Russell Wheeler

Professor Atkison

CS-300-T1160

08 October 2022

Project 1

Pseudocode

Vector

Start program

OpenFile(set two parameters)

{

initialize variables

open file

file = open file name

WHILE file not empty

line = getline(file)

parse each line

check file format for errors

IF the file format is free of errors,

check to see if prerequisites exist in course file

ELSE

return prompt stating error in file format

}

CreateCourseObject (set parameters)

{

initialize variables

open file

WHILE file is open

For each course object

store in Vector

}

SearchSpecificCourse ()

{

initialize variables

open file

WHILE file is open

print course information

store data in Vector

}

Hash Table

Start Program

Open file

read data

parse each line

check for course title

check for course number

IF prerequisites found

add to array

IF course parameters < two

skip course

display error msg: “File Improperly Formatted”

end program

ELSE

add course name, course number, and prerequisites to hash table

IF prerequisites exists

check if prerequisites come before the course

add to hash table

IF prerequisites not found

skip course

display error message

Create constructor CreateCourseObject with parameters

initialize variables

open file

read file

WHILE file is open

store the course obj in a hash table

Create constructor SearchforSpecificCourse with parameters

initialize variables

open file

read file

WHILE file is open

print course information

store data gathered in hash table

Binary Search Tree

Start Program

Open file

read data

parse each line

check for course title

check for course number

IF prerequisites found

add to Vector

IF course parameters < two

skip course

display error msg: “File Improperly Formatted”

end program

ELSE

add course name, course number, and prerequisites to Vector

IF prerequisites exist

check if prerequisites come before the course

add to Vector

IF prerequisites not found

skip course

display error message

Create course CreateCourseObject with parameters

initialize variables

open file

read file

WHILE file is open

store the course obj in a Vector

Create course SearchforSpecificCourse with parameters

initialize variables

open file

read file

WHILE file is open

print course information

store data gathered in Vector

Menu

Declare user\_Input

While user\_Input Not Equal to 9

Print “1. Load Data”

Print “2. Print Course List”

Print “3. Print Course ID”

Print “9. Exit”

Print “Please enter a number:”

Input = user\_Input

If user\_Input = 1

Read file

Store data in object

Else If user\_Input = 2

Print Courses

Else If user\_Input = 3

Print “Enter course ID”

Input = courseId

Find courseId

Print what is found

Else If user\_Input = 9

Print “Goodbye!”

Else

Print “Error wrong entry”

# Print alphabetical order:

Start Program

Open file

read data

parse each line

Store line in object

sort items to order in alphabetical order

Print all items

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Code | | Line Cost | | Vector | Hash table | Binary Search Tree | # Times Executes | Total Cost |
| OpenFile(set two parameters) | | 1 | | 1 | 1 | 1 | 1 | 1 |
| initialize variables | | 1 | | 1 | 1 | 1 | 1 | 1 |
| open file | | 1 | | 1 | 1 | 1 | 1 | 1 |
| file = open file name | | 1 | | 1 | 1 | 1 | 1 | 1 |
| WHILE file not empty | | 1 | | 1 | 1 | 1 | 1 | 1 |
| line = getline(file) | | n | | n | n | n | n | n |
| parse each line | | n | | n | n | n | n | n |
| check file format for errors | | n | | n | n | n | n | n |
| IF the file format is free of errors, | | n | | n | n | n | n | n |
| check to see if prerequisites exist in course file | | n | | n | n | n | n | n |
| ELSE | | n | | n | n | n | n | n |
| return prompt stating error in file format | | n | | n | n | n | n | n |
| CreateCourseObject (set parameters) | | 1 | | 1 | 1 | 1 | 1 | 1 |
| initialize variables | | 1 | | 1 | 1 | 1 | 1 | 1 |
| open file | | 1 | | 1 | 1 | 1 | 1 | 1 |
| WHILE file is open | | 1 | | 1 | 1 | 1 | 1 | 1 |
| For each course object | | n | | n | n | n | n | n |
| store in data type | | n | | n | n | n | n | n |
|  | | Datatype Sub  total | | 9 + 9n | 9 + 9n | 9 + 9n | 9 + 9n | 9 + 9n |
|  |  |  | Vector Total Cost | | | | | 9 + 9n |
|  |  |  | Hash Table Total Cost | | | | | 9 + 9n |
|  |  |  | Binary Search Tree Total Cost | | | | | 9 + 9n |

Advantages and Disadvantages

The Hash table is good at adding and removing data nodes as well as retrieving them. The biggest problem you may encounter with a hash table is if a collision occurs, this is when the program tries to write to the same location as another node.

The Binary Search Tree is good at searching; however, it really becomes inefficient when the list is sorted, this will lead the try to become one sided and the BST will lose its searching advantage.

The Vector data structure is a self-resizing array, however when the Vector grows beyond its initial size a new array is generated, and the old values are assigned to the new array and the old array will be freed. This could lead to spikes in greater run times when adding to large Vectors. This can be combated by doubling the capacity of the array every time it is resized.

Recommendation

The targeted employment of these functions is to help users search for and print out their necessary courses and the prerequisites to those courses. This assignment has requested that I analyze the loading task, however, I feel that the sort, search, and recovery tasks are more fitted to the application of these functions.

I would not recommend the Binary Search Tree because of the way that the course numbers are set up. The course numbers are set in numerical order. When the Binary Search Tree tries to perform search and recovery functions, there would essentially only be one long branch; an unbalanced tree.

The Hash Table is a viable choice; however, the main downfall of the hash table is the possibility of collisions, and the implementation of this table would need to avoid this. While the number of courses currently present are small as the courses grow in the future, it may lead to collisions and the initial programing may not scale as intended.

The largest benefit to using the Vector data type is that the courses will be organized; making the information more user friendly. The Vector data type is the preferential choice because it is more efficient when performing searches. In addition, when comparing the Vector and Binary Search Tree data type, they are almost identical when running complete traversals. There is one small drawback to using the Vector data type. That is that the files can take a bit longer to load. However, I would not consider this drawback to be detrimental when comparing the two because the file should only need to load-up one time, while traversals and searches will happen several times. For this reason, I recommend the Vector data type.