# Philip Trinh

SNHU/ CS300

04/02/2023

# CS 300 Pseudocode Document Binary Tree

Void fileInput(Tree<Course> courses) {

// 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

Create new courseObject

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

Class Course

void courseObject(Tree<Course> courses, String courseNumber, String courseName, String prerequisite) {

for each line read

store courseNumber

store courseName

if there is more than two value

Compare if that value same as courseNumber at the beginning of line somewhere else

store prerequisite

Else

Return Error, “error prerequisite does not exist as a course within the list”

//Binary Tree Pseudocode sort, add, and print information

}

End CLASS

create struct Course {

string courseNumber

string courseName

vector Prerequisite

}

create struct Node {

Course course

Node left

Node right

}

CLASS Tree

PRIVATE:

Initialize Node type root

void addNode and pass with Node type node, Course type course

void inOrder and pass with Node type node

void preOrder and pass Node type node

void postOrder and pass Node type node

End PRIVATE

PUBLIC:

Tree()

Virtual Tree()

Void Inorder, PostOrder, PreOrder, Insert (Course course)

End PUBLIC

End CLASS

create destructor tree

set root to nullptr

Traverse the tree in order

Reference tree: InOrder(), PostOrder, PreOrder to root

Reference tree: insert (Course course)

If course is empty

Root = add new node

Else

This -> addNode root and course

Tree in addNode(Node type node, Course course {

Implement inserting course into tree by

If node is larger then add to the left

If no left node

This node become left node

Else recurse down the left node

Else if node is lesser then add to right

if no right node

this node become right

Else recurse down the right node

Tree in inOrder(Node type node)

If node is not equal to nullptr

inOrder node point to left

output courseNumber, courseName, prerequisite

inOrder node point to right

Tree in postOrder(Node type node)

If node is not equal to nullptr

postOrder node point to left

postOrder node point to right

output courseNumber, courseName, prerequisite

Tree in preOrder(Node type node)

If node is not equal to nullptr

output courseNumber, courseName, prerequisite

postOrder node point to left

postOrder node point to right

// Tree pseudocode

int numPrerequisiteCourses(Tree<Course> courses) {

totalPrerequisites = combine of left and right children of courses

for each prerequisite p in totalPrerequisites

add left and right nodes to the totalPrerequisites

print totalPrerequisites

}

void printSampleSchedule(Tree<Course> courses) {

for all Nodes as course

print courseName

if there is left node or right node

print node as prerequisites

}

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

for all nodes

if the course is same as courseNumber

print out the course information

if the course has left or right node

print as prerequisite information of that course

}