/\* Assignment: Name Grade Sorted Link List

Author: Ryan Wood

Created On: February 11, 2018

Purpose: demonstrate use of Linked List structure

\*/

/\*Student structure\*/

#ifndef STUDENT\_H

#define STUDENT\_H

#include <string>

/\*the Student structure for our Linked List\*/

struct Student

{

std::string name;

double grade;

Student \*nameLink;//Link to order list by name

Student \*gradeLink;//link to order list by grade

Student \*aStudentLink;//link to order list by A students only

};

typedef Student\* node;//make the Student poinbter easier to deal with

#endif

/\* Assignment: Name Grade Sorted Link List

Author: Ryan Wood

Created On: February 11, 2018

Purpose: demonstrate use of Linked List structure

\*/

/\*Linked List Definition file\*/

#ifndef LINKED\_LIST\_H

#define LINKED\_LIST\_H

#include <string>

#include "Student.h"

class LinkedList

{

public:

/\* Function: LinkedList()

Purpose: default constructor. sets head to null

and opens the output file

Return: an instance of an empty LinkedList

\*/

LinkedList();

/\* Function: ~LinkedList()

Purpose: default destructor. Closes the output file

\*/

~LinkedList();

/\* Function: getNumNodes()

Purpose: retrieves the number of nodes in the list

Return: the number of elements

\*/

int getNumNodes();

/\* Function: insert(string, double)

Purpose: inserts a new node having the given name and grade

into the list in order of grade, name, AND A student

Parameters: the name, the grade as a percentage

\*/

void insert(std::string, double);

/\* Function: print()

Purpose: prints the list in name order

\*/

void print();

/\* Function: printByGrade()

Purpose: prints the list in grade order

\*/

void printByGrade();

/\* Function: printAStudents()

Purpose: prints the A students that are in the list

in order of their grades from greatest to least

\*/

void printAStudents();

private:

node head;//pointer to the first student

};

#endif

/\* Assignment: Name Grade Sorted Link List

Author: Ryan Wood

Created On: February 11, 2018

Purpose: demonstrate use of Linked List structure

\*/

/\*Linked List implementation file\*/

#include "Student.h"

#include "LinkedList.h"

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

ofstream outFile;

LinkedList::LinkedList()

{

head = NULL;

outFile.open("StudentOutput.out");

outFile << fixed << showpoint << setprecision(2);

cout << fixed << showpoint << setprecision(2);

}

LinkedList::~LinkedList()

{

outFile.close();

}

int LinkedList::getNumNodes()

{

int total = 0;

node nodePointer = NULL;

nodePointer = head;

while(nodePointer)

{

total++;

nodePointer = nodePointer->nameLink;

}

return total;

}

void LinkedList::insert(string name, double grade)

{

node newNode = NULL;

node nodePtr = NULL;

node prevNode = NULL;

node aStudnt = NULL;

node prevA = NULL;

newNode = new Student;

newNode->name = name;

newNode->grade = grade;

newNode->nameLink = NULL;

newNode->gradeLink = NULL;

if(NULL == head)

{

head = new Student;

head->grade = 0.0;

head->name = "";

head->gradeLink = NULL;

head->aStudentLink = NULL;

}

nodePtr = head->nameLink;

while(NULL != nodePtr && nodePtr->name < name)

{

prevNode = nodePtr;

nodePtr = nodePtr->nameLink;

}

newNode->nameLink = nodePtr;

if(NULL == prevNode)

{

head->nameLink = newNode;

}

else

prevNode->nameLink = newNode;

//reset vars

prevNode = NULL;

nodePtr = head->gradeLink;

while(NULL != nodePtr && nodePtr->grade >= grade)

{

prevNode = nodePtr;

nodePtr = nodePtr->gradeLink;

}

newNode->gradeLink = nodePtr;

if(NULL != prevNode)

prevNode->gradeLink = newNode;

else

head->gradeLink = newNode;

if(grade >= 90)//check a student Links

{

//reset vars

prevNode = NULL;

nodePtr = head->aStudentLink;

while(NULL != nodePtr && nodePtr->grade >= grade)

{

prevNode = nodePtr;

nodePtr = nodePtr->aStudentLink;

}

newNode->aStudentLink = nodePtr;

if(NULL != prevNode)

prevNode->aStudentLink = newNode;

else

head->aStudentLink = newNode;

}

}

void LinkedList::print()

{

node nodePtr = NULL;

cout << "Printing students by name" << endl;

outFile << "Printing students by name" << endl;

nodePtr = head->nameLink;

while(NULL != nodePtr)

{

cout << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

outFile << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

nodePtr = nodePtr->nameLink;

}

cout << endl;

outFile << endl;

}

void LinkedList::printByGrade()

{

node nodePtr = NULL;

cout << "Printing students by grade" << endl;

outFile << "Printing students by grade" << endl;

nodePtr = head->gradeLink;

while(NULL != nodePtr)

{

cout << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

outFile << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

nodePtr = nodePtr->gradeLink;

}

cout << endl;

outFile << endl;

}

void LinkedList::printAStudents()

{

node nodePtr = NULL;

cout << "Printing A students only by grade" << endl;

outFile << "Printing A students only by grade" << endl;

nodePtr = head->aStudentLink;

while(NULL != nodePtr && nodePtr->grade >= 90)

{

cout << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

outFile << left << setw(15) << nodePtr->name << setw(15) << nodePtr->grade << endl;

nodePtr = nodePtr->aStudentLink;

}

cout << endl;

outFile << endl;

}

/\* Assignment: Name Grade Sorted Link List

Author: Ryan Wood

Created On: February 12, 2018

\*/

/\*Student Linked List Main test file\*/

#include "Student.h"

#include "LinkedList.h"

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

using namespace std;

const string fileName = "LinkLnamesAndGrades.txt";

ifstream inFile(fileName);//set up an input file

//set up a structure to hold student info from the file

struct RimesStudent

{

string name;

int gradeTotal;

double gradeValue;

};

int NUM\_STUDENTS = 0;

int MAX\_POINTS = 0;

/\* Function: countStudents()

Purpose: this function loops through the lines of

the input file and counts the number of

lines that are not the first or not students.

Return: the number of students whose data is defined in the file

\*/

int countStudents()

{

int num = 0;

string strName;

while(getline(inFile, strName, '\n'))

{

if(strName.length() > 1)

num++;

}

num--;/\*minus one because first newline comes after

the first line, which is total possible points\*/

return num;

}

/\* Function: fillArray(RimesStudent[])

Purpose: fills the given array of RimesStudents with data

from the file. The array is assumed to have as many

elements as the global NUM\_STUDENTS variable

Parameters: the empty array of students

PostCond: the array is filled with data from the file

\*/

void fillArray(RimesStudent students[])

{

int iGrade = 0;

double dGrade = 0.0;

string strName = "";

RimesStudent \*iter = NULL;

inFile >> MAX\_POINTS;

if(MAX\_POINTS == 0)

{

cout << "Could not obtain the Maximum number of points from the file!" << endl;

return;

}

cout << MAX\_POINTS << endl;

cout << "Maximum points possible = " << MAX\_POINTS << endl;

iter = students;

while(inFile >> strName >> iGrade)

{

iter->name = strName;

iter->gradeTotal = iGrade;

iter->gradeValue = ((double) iGrade/MAX\_POINTS)\*100;

iter++;

}

}

/\* Function: testLinkedList(RimesStudent[])

Purpose: This function goes through the given array of

students and insert each of them into a LinkedList

which it then prints in name and grade order and then

prints only the student who made A's in order of their

grades

Parameters: the array of Rimes Student structures

\*/

void testLinkedList(RimesStudent students[])

{

int count = 0;

RimesStudent \*iter = NULL;

LinkedList \*studentList = new LinkedList;

iter = students;

for(count = 0; count < NUM\_STUDENTS; count++)

{

studentList->insert(iter->name, iter->gradeValue);

iter++;

}

cout << "Printing list in name order" << endl;

studentList->print();

cout << "Printing List in Grade order" << endl;

studentList->printByGrade();

cout << "Printing all A students" << endl;

studentList->printAStudents();

delete(studentList);

}

int main()

{

//how manu students do we have

NUM\_STUDENTS = countStudents();

//create a RimesStudent array with that many

RimesStudent students[NUM\_STUDENTS];

//now that we are at the end of the file, re-initialize it

inFile.close();

inFile.open(fileName);

//fill the array with data from the file

fillArray(students);

//create a LinkedList from the students and test functions

testLinkedList(students);

inFile.close();

return 0;

}

Printing students by name

Abore 98.33

Ace 86.00

AceAgain 81.67

AceMcFace 74.00

Airborn 96.33

Boyanze 78.33

Bozo 67.67

Brown 83.33

ByPass 66.67

Climintime 60.00

Cooldaddy 95.00

Crackley 61.33

Hunter 95.33

Icmoto 87.00

Ikantwait 57.67

Jabbo 55.33

Jayman 58.00

Jedatkins 87.67

Kidkitty 62.33

Knocktosee 97.00

Lewis 92.00

Lion 64.00

Mack 88.33

Mark 92.00

Marki 68.33

Marshall 92.33

Monnazuma 79.67

Moto 54.00

Notquityet 100.00

Oevey 76.67

Passby 91.67

Reynolds 94.00

Runs 83.33

See 84.33

Sid 98.00

Sidcar 98.33

Sidcarson 96.67

Sobeit 89.00

Taylor 95.67

Thomas 65.67

Thompson 96.00

ToSha 94.33

Tony 66.00

Tophinish 97.67

Upupaway 68.33

Vey 95.33

Vicki 66.33

WayOut 59.67

Wayoutyonder 94.00

Zieneth 83.33

Printing students by grade

Notquityet 100.00

Sidcar 98.33

Abore 98.33

Sid 98.00

Tophinish 97.67

Knocktosee 97.00

Sidcarson 96.67

Airborn 96.33

Thompson 96.00

Taylor 95.67

Vey 95.33

Hunter 95.33

Cooldaddy 95.00

ToSha 94.33

Reynolds 94.00

Wayoutyonder 94.00

Marshall 92.33

Lewis 92.00

Mark 92.00

Passby 91.67

Sobeit 89.00

Mack 88.33

Jedatkins 87.67

Icmoto 87.00

Ace 86.00

See 84.33

Brown 83.33

Zieneth 83.33

Runs 83.33

AceAgain 81.67

Monnazuma 79.67

Boyanze 78.33

Oevey 76.67

AceMcFace 74.00

Upupaway 68.33

Marki 68.33

Bozo 67.67

ByPass 66.67

Vicki 66.33

Tony 66.00

Thomas 65.67

Lion 64.00

Kidkitty 62.33

Crackley 61.33

Climintime 60.00

WayOut 59.67

Jayman 58.00

Ikantwait 57.67

Jabbo 55.33

Moto 54.00

Printing A students only by grade

Notquityet 100.00

Sidcar 98.33

Abore 98.33

Sid 98.00

Tophinish 97.67

Knocktosee 97.00

Sidcarson 96.67

Airborn 96.33

Thompson 96.00

Taylor 95.67

Vey 95.33

Hunter 95.33

Cooldaddy 95.00

ToSha 94.33

Reynolds 94.00

Wayoutyonder 94.00

Marshall 92.33

Lewis 92.00

Mark 92.00

Passby 91.67