Nicholas Wyrwas

CS-300-15095-M01

Mr. Nathan Lebel

12/07/2024

**Old and Updated Pseudocode**

-updated pseudocode at bottom of word document

**Old Pseudocode:**

**Vector:**

* Load Courses from File
  + OPEN the file for reading.
  + FOR EACH line in file:
    - SPLIT the line into courseNumber, courseTitle, and coursePrerequisites.
    - IF line has less than two parameters, print an error: "Error: Missing data in line".
    - IF coursePrerequisites exist:
      * FOR EACH prerequisite in the list:
        + IF prerequisite is not found in courseMap:

Print an error: "Error: Prerequisite course not found".

* + - CREATE a Course object with the courseNumber and courseTitle.
    - IF coursePrerequisites exist:
      * ASSIGN coursePrerequisites to the Course object.
    - ADD the Course object to the courses vector.
    - ADD the Course object to courseMap using courseNumber as the key.
  + CLOSE the file.

Creating Course Object

* Course Struct:
  + courseNumber (string)
  + courseTitle (string)
  + coursePrerequisites (list of strings)
* Constructor:
  + Initializes courseNumber, courseTitle, and coursePrerequisites with input data.
* Search Course by Number
  + PROMPT the user for a courseNumber.
  + SEARCH for the courseNumber in courseMap.
  + IF the course is found:
    - Print the course number and title: "Course Number: [courseNumber], Title: [courseTitle]".
    - FOR EACH prerequisite in course.coursePrerequisites:
      * Print: "Prerequisite: [prerequisite]".
  + IF the course is not found:
    - Print an error: "Error: Course not found".
* Error Handling
  + IF line has fewer than two parameters, print an error: "Error: Missing data in line".
  + IF prerequisite does not exist in courseMap, print an error: "Error: Prerequisite course not found".

**Hash Table:**

FUNCTION LoadDataFromFile(fileName):

OPEN the file with name fileName

IF file cannot be opened:

PRINT "Error: Unable to open file"

RETURN

WHILE not end of file:

READ line from file

IF line is empty:

CONTINUE

SPLIT line into tokens by delimiter (e.g., comma or space)

VALIDATE line format:

IF tokens.length < 2:

PRINT "Error: Line does not have enough parameters"

CONTINUE

ASSIGN first token to courseNumber

ASSIGN second token to courseTitle

ASSIGN remaining tokens to prerequisites (if any)

FOR each prerequisite in prerequisites:

IF prerequisite does not exist as a course in the file:

PRINT "Error: Prerequisite " + prerequisite + " is not a valid course"

CREATE a course object using courseNumber, courseTitle, and prerequisites

INSERT course object into the hash table

CLOSE the file

END FUNCTION

**Pseudocode to show how to create course objects and store them in the appropriate data structure.**

FUNCTION CreateCourse(courseNumber, courseTitle, prerequisites):

CREATE a course object

SET course.courseNumber = courseNumber

SET course.title = courseTitle

SET course.prerequisites = prerequisites

RETURN course

END FUNCTION

FUNCTION InsertCourseIntoHashTable(course):

COMPUTE hash key using course.courseNumber

IF hash table[hash key] is empty:

CREATE a new node

ASSIGN node.data = course

ASSIGN hash table[hash key] = node

ELSE:

TRAVERSE linked list at hash table[hash key] until end

ADD new node containing course to end of linked list

END FUNCTION

**Pseudocode that will print out course information and prerequisites**

FUNCTION PrintAllCourses():

PRINT "Course Information:"

FOR each index in hash table:

SET node = hash table[index]

WHILE node is not null:

PRINT "Course Number: " + node.data.courseNumber

PRINT "Course Title: " + node.data.title

PRINT "Prerequisites: "

IF node.data.prerequisites is empty:

PRINT "None"

ELSE:

FOR each prerequisite in node.data.prerequisites:

PRINT prerequisite

PRINT "------------------------"

node = node.next

END FUNCTION

**Tree Data:**

Function loadDataFromFile(filename):

Open the file using the filename

If the file does not open:

Print "Error: Could not open the file"

Return

Create an empty binary search tree (BST)

While there are more lines to read in the file:

Read a line from the file

Parse the line into course number, course title, and prerequisites

If there are less than two parameters on the line:

Print "Error: Line is not formatted correctly"

Skip to the next line

Create a new course object with the course number, course title, and prerequisites

Validate prerequisites:

For each prerequisite in the prerequisites list:

If prerequisite does not exist in the BST (not yet loaded):

Print "Error: Invalid prerequisite course ID"

Skip to the next line

Insert the course object into the BST (insert the course based on the course number)

Close the file

Function createCourse(courseNumber, courseTitle, prerequisites):

Create a new course object with the following:

- Set the course number

- Set the course title

- Set the prerequisites (list of course numbers)

Return the course object

Function insertCourseIntoBST(bst, course):

If bst is empty:

Set bst root to the new course

Else:

Start at the root node of the BST

While there is a node:

If the course number is less than the current node's course number:

Move to the left child node

Else if the course number is greater than the current node's course number:

Move to the right child node

Else:

If the course number matches, overwrite or update the course data (if needed)

Exit the loop

Insert the course object into the appropriate position in the BST

Function printCourseInformation(bst):

Perform an in-order traversal of the BST:

For each node in the traversal:

Print the course number, course title, and list of prerequisites

If the course has prerequisites:

For each prerequisite:

Print the prerequisite course number

Print a newline after printing each course

Function validatePrerequisites(course, bst):

For each prerequisite in course's list of prerequisites:

If the prerequisite is not found in the BST:

Return False (invalid prerequisite)

Return True (all prerequisites are valid)

**Updated Pseudocode:**

FUNCTION loadDataFromFile(filename):

OPEN the file using filename

IF file cannot be opened:

PRINT "Error: Could not open the file"

RETURN

CREATE an empty data structure (vector, hash table, or BST)

WHILE not end of file:

READ a line from the file

IF line is empty:

CONTINUE

SPLIT the line into courseNumber, courseTitle, and prerequisites

IF there are less than two parameters on the line:

PRINT "Error: Line is not formatted correctly"

CONTINUE

CREATE a new course object using courseNumber, courseTitle, and prerequisites

VALIDATE prerequisites:

FOR each prerequisite in prerequisites list:

IF prerequisite does not exist:

PRINT "Error: Invalid prerequisite course ID"

CONTINUE

INSERT the course object into the appropriate data structure (vector, hash table, or BST)

CLOSE the file

END FUNCTION

FUNCTION createCourse(courseNumber, courseTitle, prerequisites):

CREATE a new course object

SET course.courseNumber = courseNumber

SET course.title = courseTitle

SET course.prerequisites = prerequisites

RETURN the course object

END FUNCTION

FUNCTION insertIntoVector(course):

ADD course to the end of the vector

END FUNCTION

FUNCTION insertCourseIntoHashTable(hashTable, course):

COMPUTE the hash key using course.courseNumber

IF the key does not exist in hashTable:

INSERT course into hashTable at the computed key

ELSE:

ADD course to the linked list at the computed key

END FUNCTION

FUNCTION insertCourseIntoBST(bst, course):

IF bst is empty:

SET bst.root = course

ELSE:

START at the root node of the BST

WHILE node exists:

IF course.courseNumber < node.courseNumber:

MOVE to the left child node

ELSE IF course.courseNumber > node.courseNumber:

MOVE to the right child node

ELSE:

IF course.courseNumber matches node:

OVERWRITE existing course data (optional)

EXIT loop

INSERT course into the appropriate position

END FUNCTION

FUNCTION displayMenu():

PRINT "Select an option:"

PRINT "1. Load file data"

PRINT "2. Print courses in order"

PRINT "3. Print course information and prerequisites"

PRINT "9. Exit"

READ user input for option

IF option == 1:

CALL loadDataFromFile(filename)

ELSE IF option == 2:

CALL printCoursesInOrder()

ELSE IF option == 3:

CALL printCourseInformation()

ELSE IF option == 9:

PRINT "Exiting the program"

EXIT

ELSE:

PRINT "Invalid option, try again"

END FUNCTION

FUNCTION printCoursesInOrder():

PERFORM an in-order traversal of the data structure (vector, hash table, or BST)

PRINT the course number, course title, and prerequisites for each course

IF prerequisites exist:

FOR each prerequisite:

PRINT the prerequisite course number

PRINT a newline after each course

END FUNCTION

FUNCTION validatePrerequisites(course, dataStructure):

FOR each prerequisite in course.prerequisites:

IF prerequisite is not found in the data structure:

RETURN False (invalid prerequisite)

RETURN True (all prerequisites are valid)

END FUNCTION