent(idNum : integer, name : string): void

+ deleteStudent(idNum : integer): void

+ displayList() : void

// ==================================================

// **File: Student.h**

// ==================================================

#pragma once

#include <iostream>

#include <string>

using namespace std;

struct Student

{

int id;

string name;

Student\* next;

};

// ==================================================

// ==================================================

// **File: StudentList.h**

// ==================================================

#ifndef STUDENTLIST\_H

#define STUDENTLIST\_H

#include "Student.h"

class StudentList

{

private:

Student\* head;

public:

StudentList(); // Constructor

~StudentList(); // Destructor

// Linked list operations

void appendStudent(int idNum, string name);

void insertStudent(int idNum, string name);

void deleteStudent(int idNum);

void displayList() const;

};

#endif

// ==================================================

// ==================================================

// **File: StudentList.cpp**

// ==================================================

#include "StudentList.h"

StudentList::StudentList() // Constructor

{

head = nullptr;

}

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

// Destructor - This function deletes every node in the list.

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

StudentList::~StudentList()

{

Student \*temp = head; // To traverse the list

Student \*nextNode; // To point to the next node

// While temp is not at the end, move temp from node to node.

while (temp != nullptr)

{

nextNode = temp->next; // Save a pointer to the next node.

delete temp; // Delete the current node.

temp = nextNode; // Position temp at the next node.

}

}

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

void StudentList::appendStudent(int idNum, string name)

{

Student\* temp = new Student; // Allocate a new node

Student\* move; // Use this pointer to move to the end of the list

temp->id = idNum;

temp->name = name;

temp->next = nullptr;

// If there are no nodes in the list, make the node pointed

// to by *temp* the first node.

if (!head) // Same as if (head == nullptr)

head = temp;

else // Otherwise, insert new *Student* at the end.

{

// Initialize move to head of list.

move = head;

// Find the last node in the list.

while (move->next) // Same as while (move->next != nullptr)

move = move->next;

// Insert newNode as the last node.

move->next = temp;

}

}

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

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

// *displayLis*t shows the values stored in each node of the linked list.

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

void StudentList::displayList() const

{

Student\* temp = head;

// While *temp* points to a node, traverse the list.

while (temp) // Same as: while (temp != nullptr)

{

// Display the student

cout << "ID: " << temp->id << endl

<< "Name: " << temp->name << endl << endl;

temp = temp->next; // Move to the next node.

}

}

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

void StudentList::insertStudent(int idNum, string name)

{

Student\* temp = new Student;

temp->id = idNum;

temp->name = name;

// If there are no nodes in the list make the

// node pointed to by *temp* the first node.

if (!head)

{

head = temp;

temp->next = nullptr;

}

else // Otherwise, insert the new *Student* node

{

Student\* lead = head; // Position lead at the head of list

Student\* follow = nullptr;

// Skip all nodes whose *id* is less than *idNum*.

while (lead != nullptr && lead->id < idNum)

{

follow = lead;

lead = lead->next;

}

// If the new node is to be the 1st in the list, insert it before all other nodes.

if (follow == nullptr)

{

head = lead;

temp->next = lead;

}

else // Otherwise insert after the previous node.

{

follow->next = temp;

temp->next = lead;

}

}

}

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

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

// The deleteStudent function searches for a node with an ID.

// If the ID is found, the node is deleted from the list and from memory.

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

void StudentList::deleteStudent(int idNum)

{

Student\* lead;

Student\* follow; // To point to the previous node

// If the list is empty, do nothing.

if (!head)

return;

// If the first node is the one to be deleted, first move head to

// the next node, then delete the first one

if (head->id == idNum)

{

lead = head->next;

delete head;

head = lead;

}

// Otherwise, the node is in the middle of the list, or the last one in the list, or not in the list

else

{

// Initialize lead to the head of the list

lead = head;

follow = head;

// Skip all nodes whose ID is not equal to *idNum.*

while (lead->next != nullptr && lead->id != idNum)

{

follow = lead;

lead = lead->next;

}

// If the ID is not in the list

if (lead->id != idNum)

{

cout << "The ID number is not in the list.\n\n";

}

// check to see if it's the last one in the list

else if (lead->next == nullptr && lead->id == idNum)

{

follow->next = nullptr;

delete lead;

}

// It must be a node in the middle of the list - Not the first or last

// Link the previous node pointed to by follow, to the node after lead, then delete lead.

else

{

follow->next = lead->next;

delete lead;

}

}

}

// ==================================================

// ==================================================

// **File: main.cpp**

// ==================================================

#include "StudentList.h"

int main()

{

char answer;

int id;

string name;

StudentList list;

cout << "Insert Students into the list:\n\n";

do {

cout << "ID: ";

cin >> id;

cin.ignore();

cout << "Name: ";

getline(cin, name);

list.insertStudent(id, name);

cout << "Insert another record (Y or N)? ";

cin >> answer;

} while (toupper(answer) == 'Y');

system("cls");

cout << "Here is the list:\n\n";

list.displayList();

return 0;

}

// ==================================================