PseudoCode

Vector Data Structure

1.

Establish required fields

Establish vector

Function for loading courses ex(filename) {

open file in parentheses for reading

If file fail return 1;

loop file in parentheses

split lines into separate tokens labeled courseNumber, name, prerequisite

If prerequisite does not exist print courseNumber and name

Else not enough tokens to read the file return 1;

Pushback contents to vector

}

File close();

2.

Function for creatingCourseObjcets(file):

Establish empty vector

for each line in file that was previously read

loop and read through each established field in file

courseNum, title, prereq1, prereq2

add to newobject that stores fields as parameters each loop

return newobject

3.

Function for printingcourseinfo

Establish parameters for reading object and course number

if coursenumber matches inputted course

print course.number and course.title

print prerequisite if containing any

else

coursenumber not in database

print course not found

END

Hash Table Data Structure PsuedoCode

1.

Establish required fields

Establish hashtable

Function for loading courses ex(filename) {

open file in parentheses for reading

If file fail return 1;

loop file in parentheses

split lines into separate tokens labeled courseNumber, name, prerequisite

If prerequisite does not exist print courseNumber and name

Else not enough tokens to read the file return 1;

Add contents to table

}

File close();

2.

Function for creatingCourseObjcets(file):

Establish hashtable for storing objects keys(objects) and values

for each line in file that was previously read

loop and read through each established field in file

If file reads at least 2 lines

courseNum, title, prereq1, prereq2

Automatically instantiate newobject that stores inputs

Add newobject into hashtable storing course contents into hashtable

3.

Function for printingcourseinfo

Establish parameters for reading object and course number

For each course object

Print the coursenumber and title

Print the prequisities  
END

Tree Data Structure PseudoCode

1.

Establish required fields

Establish binarySearchTree and Nodes

Function for loading courses ex(filename) {

open file in parentheses for reading

If file fail return 1;

loop file in parentheses

split lines into separate tokens labeled courseNumber, name, prerequisite

If prerequisite does not exist print courseNumber and name

Else not enough tokens to read the file return 1;

Add contents to Set

}

File close();

2.

Function for creatingCourseObjcets(file):

Establish SearchTree and Nodes for storing objects keys(objects) and values

for each line in file that was previously read

loop and read through each established field in file

If file reads at least 2 lines

courseNum, title, prereq1, prereq2

Automatically instantiate newobject that stores inputs

Add newobject into tree storing course contents into tree

3.

Function printTree(node)

If node is not null then

Call printTree node->left

Print node course number

Print “Title:” , node.title

If node->prerequisites is not empty then

Print prerequisites

Else

Print “no prerequisites”

Endif

Call printTree(node->right)

Endif

End function

Call printTree(tree->root)

END

2.***Create PsuedoCode For A Menu:***

Option 1

Initialize timer variable set equal to clock function

Complete the function call for loadBids()

Input Arguments csvpath and data structure ex bidTable

Print calculation of elapsed time and display result

Option 2

Establish the function that you will use for printing the courses

Use the arrow operator to point the data structure to the function method

Option 3

Establish the function that will allow searching that prints course title and perquisites for specified individual course

Initialize timer variable that is set equal to clock function

If inputted argument does not match required field

Print field not found

Else

Print field with elapsed time and display results

Option 9

Initialize keyword break

Or

Print program "Exit” with return 1

*3.* ***Design pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order***

Vector

Assume that each course is stored in a structure

Initialize sorting method .sort() that sorts the specified list by alphanumeric order

For each course in courses

Print Course swith specified fields

HashTable

Define hash table with course numbers as keys and course names as values

Define function to sort and print courses from the hashtable

Sort list of tuples by course number or key (elements sorted lexicographically)

For coursename, coursekey in courselist

Print Courses with specified fields

Call function to sort and print courses

Tree

Initalize Tree and Tree Nodes

Define traversal function that traverse’s through trees left subtrees

Define traversal function that traverse’s through trees right subtrees

If node is not null

Traverse left sub tree with function that specifies the node

Print courses at current nodes

Next traverse right subtree that specifies the node

Print courses at the current nodes

|  |  |  |  |
| --- | --- | --- | --- |
| Vectors | Cost Per Line | Number Of Times Executed | Big O Value |
| Opening And Reading A File | 1 | O(n) | O(n) |
| Parsing And Creating Course Objects | 1 | O(n) | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Hash Table | Cost Per Line | Number Of Times Executed | Big O Value |
| Opening And Reading A File | 1 | O(n) | O(n) |
| Parsing And Creating Course Objects | O (1) | O(n) | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Binary Search Tree | Cost Per Line | Number Of Times Executed | Big O Value |
| Opening And Reading A File | 1 | O(n) | O(n) |
| Parsing And Creating Course Objects | O (log n) | O(n) | O(n log n) |

Vectors

Advantages:

Vectors can store large memory sequences

Vectors perform well when viewing sequential access of data element structures.

Disadvantages:

Accessing Vectors with many elements can be difficult to traverse

Traversing and finding elements can be slower compared to hash tables and trees.

Hash Table

Advantages:

Accessing keys and values tend to be one of the quickest algorithms out of the three structures

Hash Tables offer constant time for searching, insertion, and deletion

Disadvantages:

Requires high memory overhead due to array size and collision resolution method

When the load factor is too high or too low, difficulty in ordering or sorting elements based on keys or values, and vulnerability to hash-based attacks, such as denial-of-service or collision attacks

Hash Tables might be the most complex for implementation out of the three structures

Trees

Advantages:

Efficient insertion and deletion of data

Suitable for algorithms such as sorting

Disadvantages:

Searching operations tend to take longer

Requires additional memory for additional storage of pointers and children

Unbalanced trees can lead to poor performance

6. Hash Tables provide fast retrieval with runtime being O(1). For this particular assignment unique identifiers and hash tables seem suitable because it allows easy access to the course information which can be dynamically allocated and is flexible. Also hash table structure ensures no errors can be stumbled upon like collisions and open addressing methods.