Pseudocode

Function read file

PASS IN: file path

Open stream to read file

IF file is not open

DISPLAY "File cant be read"

End Function

ENDIF

PASS OUT: contents of file

End Function

Function get hash table

PASS IN: file path

SET contents with read file passing in file path

SET array of contents by splitting by new line

INITIALIZE hash table

SET index i to 0

WHILE index i is less than size of array

SET details of array by splitting array by ","

INITIALIZE course

SET course.section to details[0]

SET course.name to details[1]

CALL insert function of hash table passing in course

SET index i to index i + 1

ENDWHILE

End Function

Function get Courses

PASS IN: file path

SET contents with read file passing in file path

SET array of contents by splitting by new line

INITIALIZE Courses

SET index i to 0

WHILE index i is less than size of array

SET details of array by splitting array by ","

IF size of details is greater or equal to 2

INITIALIZE Course

SET course.section to details[0]

SET course.name to details[1]

SET index j to 2

WHILE index j is less than size of details

ADD details[j] to course.prerequisites

SET index j to j + 1

ENDWHILE

ENDIF

SET index i to i + 1

ENDWHILE

End Function

Function print courses

PASS IN: Nothing

SET temp to head of courses

WHILE temp is not at the end

OUTPUT temp.section

OUTPUT temp.name

SET preq to head of course.prerequisites

WHILE preq is not at the end

OUTPUT preq

SET preq to preq.next

ENDWHILE

SET temp to temp.next

ENDWHILE

End Function

Function main

INITIALIZE course data

INITIALIZE course section

IF there is something in args

SET course section to args[0]

ELSE

SET course section to first course

ENDIF

SET continue to True

WHILE continue

SET answer to return of display menu()

SWITCH answer

CASE 1:

CALL get courses

END SWITCH

CASE 2:

CALL print courses

END SWITCH

CASE 3:

CALL print course passing in course section

END SWITCH

CASE 4:

SET continue to False

END SWITCH

default:

OUTPUT "please input correct number"

END SWITCH

END SWITCH

ENDWHILE

End Function

Function display menu

PASS IN: Nothing

OUTPUT "Menu"

OUTPUT "1: Load data"

OUTPUT "2: Print courses"

OUTPUT "3: Print Course"

OUTPUT "4: Exit"

INPUT user answer

PASS OUT: answer

End Function

Function sort by alphanumeric

PASS IN: current node

IF current node is Null

End Function

ELSE

sort by alphanumeric passing in left of current node

OUTPUT current node

sort by alphanumeric passing in right of current node

ENDIF

End Function

Evaluation

|  |  |
| --- | --- |
| Function | Runtime |
| Read File | O(n)  It will be reading the entire file |
| Get Hash Table | O(n)  Its adding each element to the table |
| Get Courses | O(n)  Its adding each element to the list.  Sometimes each element could have, in example, 4 prerequisites. This would make the notation at O(4n) which is still O(n). |
| Print Courses | O(n)  Its traversing through every element.  Same logic as get courses. |
| Sort by Alphanumeric | O(n)  Its traversing through every element in binary tree. |

|  |  |  |
| --- | --- | --- |
| Algorithm | Advantage | Disadvantage |
| Vector | Easiest to configure and maintain. | Slow in searching for element. Slow in sorting a list. |
| HashTable | Fastest in searching for a specific element. | Slow in sorting a list. |
| Binary Search Tree | Fastest in sorting a list by alphanumeric. | Not as fast as searching for an element as hash table. |

Based on the evaluation, I recommend using a Binary Search Tree structure. Even though it is not as fast as searching for an element as a hash table, it is still quite fast at O(log n). Sorting a list is also quite fast too at O(n), compared to the other structures worst case: O(n^2).