Austin Frey

Professor Ling

CS-300

November 27, 2022

Milestone Five Tree Data Structure Pseudocode

1. **File opening, reading, parsing, and formatting**

Function Parser

Pass in: file name (file), the data type (type), and the separator character (sep)

Initialize line variable for reading in data and Vector list

IF type is an eFILE

Create input file stream for file

IF file is open

WHILE input file stream does not have errors

Read line

IF line is not empty

Push line to list

END IF

END WHILE

Close file

IF file is empty

Throw error

END IF

Call: parseHeader() & parseContent()

ELSE Throw error

END IF ELSE

ELSE

Create input stream for user-entered data

WHILE there are lines to process

IF line is not empty

Push line to list

END IF

END WHILE

IF input size is 0

Throw error

END IF

Call: parseContent()

END IF ELSE

Pass out: Parser object

End function

Function parseHeader // Potentially redundant function

Pass in: None

Create stream for each line to be parsed

Create item variable for storing header names

Create course object called header to store header lines

WHILE Not at the end of the line

Add item to header course

END WHILE

Pass out: None

End function

Function parseContent

Pass in: data structure

Initialize vector iterator index to beginning of the file + 1 to skip the header

FOR iterator doesn’t equal file end, increment iterator

Initialize courseID,

FOR length of iterator’s line

IF less than two elements in line

Throw error

ELSE

Add each element to array for creating the course object

END IF ELSE

END FOR

Create course object and add it to data structure

END FOR

Pass out: None

End function

Function validateCourseInfo

Pass in: Array of courses, prerequisite course ID being tested

FOR each element in array

IF a course ID matches prerequisite

Return true

END IF

END FOR

Return false

Pass out: Boolean representing whether or not the course ID was found

End function

1. **Course objects and data structures**

Struct BinarySearchTree

Pass in: None

Initialize Node root and set root to null

Initialize int size and set to 0

Pass out: BinarySearchTree object

End struct

Struct Course

Pass in: Course ID, course title, and prerequisites

Initialize class variables (ID, title, prerequisites array, key,)

Pass out: Course object

End struct

Struct Node

Pass in: None OR course OR course, parent

Declare Course, Left Node, Right Node, Parent Node

IF Node(None) // Default constructor

SET Course to empty Course object

SET Left Node, Right Node, Parent Node to null

ELSE IF Node(Course) //Initialize with Course

SET Course to passed in Course

SET parent, left, & right nodes to null

ELSE IF Node(Course, Node) //Initialize with Course and parent

SET Course to passed in Course, parent to passed in parent

SET left & right to null

END IF, ELSE IF, ELSE IF

Pass out: Node object

End struct

*Note: If, else if, else if are actually constructors, but for simplicity’s sake they have been written as conditional statements.*

Function addCourse

Pass in: Current node, Course object

IF BinarySearchTree root is null

SET root to new node(Course)

ELSE

IF Course id is less than or equal to current node course id

IF current Node’s left pointer is null

SET current Node’s left pointer to new node(Course, Current node)

ELSE

Call: addCourse(current node’s left, course)

END IF ELSE

ELSE //Course id is greater than current node course id

IF current Node’s right pointer is null

SET current Node’s right pointer to new node(Course, current Node)

ELSE

Call: addCourse(current node’s right, course)

END IF ELSE

END IF ELSE

END IF ELSE

Pass out: None

End function

1. **Print course information & prerequisites**

Function inOrderPrintAll

Pass in: Node to start at

IF Node is null

RETURN

END IF

IF Node’s left child is not null

Call: inOrderPrintAll(Node’s left child)

END IF

Print course information

Call: inOrderPrintAll(Node’s right child)

End function