Afahri Kerr

Professor David Ostrowski

Project One

4/09/2023

**Program PseudoCode**

//Structure to hold course data

struct Course {

string courseID

string courseTitle

vector<string> preReq //vector to hold prerequisites

Course() {

courseID = “”

courseTitle = “”

preReq = “”

}

}

Menu PseudoCode

ASSIGN userInput = 0

PRINT menu

While userInput not 9 {

GET userInput

Switch (userInput) {

case 1:

OPEN specified text file

LOOP until end of file

READ line as string

SPLIT string on a comma

CREATE new Course object course

ASSIGN first part to Course object courseID property

ASSIGN second part to Course object courseTITLE property

ASSIGN remaining parts to Course object preReq property

ADD Course object to data structure type

SORT data structure

BREAK

case 2:

PRINT Courses in alphanumeric order (courseID, courseTitle)

BREAK

case 3:

GET courseID to search for

SEARCH data structure for courseID

IF found

PRINT Course information (courseID, courseTitle, preReq)

BREAK

ELSE

PRINT “Course Not Found”

BREAK

case 9:

PRINT “Thank you for using the course planner!”

BREAK

default:

PRINT “That is not a valid input!”

BREAK

}

}

**Vector Data Structure**

Creating course objects

CREATE Course class with ID, TITLE, and PREREQUISITES properties

CREATE empty Course vector courseList

OPEN specified text file

//Assign lines of input file to a Course Object

LOOP until end of file

READ line as string

SPLIT string on a comma

CREATE new Course object course

ASSIGN first part to course object ID property

ASSIGN second part to course object TITLE property

ASSIGN remaining parts to course object PREREQUISITES property

ADD Course object to courseList vector

SORT courseList using quick sort

Search for specific course

CREATE search function that takes courseNumber as input

GET courseNumber from user to search for

LOOP until end of vector or course found

SEARCH courseList vector

IF courseNumber is in courseList vector

PRINT Course information (ID, TITLE, PREREQUISITES)

STOP search

PRINT “Course not found”

**Hash Table Data Structure**

Open file, Create course objects, and Add it to data structure

CREATE a hashTable Class that has a vector of Course Nodes in it

CREATE a function to generate a key from courseId

CREATE a search function that takes courseNumber as input for hashTable Class

CREATE Course structure with ID, TITLE, and PREREQUISITES properties

OPEN specified text file

LOOP until end of file

READ line as string

SPLIT string on a comma

CREATE new Course object course

ASSIGN first part to course object ID property

ASSIGN second part to course object TITLE property

ASSIGN remaining parts to course object PREREQUISITES property

FIND key from course object ID

IF Course vector in hashTable at key position is empty

ADD Course object to CourseVector at key position

ELSE go to end of linked list at Course vector key position

ADD Course object

Print out course information and prerequisites

GET courseId from user to search for

FIND key from courseId

SEARCH Course Nodes vector in hashTable for key position

IF empty

PRINT "Course not found"

ELSE

WHILE not end of linked list at Course Nodes vector key position

IF courseNumber is in linkedList at Course Nodes vector key position

RETURN Course information (ID, TITLE, PREREQUISITES)

BREAK

ELSE

PRINT "Course not found"

**Binary Tree Data Structure**

Open file, Create course objects, and Add it to data structure

CREATE a Course structure to hold course ID, TITLE and PREREQUISITES

CREATE a Node structure to hold pointers and Course objects

CREATE a BinarySearchTree Class to hold Nodes

OPEN specified text file

LOOP until end of file

READ line as string

SPLIT string on a comma

CREATE new Course structure object

ASSIGN first part to course object ID property

ASSIGN second part to course object TITLE property

ASSIGN remaining parts to course object PREREQUISITES property

ADD Course object to a Node

ADD Node to BinarySearchTree

Print out course information and prerequisites

GET courseId from user to search for

SEARCH BinarySearchTree key for course object ID

IF empty

PRINT "Course not found"

ELSE

WHILE not end of BinarySearchTree

IF courseNumber is in BinarySearchTree

RETURN Course information (ID, TITLE, PREREQUISITES)

ELSE

PRINT "Course not found"

**Run Time Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Vector | Hash Table | Binary Tree |
| Load Data | O(1) | O(1)-O(N)  \*Depends on collisions | O(log N) |
| Search | O(n) | O(1) - O(N)  \*Depends on collisions | O(log N) - O(N)  \*depends on if data is already sorted when added to tree |
| Sort and Print | O(N log N)  \*using quick sort | O(N)  \* Table would have to be created in order or find all the keys to sort. | O(log N)  \*Should sort on addition of nodes  \*In order traversal |

**Advantages And Disadvantages**

The advantages for the vector class would be that it is easy to implement and since there are not a large number of classes, the program would not be much slower than the other data structures. It would be a disadvantage if there were a lot more classes.

The advantage for the Hash Table class is that it is very fast for searching large amounts of data but since there are not a lot of classes in this program, I don’t believe this is too much of an advantage. A disadvantage is that I don’t believe it can be sorted outright so we would have to get the keys of all the items and sort those.

The advantages of the Binary Tree are that it is very fast and the data is sorted as it is added to the tree. The disadvantage is that if the data is already sorted when added then the tree will be very large instead of uniform.

**Recommendation**

For this project I would recommend a Binary Search Tree data structure since it is fast to sort through and we need to be able to print out the data in alphanumeric order. Without the alphanumeric order requirement, I would have chosen a Hash Table.