# Philip Trinh

SNHU/ CS300

03/25/2023

# CS 300 Pseudocode Document Hashtable

// OPEN AND READ FILE, PARSES EACH LINE, AND CHECKS FOR FILE FORMAT ERRORS

Use fstream to open file “File Name”

If file does not exist, print “file not found, please check name”

Else file found

While it is not end of file

For each line of the file

Strip white space from beginning and end of line

Split the line into parameters by the commas

If less than two parameters return from each line

Return ERROR, invalid format

Else

Format is valid, read the parameters

If the third or more parameters exist

Compare if it exist at the beginning of the line somewhere else

Continue

Else

Return Error, prerequisite not in list

Close file

//CREATE COURSE OBJECTS AND STORE THEM IN THE APPROPRIATE DATA STRUCTURE

// Hashtable pseudocode

Create Structures to hold course, struct Node {}

Course course

Unsigned int key

Node\*next

Create default constructor Node() {

Key = UINT\_MAX

Next = nullptr

Initialize with course and a key

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

Key = akey

}

Define tableSize

Unsigned int hash(int key)

Create default constructor for HashTable

Initialize node structure by resizing tableSize

Invoke local tableSize with this->

Resize node size

Create Destructor

Erase nodes beginning

Implement logic to calculate hash value by

Define Unsigned int hash(int key)

Return key % tableSize

Create insert method void HashTable

create the key for the given course, courseNumber, courseName, with the key value

Retrieve node using key

if no entry found for the key

assign this node to the key position

else if node is not used

assign old node key to UINT\_MAX, set to key, set old node to course, update with the actual course and old node to next nullptr

else if not found find the next open node available

add new newNode to end

Create new object struct Course {}

Create String unique identifiers courseNumber, courseName, Prequisisite

Loop through each line using for (int i=0; i<file.rowCount(); i++){

Create data structure and add courses

Set parameter[0] to courseNumber

Set parameter[1] to courseName

Set parameter >1 add it to prerequisite}

hashTable->Insert(course);

// PRINT OUT COURSE INFORMATION AND PREREQUISITES

int numPrerequisiteCourses(Hashtable<Course> courses) {

Take input for courseNumber = couse c

Assign key = courseNumber

Assign node to the node at key

currentList = Hashtable of item pointing to key

If current node pointing to currentList match course c inputted and not null

Return prerequisites of course c

Else

Return null

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

add prerequisites of p to totalPrerequisites

print number of totalPrerequisites

}

void printSampleSchedule(Hashtable<Course> courses) {

for all courses from node begin to end

print courseNumber and CourseName

if course have prerequisites

for each prerequisite

print prerequisites

node is equal to next node

}

void printCourseInformation(Hashtable<Course> courses, String courseNumber) {

for all courses

get course at table index of hashed courseNumber

if the course is the same as courseNumber

print out the course information

for each prerequisite of the course

get course at table index of hashed prerequisite courseNumber

print the prerequisite course information

}