To open and read file-

Function Load courses from the file

Open file with file name

If the file fails to open

Print “File could not be opened”

Then it Returns

Initialize empty list called course Lines

Initialize empty set called course Numbers

For each line in file

Split line by comma into token

Add check if tokens are less than 2

Print “Error invalid format”

Then Return

Add the first token to course Number

Add the line to course Lines

End

To build the course tree

For each line in course Lines

Split line by comma into tokens

If token not inside course Numbers

Print “Error, the Prerequisite was not found”

Then Return

End

Call Build Course Tree

End

Function Build course Tree

Initialize the empty Binary Search Tree called course Tree

For each line in course Lines

Split line by comma into tokens

Course Number to tokens

course Title To tokens

prerequisites to empty list

For each token

Add token to prerequisites list

Create new Course object using

Number Is course Number

Title Is course title

Prerequisites to prerequisites

Insert Course object into course Tree

Return to the course tree

Then end

Course subject

Class Course

String number

String title

List of String prerequisites

Constructor

Set this number to number

Set this title to title

Set this prerequisites To prerequisites

Print course information

Function Print Course

Course to course Tree Search and the course number

If course is null Then

Print “Course was not found”

Then Return

Print course number And course title

If course prerequisites is not empty then

Print “Prerequisites - “

For each of the prereq in course prerequisites

Prerequisites

Print “no prerequisites ”

End

Cs-300

Justine kummer

For filing input and errors

//use course not class

FUNCTION LoadCourseData

CREATE empty list course Lines

CREATE empty set

validCourseIDs

CREATE empty hash table courseTable

//if file cant be opened the show the error message

OPEN file for reading

If file cannot be opened then

PRINT “Error, File was not found.”

RETURN

//the error message for if file doesn’t open

For each line in file

//if line is empty then ignore it

If line Is empty Then

END

If the length of tokens less than 2 the

PRINT “Error: Line doesn’t have enough info.”

Then CONTINUE

END

course ID to tokens[0]

course Title to tokens[1]

prerequisites to tokens[2 to end]

Prerequisite to validCourseIDs

ADD (course ID, course Title, prerequisites) to course Lines

END

// Check if prerequisites exist, if not then use an error message

For each course Line IN course Lines

course ID to course Line[0]

prerequisites to courseLine[2]

//if prerequisite is not in the valid course it’s the use error message

For each prereq in prerequisites

If prereq not In validCourseIDs then

PRINT “Error, not defined”,

Then continue

END

// where to store course objects

For each course Line Inside course Lines

course ID to course Line[0]

course Title to course Line[1]

prerequisites to course Line[2]

//create A new course data by making a new object

Create the Course object course

Course. IdPrerequisi.ID

Course. Title to course Title

Course. Prerequisites to prerequisites

Put the course in course Table using the course ID as a key

END

RETURN course Table

END

//the object design for courses

STRUCT Course

STRING id

STRING title

List the prerequisites

end

//To print course info

Function

Print course info

Course to course Table[course ID]

//if course isn’t there use error message

If course is not there

PRINT Course was not found.

Return

End the program

//print the course id and title

PRINT course id and course title

//if prerequisites is not empty print them, if it is use error

If course. Prerequisites isnt empty then

Print Prerequisites:

For each of the prereq In the course Prerequisites

Print prerequisites

Or

//if prerequisites aren’t listed

PRINT Prerequisites: None

End

PRINT course. Id + “: “ + course. Title

If the course. Prerequisites isnt empty then

Print Prerequisites-

For each of the each prereq In the course prerequisites

Print prerequisites

Else

PRINT Prerequisites are not found

END

To open and read file-

Function load courses from the file

Open file with file name

If the file fails to open

Print File could not be opened

Then it returns

Initialize empty list called course lines

Initialize empty set called course numbers

For each line in file

Split line by comma into token

Add check if tokens are less than 2

Print Error invalid format

Then return

Add the first token to course number

Add the line to course lines

End

To build the course tree

For each line in course lines

Split line by comma into tokens

If token not inside course Numbers

Print Error, the Prerequisite was not found

The return

End

Call Build course tree

End

Function Build course Tree

Initialize the empty Binary Search Tree called course Tree

For each line in course Lines

Split line by comma into tokens

Course Number to tokens

Course Title To tokens

Prerequisites to empty list

For each token

Add token to prerequisites list

Create new Course object using

Number Is course Number

Title Is course title

Prerequisites to prerequisites

Insert Course object into course Tree

Return to the course tree

Then end

Course subject

Class Course

String number

String title

List of String prerequisites

Constructor

Set this number to number

Set this title to title

Set this prerequisites To prerequisites

Print course information

Function Print Course

Course to course Tree Search and the course number

If course is null Then

Print Course was not found

Then Return

Print course number And course title

If course prerequisites is not empty then

Print Prerequisites

For each of the prerequisites in course prerequisites

Prerequisites

Print no prerequisites

End

This is the Pseudocode to display the menu

Function Display Menu

Begin a Loop

Print Menu Options

Print 1 to Load course data from file

Print 2 to Print all the courses in alphanumeric order

Print 3 to Print course details and prerequisites

Print 9 to Exit program

The prompt the user to Enter the course

Then read the choice

If choice is 1 Then

Pull up Load Courses From the File

Print course load

If choice is 2 Then

Print all courses

Else If choice is 3 Then

Ask user to enter the course number to search

Use number to search

If choice is 9 Then

Print Ending program

End the loop and close the program

Or

Invalid input

Print please try again, invalid option

End

Vector

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | #number of times executed | Total cost |
| For all courses | 1 | N | N |
| If the course is the same | 1 | N | N |
| Print out the course info | 1 | 1 | 1 |
| For each prerequisite of course | 1 | # of prerequisites (k) | K |
| Print the prerequisite info | 1 | K | K |
| Total cost |  |  | 2n+2k+1 |
| Runtime |  |  | O(n+k) |

Binary

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | # of times executed | Total cost |
| Search course | O(logn) | 1 | O(logn) |
| If the course is not found | 1 | 1 | 1 |
| Print course info | 1 | 1 | 1 |
| For each course prerequisite | 1 | K | K |
| Print the prerequisites info | 1 | K | K |
| Total cost |  |  | O(logn) + 2k+2 |
| Runtime |  |  | O(logn+k) |

Hash table

|  |  |  |  |
| --- | --- | --- | --- |
| Code cost | Line cost | #of times executed | Total cost |
| Look up the course by the course # | (0)1 | 1 | (0)1 |
| If the course is not found | 1 | 1 | 1 |
| Print the course info | 1 | 1 | 1 |
| For each of the prerequisite of the course | 1 | K | K |
| Prerequisite info | 1 | K | K |
|  |  |  |  |
|  |  |  |  |
| Total cost |  |  | (0)1+2k+ 2 |
| Runtime |  |  | (0)k |

Evaluation

One of the factors it wants us to use is the big o analysis, using this we know the more courses, the bigger an impact it is. Since alphanumeric order is wanted, I’d recommend binary search tree. Since is already put in order its already factored into the program’s performance, this is an advantage. All the data structures have good and bad reasons to choose one over the other. With vector, order is not kept and done after. With hash tables it uses extra memory for extra information to search better. Hash tables are good for looking fast but don’t worry about the order. Vector is good for if you want a simpler structure but sorting is done after and may take longer. Binary search trees are more complex but are already in order saving time.