Aleksandr Donovan

Professor Nathan Lebel

6/11/2025

Project One Milestone Three: pseudocode

Pseudocode to define how the program opens the file, reads the data from the file, parses each line, and checks for file format errors:

| FUNCTION loadCoursesFromFile(fileName)  OPEN fileName FOR READING AS file  IF file IS NOT opened THEN  OUTPUT "Error: Unable to open file."  RETURN empty tree  END IF  DECLARE courseTree AS empty binary search tree (BST)  DECLARE courseNumbers AS empty set // used to validate prerequisites  WHILE NOT end of file DO  READ line FROM file  SET tokens TO SPLIT(line, ',')  IF SIZE OF tokens < 2 THEN  OUTPUT "Error: Insufficient parameters in line: ", line  CONTINUE TO NEXT ITERATION  END IF  SET courseNumber TO tokens[0]  SET courseTitle TO tokens[1]  ADD courseNumber TO courseNumbers  DECLARE prerequisites AS empty vector  IF SIZE OF tokens > 2 THEN  FOR EACH token IN tokens FROM INDEX 2 TO END DO  ADD token TO prerequisites  END FOR  END IF  DECLARE newCourse AS Course object  SET newCourse.number TO courseNumber  SET newCourse.title TO courseTitle  SET newCourse.prerequisites TO prerequisites  INSERT newCourse INTO courseTree  END WHILE  // Validate that all listed prerequisites exist in the course list  FOR EACH course IN courseTree (in-order traversal) DO  FOR EACH prerequisite IN course.prerequisites DO  IF prerequisite NOT IN courseNumbers THEN  OUTPUT "Error: Invalid prerequisite ", prerequisite, " for course ", course.number  REMOVE course FROM tree OR MARK course AS invalid  END IF  END FOR  END FOR  CLOSE file  RETURN courseTree  END FUNCTION |
| --- |

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

| STRUCT Course  DECLARE number AS STRING // e.g., "CSCI300"  DECLARE title AS STRING // e.g., "Intro to Algorithms"  DECLARE prerequisites AS VECTOR OF STRING  DECLARE left AS Course OR null // Left child in BST  DECLARE right AS Course OR null // Right child in BST  END STRUCT  STRUCT BinarySearchTree  DECLARE root AS Course OR null // Entry point to the tree  END STRUCT  FUNCTION parseCourseLine(line)  SET tokens TO SPLIT(line, ',')  SET courseNumber TO tokens[0]  SET courseTitle TO tokens[1]  DECLARE prerequisites AS empty vector  IF SIZE OF tokens > 2 THEN  FOR EACH token IN tokens FROM INDEX 2 TO END DO  ADD token TO prerequisites  END FOR  END IF  DECLARE newCourse AS Course object  SET newCourse.number TO courseNumber  SET newCourse.title TO courseTitle  SET newCourse.prerequisites TO prerequisites  SET newCourse.left TO null  SET newCourse.right TO null  RETURN newCourse  END FUNCTION  FUNCTION insertCourse(tree, course)  IF tree.root IS null THEN  SET tree.root TO course  RETURN  END IF  SET current TO tree.root  WHILE TRUE DO  IF course.number < current.number THEN  IF current.left IS null THEN  SET current.left TO course  RETURN  ELSE  SET current TO current.left  END IF  ELSE  IF current.right IS null THEN  SET current.right TO course  RETURN  ELSE  SET current TO current.right  END IF  END IF  END WHILE  END FUNCTION |
| --- |

Pseudocode that will print out course information and prerequisites:

| FUNCTION searchCourse(tree, courseNumber)  SET current TO tree.root  WHILE current IS NOT null DO  IF courseNumber EQUALS current.number THEN  OUTPUT "Course Number: ", current.number  OUTPUT "Course Title: ", current.title  IF SIZE OF current.prerequisites EQUALS 0 THEN  OUTPUT "Prerequisites: None"  ELSE  OUTPUT "Prerequisites:"  FOR EACH prerequisite IN current.prerequisites DO  OUTPUT " - ", prerequisite  END FOR  END IF  RETURN  ELSE IF courseNumber < current.number THEN  SET current TO current.left  ELSE  SET current TO current.right  END IF  END WHILE  OUTPUT "Course ", courseNumber, " not found."  END FUNCTION |
| --- |