CS-300

Davit Mumladze

Project One

Professor Michael Rissover

June 13th, 2025

Common Course Structure

STRUCT Course

STRING courseId

STRING courseTitle

LIST<STRING> prerequisiteIds

END STRUCT

Vector-Based Design

Data Declaration

DECLARE VECTOR<Course> courseList

Load and Parse File

FUNCTION loadCourses(filePath)

OPEN file at filePath

IF file failed to open THEN

PRINT "Error: Cannot open file"

RETURN

END IF

WHILE NOT end of file

READ currentLine FROM file

SPLIT currentLine BY comma INTO tokens

IF LENGTH(tokens) < 2 THEN

PRINT "Error: Invalid format"

CONTINUE

END IF

DECLARE Course course

SET course.courseId = tokens[0]

SET course.courseTitle = tokens[1]

FOR i FROM 2 TO LENGTH(tokens) - 1

ADD tokens[i] TO course.prerequisiteIds

END FOR

ADD course TO courseList

END WHILE

CLOSE file

CALL validatePrerequisites(courseList)

END FUNCTION

Validate Prerequisites

FUNCTION validatePrerequisites(courseList)

DECLARE SET<STRING> validCourseIds

FOR EACH course IN courseList

ADD course.courseId TO validCourseIds

END FOR

FOR EACH course IN courseList

FOR EACH prereqId IN course.prerequisiteIds

IF prereqId NOT IN validCourseIds THEN

PRINT "Error: Prerequisite " + prereqId + " not found in file"

END IF

END FOR

END FOR

END FUNCTION

Print All Courses

SORT courseList BY course.courseId

FOR EACH course IN courseList

PRINT course.courseId + ": " + course.courseTitle

Print Specific Course

FUNCTION printCourseDetails(courseId)

FOR EACH course IN courseList

IF course.courseId == courseId THEN

PRINT "Course Number: " + course.courseId

PRINT "Course Title: " + course.courseTitle

IF course.prerequisiteIds IS EMPTY THEN

PRINT "Prerequisites: None"

ELSE

PRINT "Prerequisites:"

FOR EACH prereqId IN course.prerequisiteIds

PRINT " - " + prereqId

END FOR

END IF

RETURN

END IF

END FOR

PRINT "Course not found."

END FUNCTION

Hash Table-Based Design

Data Declaration

DECLARE HASH\_TABLE<Course> courseMap

Load and Parse File

FUNCTION loadCourseData(fileName)

OPEN file

IF file does not open THEN

PRINT "Failed to open input file."

RETURN

END IF

FOR EACH line IN file

SPLIT line BY comma INTO fields

IF LENGTH(fields) < 2 THEN

PRINT "Line format error"

CONTINUE

END IF

SET courseId = fields[0]

SET courseTitle = fields[1]

CREATE LIST<STRING> prerequisites

FOR i FROM 2 TO LENGTH(fields) - 1

ADD fields[i] TO prerequisites

DECLARE Course course

SET course.courseId = courseId

SET course.courseTitle = courseTitle

SET course.prerequisiteIds = prerequisites

SET courseMap[courseId] = course

END FOR

CLOSE file

END FUNCTION

Validate Prerequisites

FUNCTION checkMissingPrerequisites(courseMap)

FOR EACH course IN courseMap

FOR EACH prereqId IN course.prerequisiteIds

IF prereqId NOT IN courseMap THEN

PRINT "Warning: prerequisite " + prereqId + " not found for course " + course.courseId

END IF

END FOR

END FOR

END FUNCTION

Print All Courses

GET allKeys FROM courseMap

SORT allKeys ALPHABETICALLY

FOR EACH key IN allKeys

PRINT courseMap[key].courseId + ": " + courseMap[key].courseTitle

Print Specific Course

FUNCTION showCourseDetails(courseMap, queryId)

IF queryId NOT IN courseMap THEN

PRINT "Course not found."

RETURN

END IF

SET course = courseMap[queryId]

PRINT "Course Number: " + course.courseId

PRINT "Course Title: " + course.courseTitle

IF course.prerequisiteIds IS EMPTY THEN

PRINT "Prerequisites: None"

ELSE

PRINT "Prerequisites:"

FOR EACH prereqId IN course.prerequisiteIds

PRINT prereqId

END FOR

END IF

END FUNCTION

Binary Search Tree-Based Design

Data Declaration

BST<Course> courseTree

LIST<STRING> allPrerequisiteIds

Load and Parse File

WHILE NOT EOF(courseFile)

READ currentLine

SPLIT currentLine BY comma INTO tokens

IF LENGTH(tokens) < 2 THEN CONTINUE

DECLARE Course course

course.courseId = tokens[0]

course.courseTitle = tokens[1]

FOR i FROM 2 TO LENGTH(tokens) - 1

ADD tokens[i] TO course.prerequisiteIds

ADD tokens[i] TO allPrerequisiteIds

courseTree.INSERT(course.courseId, course)

END WHILE

Validate Prerequisites

FOR EACH prereqId IN allPrerequisiteIds

IF NOT courseTree.CONTAINS\_KEY(prereqId) THEN

PRINT "Error: Prerequisite " + prereqId + " not found in catalog."

END FOR

Print All Courses

IN\_ORDER\_TRAVERSAL(courseTree)

PRINT course.courseId + ": " + course.courseTitle

Print Specific Course

INPUT queryId

SET course = courseTree.FIND(queryId)

IF course IS NULL THEN

PRINT "Course not found."

ELSE

PRINT course.courseId + ": " + course.courseTitle

IF course.prerequisiteIds IS EMPTY THEN

PRINT "Prerequisites: None"

ELSE

PRINT "Prerequisites:"

FOR EACH prereqId IN course.prerequisiteIds

PRINT prereqId

END FOR

END IF

**Runtime and Memory Analysis**

| Operation | Vector | Hash Table | Binary Search Tree |
| --- | --- | --- | --- |
| Load and Insert | O(n) | O(n) | O(n log n) avg, O(n²) worst |
| Lookup Course | O(n) | O(1) avg, O(n) worst | O(log n) avg, O(n) worst |
| Print All Courses | O(n log n) | O(k log k) | O(n) |
| Print One Course | O(n) | O(1) | O(log n) |
| Memory Usage | Low | Moderate | High |

Final Recommendation;

After analyzing the performance characteristics of each structure, the Hash Table is the most efficient and appropriate data structure for this application. It provides constant time average lookups, efficient insertions, and simple implementation. Sorting can be deferred until display time by ordering keys. Although a binary search tree offers in-order traversal for natural sorting, it requires additional memory and can degrade to linear performance without balancing. The vector is easiest to use but inefficient for lookups and requires extra processing to sort.

Thus, the Hash Table is the recommended choice for implementation in the next phase of the project.