Brandi Noell

Project 1

CS 300

April 14, 2023

Project 1 Pseudocode

Vector Pseudocode

//opening files and reading files

//Defining the Course Structure

Class Couse {

String course Number

String course Name

Create a Vector to String the Prerequisites

Create the constructor

//Loading a file, opening file, and reading file

Create a vector <string> to OpenFile (string “FileName”)

IF file is opened

Open FileName with instream

If the file doesn’t open with instream

Output error “file cannot be opened”

Loop back to the line with contents

Close contents using the instream

Return to courses

//Creating a course for each line

Create a Vector <String> for courseNumber and searchCourses

FOR  the courses in course vector

IF courseNumber equals to courseNum

Print the course Number and Name

FOR the preReqs in the course prerequisites

Print the preReqs

// Printing course information

Open FileName

While filename is open

Print course information

Return

}

Hash Table Pseudocode

//We must define the course structure

Structure Course Class {

String courseNumber;

String courseName;

Vector<string>prereq;

//We need to print the hashtable

Void Print()

{

Print courseName and CourseNumber

Print preReq to output

}

}

Constant unsigned int Default\_size = 8;

//Create hashtable class

Class hashTable

Struct Node {

Course course;

Unsigned int key;

Node next;

Node() {

Key = Uint\_Max

Next = nullptr;

}

//course and coursekey integer

Node(Course course, unsigned int key)

Unsigned int size = Default\_size;

Vector<Node>table;

HashTable();

Insert (course course);

}

Int numPrereqCourses(HashTable<Course> courses, String courseNumber)

{

//Create keys for course and addd it to the hash table in the key position.

Create key for courseNumber from hashing

Retrieve the node by using the key and it’s new node variable

WHILE node ≠ nullptr

If (node == nullptr)

totalPrereq = node.Prereq.size();

For each prereq in total Prereq

Add prereq of P to totalPrereq

Print # of totalPrereq

Else

Node ( == node.pointer)

//Print to course

Void pritnCourseInfo (Hashtable<Course>course, String courseNumber)

{

Create an empty course object

WHILE node is ≠ nullptr

If (nod course courseNumber == courseNum) return node course

Print course information

FOR each prereq of course

Print course prereqs

Break;

ELSE

Node = net node

}

Unsigned int hash(int key) {

Return key % and table size;

Void insert (course course)

Unsigned key = hash(stoi(course.courseID)

Check to see if node is empty

IF node is empty

Insert course for the node

If it is not empty

While loop until an empty node is found

Insert course for the node

}

//Open store, and read courses from the hash table file

Void LoadCourses(string filename, Hashtable and Hashtable)

{

IF (line size = 2)

Course newCourse

Course courseName = line[0]

Course courseNumber = line[1]

Set course prereq to empty vector

Return newCourse

ELSE

Vector<string>TempPrereq

FOR (int I = 2; I < line.size(); i++\_

tempPrereq.pushback(line[i]);

course newCourse

course coureNumber[1]

set course prereq to tempPrereq vector

return newCourse

Add course to HashTable

Clear linestream for next line;

}

Pseudocode Tree Data

Use fstream at START to open the file

CALL to open the file

IF the returned value is less than one

RETURN “Error, can’t find file”

ELSE the file can be found

WHILE not the end of the file

READ each line of the file

IF there are less than two values found in a line

RETURN “Error in parameter”

ELSE read the parameters found

IF there are values found greater than or equal to three

Continue with file

ELSE

Return “Error”

CLOSE the file

*Pseudocode for a course object*

START the program

INITIALIZE the vector to create a course structure node

WHILE it’s not the end of the file

FOR each line of the file

FOR the 1st and 2nd values of the file

ADD a course ID and course Name

IF there is a 3rd value that exists

ADD the prerequisites until a new line can be found

*Creating a Binary Tree*

DEFINE the Binary Tree and what it is

CREATE a root that points to null

CREATE a method of insert

IF the root null

The current course is the root

ELSE

IF the course number is less than the root

ADD to the left node

IF the left node is equal to null

ADD the course number

ELSE

IF the course number is less than half of the leaf

ADD a value to the left node

IF the course number is greater than half the leaf

ADD a value to the right node

ELSE

IF the right is equal to null

ADD the course number

IF the course number is greater than the root

ADD the course to the left side of the binary tree

IF the course number is less than the leaf

ADD course to the right side of the leaf

*Pseudocode for printing the course and the prerequisites information*

SET the current node equal to the root

ASK the user for their INPUT

IF root does not equal null

Traverse Left

ELSE if the course number is greater than zero

Traverse Right

LOOP until a match can be found

ELSE

PRINT “error, course can not be found”

END

The advantages for vector data is it is one dimension making it easier to handle and uses less memory.

The disadvantage of vector data is that the elements cannot be deleted.

Hash tables seem to be the best for use because they are organized at how they store the information.