Trangdai Huynh

CS-300

Terry Atkison

4/9/23

6-2 Assignment: Project One

**Reading the File**

Use fstream to open the file

Create the method void loadCourses (dataStructre, string csvPath)

Make the call to open the file, if return value is “-1” then the file is not found

Else the file is found

While its not the EOF

Read each of the lines

IF There are less than two values in the line, return the ERROR

ELSE read the parameters

IF there is third or more parameter

IF the third or more parameters are in the first parameter elsewhere, then continue

ELSE then return the ERROR

Close the file

**Holding the Course Information**

Create the struct Course{}

Create the Identifers: CourseName, CourseID, and Prerequistite

//**Vector**

Vector<Course>loadCourses(string csvPath)

for (int i=0; I < file.rowCount();i++) {

Create the data structure and add it to the collection of the course

Course course;

course.name=file[i][0];

course.courseId=file[i][1];

while not thte end of the line

course.prereq.=file[i][8];

courses.push\_back(course);

//**HashTable**

Create the Hashtable

Create the Node structure

Course course

Unisigned the int key

Vector<Node>nodes

Define the tableSize

Unisigned int has(int key)

Create the insert method vod HashTable::Inser(Coursecourse)

Create key that is for the given course, search for the node with key value

If there is no entry that is found for the key

Assign the node to key position

else if the node is used

assign the old node key to the UNIT\_MAX, set it to the key, and set the old node

to the course and old the next node to

null pointer

else find next open node

add the new newNODE to the end

void the loadCourses(string csvPath, HashTable\*hashTable)

loop the read rows of the CSV file

for (unsigned int i=0;i<file.roqCount();i++) {

Create the data structre and add it to the collection of the couses

Course course;

course.name=file[i][0];

course.courseId=file[i][1];

while not the end of the line

course.prereq.=file[i][8];

hashTable->Insert(course);

//**Tree**

Define the binary search tree to hold all of the courses

BinarySearchTree\*bst;

Bst=new BinarySearchTree();

Course course;

Create the add node method to void BinarySearchTree::addNode(Node\*node,Course course)

If the root is null, add the root

If node is led than the root, then add it to the left

If there is no left node

this node then becomes left

if the node is greater than the root then add right

if there is no node right

this node then becomes right

void loadCourses(string csvPath, BinarySearchTree\*bst)

loop to read the rows of the CSV file

for (unsigned int i=0; i<file.rowCount();i++) {

Create the data structure and then add it to the collection of the courses

Course course;

course.name=file[i][0];

course.courseId=file[i][1]];

while it is not end of the line

course.prereq.=file[i][8];

bst->Insert(course);

**Print the Course Information and the Prerequisites**

//**Vector**

Create the method void printCourseInformation(Vector<Course>courses, String coursed)

Get the input for the coursed

While the vector is not empty

if input is the same as the coursed

output course.courseId<<output course.name

while(prereq=true)

output the course.prerew

**//HashTable**

Create the method void printCourseInformation(Hashtable<Courser>courses,String coursed)

Get the input for the coursed

Assign the key = courseId

Assign the node tot eh node.at(key)

If the current node does match the key

Return the course, displayCourse(nodes[key].course)

If the node points to the null, return null

Else while node is not the Null, check against key

If the key does match the couseId, Return the course, and displayCourse(nodes[key].course)

Point to next node

//**Tree**

Create the method void printCourseInformation(Tree<Course> courses, String courseId)

Get the input for the coursed

Assign the current node to the root

While the current is not the NULL

If the course.courseId does match the current

Return the current, output course.courseId<<output the course.name

while(prerew=true)

output the course.prerew

If the coursed is less than the root

Set the current to the left

Else set the current to the right

**Menu**

Set the choice to 0;

Create while loop for the menu. While choice is not equal to four

Output the menu choices(1.Load the Course File, 2 Print the Course List, 3. Print the Individual Course. 4Exit)

Create the switch(choice)

Case1: loadCourses(courseFile, dataStructure)

Case2: printSorted(courses)call the function to print sorted class list

Case3: printCourseInformation(coursed)

Case4: Terminate the Program

**Print the Sorted List**

//**Vector**

Create the corted print method printSorted(courses)

Create the partition method int partition(vector<Course>& courses,int begin, int end)

Set the lowIndex to the first element, set the highIndex to the last element

Set the midpoint to the lowIdex+(highIndex-lowIndex)/2

Set the pivot to the midpoint

Decrement the highIndex while pivot is less than the highIndex

Swap the lower values to the left of the pivot, higher values to the right of the pivot

Set the temp value to the low index

Set the low index to the high index

Set the high index to the temp

Create the quicksort method void quicksort(vector<Course>&courses, int begin, int end)

Set the mid to 0, the lowIndex to begin and the highIndex to end

If begin>=end, return

Set the lowEndIndex to partition(courses, lowindes, highindex)

Make the recursive call to the quicksort

quicksort(course, lowIndex, lowEndIndex);

quicksort(courses, lowEndIndex +1,highIndex)

Create the display course method void displayCourses(Course course) {

Cout<<course.courseId<<”:”<<course.name<<”|”<<course.prereq<<endl;

Loop through the vector to the display courses

for(int i=0; i<courses.size{};++i)

displayCourse(courses[i])//Tree

Create the inOrder method void the BinarySearchTree::inOrder(Node\*node)

If(node !=Null)

Check most of the left side first

inOrder(node=>left)

cout<<course.courseID<<”:”<<course.name<<”|”<<course.prereq<<endl;

check the next right leaf

inOrder(node->right)

cout<<course.courseID<<”:”<<course.name<<”|”<<course.prereq<<endl;

Table

Description automatically generated