Edwin Jones

CS300 3-3: Project One Milestone One

constant unsigned int DEFAULT\_SIZE = desired table size

// Defaults to public members

struct CourseInfo {

string code

string name

vector<string> prerequisites

}

//Structure of each course in courses given respective parameters

class HashTable {

private

struct Node {

CourseInfo course

Unsigned int key

Node type point next

Node() {

key = max possible value

next = nullptr

}

Node(CourseInfo course) : Node() {

this course = course

}

Node(CourseInfo course, unsigned int key) : Node(course) {

this key = key

}

}

vector<Node> courses

unsigned int tableSize = DEFAULT\_SIZE

unsigned int hash(int key)

public

HashTable() {

resize courses to tableSize

}

HashTable(unsigned int size) {

set tableSize = size

resize courses to tableSize

}

~HashTable() { //delete everything in the table when class is destroyed

traverse courses and at each course at index i

create Node pointer current of type Node = the next node

while current != nullptr

create a new pointer toDelete = current

set current = next course after itself

delete toDelete

}

hash(int key) { //use input parameter as the hash key

return the resulting hash value

}

Insert(CourseInfo course) { // Assign course to a bucket and integrate into linkedlist

set unsigned int currentKey = hash(the course info to hash)

if the course at currentKey has key = max possible value

set course at currentKey = Node(course, currentKey)

else

create Node pointer nextCourse = content of course at currentKey

while the course after nextCourse != nullptr

set nextNode = the course after itself

set the course after nextNode = a new pointer for Node(course, currentKey)

}

void validateCourses(){ // Verify that each unique course prerequisite is listed as a course

vector<string> tempPrereqs

vector<string> tempCourses

//The courses to compare prequisites against

for each Node course in courses

add course.code to the end of the tempCourses vector

//check each course for their prerequisites

for each Node course in courses

//check if the prerequisite course is listed as course

for each string prereq in course.prerequisites

if prereq is in tempCourses

continue //check next course

else

throw an error //course not found, invalid

// delete the temp vectors from memory since no errors

vector<string>().swap(tempPrereqs)

vector<string>().swap(tempCourses)

}

void PrintAll() { //Iterate through all the courses and print the information

for each Node course in courses

if the course key != max possible value

print every course attribute

create Node pointer next = next course

while next != nullptr

print every course attribute

set next = course after course pointed to by next

}

void searchCourse(string courseCode) { // iterate through matching bucket for matching code

create an empty CourseInfo course object

set unsigned int currentKey = hash(the course info to hash)

if the code of course at currentKey = courseCode

return the course's info

else

create Node pointer nextCourse = next course after course at currentKey

create Node point prevCourse = course at currentKey

while next != nullptr

if the course's code at nextCourse = courseCode

return nextCourse bid object

set prevCourse = nextCourse

set nextCourse = course after nextCourse

return course //empty values

}

}

void parseCSV(string csvFile, HashTable \*courseTable) { //Read the CSV file and format content for the hash table

ifstream file(csvFile)

string line

if file is not open

throw an error

while there is a row, assign the content of the row to line

if the length of the line is 0

continue

if the end of the line doesn’t have a comma

add a comma to the end

if the number of commas found is < 2

throw an error

create an empty CourseInfo course object

set course.code = string before first comma

delete index 0 to index of comma

set couse.name = string before first comma

delete index 0 to index of comma

while there are more than 1 chars in line

if there is no comma in line

add line to end of course.prerequisites vector

break

add string before next comma to the end of the course.prerequisites vector

set line = line with index 0 index of comma deleted

use Insert(course) to add to courseTable

close file

}

main { //Follow lines achieve basic functionality

//Process, store, then validate CSV file content

assign variable with CSV file

Define the hash table to store content in

create an empty CourseInfo course object

set the hash table = a new HashTable() instance

pass the CSV file variable through parseCSV()

call the validateCourses method

//View all information in the hash table

call the PrintAll() method

//View specific course information

pass a course's code through the searchCourses() method

}