Data structures: Vector

struct Course

courseNumber

courseTitle

prerequisites

Initialize an empty vector 'courseVector'

openFile()

WHILE line in the file

Split the line using comma

CREATE newCourse

INPUT courseNumber = courseData[0]

INPUT courseTitle = courseData[1]

FOR EACH courseData from courseData

ADD courseData to newCourse.prerequisites

ADD newCourse TO courseVector

Close file

FOR EACH course in course

IF course.courseNumber == courseNumber THEN

OUTPUT course.number

OUTPUT course.title

IF course.prerequisites is null THEN

FOR EACH prerequisites in course.prerequisites

OUTPUT prerequisites

ELSE

OUTPUT "No prerequisites"

OUTPUT "Course not found"

**searchAndPrint(courseNumber,)**

**FOR each course in course**

**IF course.courseNumber == courseNumber THEN**

**OUTPUT course.number**

**OUTPUT course.title**

**IF course.prerequisites is null THEN**

**For each prerequisites in course.prerequisites**

**OUTPUT prerequisites**

**ELSE**

**OUTPUT "No prerequisites"**

**Return**

**OUTPUT "Course not found"**

Data structures: Hash Table

CREATE CreateCourse Objects

courseNumber

courseTitle

prerequisites

CREATE a hash table to store courses

courseHashTable

KEY: courseNumber

VALUE: Course OBJECT

openFile()

IF file open THEN

WHILE there is line in file

READ NEXT LINE FROM FILE

PARSE LINE INTO courseNumber, courseName, prerequisites

IF LINE FORMAT IS INVALID THEN

OUTPUT "Invalid line format"

IF prerequisites ARE NOT VALID THEN

OUTPUT "Invalid prerequisite"

CREATE NEW Course OBJECT

INPUT Course.courseNumber = courseNumber

INPUT Course.courseTitle = courseTitle

INPUT Course.prerequisites = prerequisites

ADD Course OBJECT TO courseHashTable WITH KEY courseNumber

Close file

ELSE

OUTPUT "Could not open file"

FUNCTION printCourseInformation(courseNumber)

ADD Course OBJECT IN courseHashTable WITH KEY courseNumber

IF Course OBJECT == Course OBJECT THEN

OUTPUT "Course Number: " + Course.courseNumber

OUTPUT "Course Title: " + Course.course.Title

OUTPUT "Prerequisites:"

IF Course.prerequisites == NULL THEN

FOR EACH prerequisite IN Course.prerequisites

OUTPUT prerequisite

ELSE

OUTPUT "None"

ELSE

OUTPUT "Course not found"

Data structures: Tree

struct Course

courseNumber

courseTitle

prerequisites

insertCourse(course, tree)

insert(tree, course)

tree = NULL

CALL openFile()

IF file open THEN

WHILE there is line in file

READ NEXT LINE FROM FILE

PARSE LINE INTO courseNumber, courseName, prerequisites

SPLIT line, courseNumber, courseTitle, prerequisites

IF line format IS INVALID THEN

OUTPUT "Invalid line format"

IF prerequisites ARE NOT VALID THEN

OUTPUT "Invalid prerequisite"

CREATE NEW Course OBJECT

INPUT Course.courseNumber = courseNumber

INPUT Course.courseTitle = courseTitle

INPUT Course.prerequisites = prerequisites

IF courseNumber is not valid format THEN

OUTPUT "Invalid course number format"

IF courseTitle is not valid format THEN

OUTPUT "Invalid course title format"

FOR EACH prerequisite IN prerequisites

IF (prerequisite is not found in the tree) THEN

OUTPUT "Invalid prerequisite"

INPUT tree = insertCourse(tree, newCourse)

Close file

ELSE

OUTPUT "Could not open file"

IF node IS NOT NULL THEN

OUTPUT node.data.courseNumber + " - " + node.data.title

IF node.data.prerequisites.length > 0 THEN

OUTPUT "Prerequisites: "

FOR EACH prerequisite IN node.data.prerequisites

OUTPUT prerequisite

printCourseInfo(node.left)

printCourseInfo(node.right)

printCourseInfo(tree.root)

Menu

INPUT dataStructure = NULL

userChoice = 0

WHILE userChoice != 9

OUTPUT "Course Menu:"

OUTPUT "1. Load data into data structure"

OUTPUT "2. Print ordered course list"

OUTPUT "3. Print course details"

OUTPUT "9. Exit"

INPUT userChoice

SWITCH userChoice

CASE 1:

loadData()

break

CASE 2:

IF dataLoaded THEN

printCourseList(dataStructure)

break

ELSE

OUTPUT "Error: Load data first"

break

CASE 3:

IF dataLoaded THEN

OUTPUT "Enter the course ID:"

INPUT courseId

printCourseDetails(dataStructure, courseId)

break

ELSE

OUTPUT "Error: Load data first"

break

CASE 9:

OUTPUT "Exiting program..."

break

DEFAULT:

OUTPUT "Invalid choice. Please try again."

break

loadData()

fileData = Read file

IF fileData is not empty THEN

dataStructure = file\_data into desired data structure

printCourseList(dataStructure)

orderedCourses = Sort courses in dataStructure alphanumerically

FOR EACH Course in orderedCourses

OUTPUT Course

printCourseDetails(dataStructure, courseId)

IF courseTitle in dataStructure THEN

OUTPUT "Course Title: " courseTitle

OUTPUT "Prerequisites: " prerequisites of courseTitle in dataStructure

ELSE

OUTPUT "Course not found."

Alphanumeric Order

Vector

SORT courseList by alphanumeric course number (ascending)

FOR EACH Course in courseList

OUTPUT course.courseNumber, course.courseTitle

HashTable

keys = []

FOR EACH key in courseHashTable

ADD key to keys

SORT keys by alphanumeric order

FOR EACH key in keys

PRINT courseHashTable[key].courseNumber, courseHashTable[key].courseTitle

Tree

FUNCTION inOrderTraversal(node)

IF node IS NOT NULL THEN

inOrderTraversal(node.left)

PRINT node.data.courseNumber, node.data.courseTitle

inOrderTraversal(node.right)

inOrderTraversal(csTree.root)

Vector

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **Initialize an empty vector 'courseVector'** | 1 | 1 | **1** |
| **openFile()** | 1 | 1 | n |
| **WHILE line in the file** | 1 | n | n |
| **Split the line using comma** | 1 | n | n |
| **CREATE newCourse** | 1 | n | n |
| **INPUT courseNumber = courseData[0]** | 1 | n | n |
| **INPUT courseTitle = courseData[1]** | 1 | n | n |
| **FOR EACH courseData from courseData** | 1 | n | n |
| **ADD courseData to newCourse.prerequisites** | 1 | n | n |
| **ADD newCourse TO courseVector** | 1 | n | n |
| **Close file** | 1 | 1 | 1 |
| **FOR EACH course in course** | 1 | n | n |
| **IF course.courseNumber == courseNumber THEN**  **OUTPUT course.number**  **OUTPUT course.title** | 3 | n | n |
| **IF course.prerequisites is null THEN** | 1 | n | n |
| **FOR EACH prerequisites in course.prerequisites** | 1 | 1 | 1 |
| **OUTPUT prerequisites** | 1 | n | n |
| **ELSE**  **OUTPUT "No prerequisites"** | 2 | n | n |
| **OUTPUT "Course not found"** | 1 | n | n |
| **Total Cost** | | | **21n+3** |
| **Runtime** | | | **O(n)** |

Hash Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **CREATE Course Objects**  **courseNumber**  **courseTitle**  **prerequisites** | **4** | **n** | **n** |
| **CREATE a hash table to store courses**  **courseHashTable**  **KEY: courseNumber**  **VALUE: Course OBJECT** | **3** | **n** | **n** |
| **openFile()** | **1** | **1** | **1** |
| **IF file open THEN**  **WHILE there is line in file** | **2** | **n** | **n** |
| **READ NEXT LINE FROM FILE PARSE LINE INTO courseNumber, courseName, prerequisites** | **2** | **n** | **n** |
| **IF LINE FORMAT IS INVALID THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid line format"** | **1** | **1** | **1** |
| **IF prerequisites ARE NOT VALID THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid prerequisite"** | **1** | **1** | **1** |
| **CREATE NEW Course OBJECT**  **INPUT Course.courseNumber = courseNumber**  **INPUT Course.courseTitle = courseTitle**  **INPUT Course.prerequisites = prerequisites** | **4** | **n** | **n** |
| **ADD Course OBJECT TO courseHashTable WITH KEY courseNumber** | **1** | **n** | **n** |
| **Close file**  **ELSE**  **OUTPUT "Could not open file"** | **3** | **n** | **n** |
| **Total Cost** | | | **24n+3** |
| **Runtime** | | | **0(n)** |

Tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **struct Course**  **courseNumber**  **courseTitle**  **prerequisites** | **4** | **n** | **n** |
| **insertCourse(course, tree)**  **insert(tree, course)** | **2** | **n** | **n** |
| **tree = NULL** | **1** | **1** | **1** |
| **openFile()** | **1** | **n** | **n** |
| **IF file open THEN** | **1** | **n** | **n** |
| **WHILE there is line in file**  **READ NEXT LINE FROM FILE**  **PARSE LINE INTO courseNumber, courseName, prerequisites** | **3** | **n** | **n** |
| **SPLIT line, courseNumber, courseTitle, prerequisites** | **1** | **n** | **n** |
| **IF line format IS INVALID THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid line format"** | **1** | **1** | **n** |
| **IF prerequisites ARE NOT VALID THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid prerequisite"** | **1** | **1** | **1** |
| **CREATE NEW Course OBJECT** | **1** |  | **n** |
| **INPUT Course.courseNumber = courseNumber** | **1** | **n** | **n** |
| **INPUT Course.courseTitle = courseTitle** | **1** | **n** | **n** |
| **INPUT Course.prerequisites = prerequisites** | **1** | **n** | **n** |
| **IF courseNumber is not valid format THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid course number format"** | **1** | **1** | **1** |
| **IF courseTitle is not valid format THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid course title format"** | **1** | **1** | **1** |
| **FOR EACH prerequisite IN prerequisites** | **1** | **n** | **n** |
| **IF (prerequisite is not found in the tree) THEN** | **1** | **n** | **n** |
| **OUTPUT "Invalid prerequisite"** | **1** | **1** | **1** |
| **INPUT tree = insertCourse(tree, newCourse)** | **1** | **1** | **1** |
| **Close file** | **1** | **1** | **1** |
| **ELSE**  **OUTPUT "Could not open file"** | **2** | **1** | **1** |
| **IF node IS NOT NULL THEN** | **1** | **1** | **1** |
| **OUTPUT node.data.courseNumber + " - " + node.data.title** | **1** | **n** | **n** |
| **IF node.data.prerequisites.length > 0 THEN** | **1** | **n** | **n** |
| **OUTPUT "Prerequisites: "** | **1** | **1** | **1** |
| **FOR EACH prerequisite IN node.data.prerequisites** | **1** | **n** | **n** |
| **OUTPUT prerequisite** | **1** | **1** | **1** |
| **Total Cost** | | | **38n+11** |
| **Runtime** | | | **0(n)** |

Vectors offer dynamic resizing, making them adaptable for changing data

sizes and ideal for scenarios with unknown data amounts or frequent element

additions/removals. They provide ease of use for managing collections efficiently and

flexible element management. However, they may have slightly slower index access

and memory overhead compared to arrays, especially with large datasets.

Hash tables offer fast lookup and retrieval, flexibly supporting various key

types and ensuring efficient insertion and deletion. They prioritize speed by trading

space, but collisions and space overhead can hinder performance. Hash tables lack

order and may degrade to linear time complexity, heavily relying on the hash function

for optimal performance.

Advantages of tree structures include efficient O(log n) searching and sorting

for large datasets, natural representation of hierarchies, dynamic growth/shrink

capabilities, simplified recursive algorithms, and balanced trees ensuring optimal

performance. Disadvantages include complex implementation for balanced trees,

performance degradation in unbalanced trees, memory overhead from node

connections, and the need for traversal and comparison for retrieval due to a lack of

inherent data ordering.

Since the vector structure has a lower total cost than the other two structures

and a faster run time, that is the structure I will be using. Given that it takes less to

execute to carry out the particular function.