Phillip Hogan

CS300

4/7/2022

Project One Milestone 2 – Pseudocode

Class Courses{

Private:

String courseNumber

String title

String prereq1

String prereq2

Public:

Void setCourseNum(string cn) {

Coursenumber = cn

}

Void setTitle(string t) {

Title = t

}

Void setPrereq1(string pr1) {

Prereq1 = pr1

}

Void setPrereq2(string pr2) {

Prereq2 = pr2

}

String GetCourseNumber(){

Return courseNumber

}

String GetTitle(){

Return Title

}

String getPrereq1(){

If prereq is not empty

Return prereq1

Else

Return None

}

String getPrereq2(){

If prereq is not empty

Return prereq2

Else

Return None

}

Int numPrerequisiteCourses(vector<Courses> courses, Course c) {

Totalrerequisites = prerequisites of course c

For each prerequisite p in totalPrerequisites

Add prerequisites of p to totalPrerequisites

Pring number of totalPrerequisites

Void printSampleSchedule(Vector<Courses> courses){

For each course in vector

Print out all information for course

}

Void printCourseInformation(vector<Courses> courses, String courseNumber){

For all courses

If the course is the same as courseNumber

Print out the course information

For each prerequisite of the course

Print the prerequisite course information

}

const unsigned int DEFAULT\_SIZE = 100

Class HashTable {

private:

Struct Node {

Courses curr

Unsigned int key

Node pointer next

Node(){

key = UINT\_MAX

next = null pointer

}

Node(Courses cc) : Node(){

Curr = cc

}

Node(Course cc, unsigned int aKey) : Node(cc) {

key = aKey

}

};

vector<Node> curriculum

unsigned int tableSize = DEFAULT\_SIZE

unsigned int hash (int key)

public:

HashTable()

Virtual ~HashTable()

hash(int key)

void Insert(Courses curr)

int numPrerequisiteCourses(Hashtable<Courses> curr)

void printSampleSchedule(Hashtable<Courses> curr)

void printCourseInformation(Hashtable<Courses> curr, String courseNumber)

}

HashTable::HashTable(){

Resize node to tableSize

}

HashTable::~HashTable(){

Delete all nodes

}

Unsigned int HashTable::hash(int key) {

return remainder of key divided by tableSize

}

void HashTable::Insert(Courses curr){

create key from course number

create pointer node to key location

if node is empty

create new node

add node to table

else if node is not empty

if current node key is equal to UINT\_MAX

current node data equals courses data

else if current key does not equal UINT\_MAX

while current node pointer is not null

current node equal next node.

}

int HashTable::numPrerequisiteCourese(HashTable< Courses> curr) {

iterate through hashtable

once course has been found

count prereq

return count

}

void printSampleSchedule(HashTable<Courses> curr) {

Iterate through hashtable

Print each course object

}

void printCourseInformation(Hashtable<Courses> curr, String courseNumber){

Search for course in hashtable

Print information for course when found

}

main() {

Hashtable\* courses

Courses curriculum

ifstream file

Open file

string crNum

string title

string prereq1

string prereq2

string line

int count = 0

int i

int choice

int sumPre = 0

unsigned int key

while (choice does not equal){

print “Menu: “ end line

print “1. Load Curriculum” end line

print “2. Search Specific Course” end line

print “3. Print Full Curriculum” end line

print “4. Sum of Prerequisites” end line

print “0. Exit” end line

read in next input into variable choice

switch statement(choice) {

case 1:

open file

if (file is not open){

print “Could not open file document”

}

While file is not at the end of the file {

Get the first line of the file

For each ‘,’ in the line

Increment count by 1

While line is not empty{

If count = 3

crNum = “ “

CrNum = text before the comma

currCourses course number setter is set to crNum

Delete the comma and everything before it from line

Title = “ “

Title = text before the comma

currCourses title setter is set to title

delete the comma and everything before it from line

prerequisite1 = “ “

prerequisite1 = text before the comma

currcourses prerequisite1 setter is set to prerequisite1

delete the comma and everything before it from line

prerequisite2 = “ “

prerequisite2 = text before the comma

currcourses prerequisite2 setter is set to prerequisite2

delete the comma and everything before it from line

else if (count = 2)

crNum = “ “

CrNum = text before the comma

currCourses course number setter is set to crNum

Delete the comma and everything before it from line

Title = “ “

Title = text before the comma

currCourses title setter is set to title

delete the comma and everything before it from line

prerequisite1 = “ “

prerequisite1 = text before the comma

currcourses prerequisite1 setter is set to prerequisite1

prerequisite2 = “ “

currcourses prerequisite2 setter is set to prerequisite2

delete the comma and everything before it from line

else if (count = 1)

crNum = “ “

CrNum = text before the comma

currCourses course number setter is set to crNum

Delete the comma and everything before it from line

Title = “ “

Title = text before the comma

currCourses title setter is set to title

prerequisite1 = “ “

currcourses prerequisite1 setter is set to prerequisite1

prerequisite2 = “ “

currcourses prerequisite2 setter is set to prerequisite2

delete the comma and everything before it from line

Push newly populate course object into vector

Count = 0

}

Print Curriculum Loaded Successfully end line

Break

Case 2:

Print “Enter Course Number” end line

Read input into crNum

For (I = 0; I < curriculum size; increment i){

If (crNum equals curriculum at index I.getCourseNumber){

Print curriculum

}

If ( I equals curriculum size – 1 and crNum does not equal curriculum at index i) {

Print “Error: Course Not Found” end line

}

Break

Case 3:

For ( I = 0; I is less that curriculum size; increment i){

Print course at index i

}

Case 4:

Print new line

Print “Enter Course Number” end line

Read in input into crNum

For (I = 0; I is less than curriculum size; increment i) {

If (crNum is equal to curriculum at index i) {

sumPre = sumPre + curriculum count function

print new line

print “Prerequisite Total For Course: “ + sumPre

print new line

print course at index i

}

If (I = curriculum size – 1 and crNum does not equal curriculum at index i){

Print “Error: Course Not Found”

}

}

Break

}

Print “Good Bye.”