4-3 Milestone: Hash Table Data Structure Pseudocode

David Vega

Southern New Hampshire University

CS-300 DSA: Analysis and Design

Professor David Ostrowski

July 24, 2022

#include appropriate libraries and headers

**REQUIRMENT 1: FILE INPUT**

DECLARE method of type void loadCourses(string csvPath, HashTable \*hashTable)

// passes in the file path and instance of hashTable

{

DECLARE filestream variable courseList //will be used in the filestream to run various methods on the passed in file.

ifstream courseList;

OPEN file

courseList.open(csvPath);

WHILE courseList is good //check for errors opening the file

{

DECLARE string line; // used to store data as filestream parses the file

INVOKE method getline(courseList, line, ','); // passes in filestream name, variable to store data and delimiter to let the parser know to stop at that line and store data.

PRINT line

}

}

**REQUIREMENT 2 - Course Object Pseudocode**

DEFINE struct to hold course data // data provided by csv file previously loaded

Struct Course

{

DECLARE variable courseID; //stores the course id and will be used as the key in hash table.

DECLARE variable title; // stores the name of the course

DECLARE variable preReq1; // stores the first prerequisite

DECLARE variable preReq2; // stores the second prerequisite

DECLARE variable preReqCount; // variable to store the number of prerequisites

DECLARE method Course() // method that initializes the preReqCount

{

INITIALIZE preReqCount equal to 0; //method stores # of prerequisites

}

};

DEFINE class HashTable //defines structure of a hash table

{

DEFINE private data members of class HashTable

PRIVATE:

DEFINE Struct Node to hold courses //a structure to hold courses

{

DECLARE Course courses;

DECLARE unsigned int key; // will be used store hashed courseID

DECLARE Node \*next; // pointer to the next node

// Default constructor

DEFINE Node() method

{

INITIALIZE “key” to UINT\_MAX; // stores unsigned int max value to key

INITIALIZE “next” to null pointer; // sets the next nodes pointer to nullity

}

// Define Node method that passes in a course as a parameter and uses single colon to inherit from Node()

DEFINE Node (Course aCourse) : Node ()

{

INITIALIZE “course” to aCourse;

}

// Define Node method with two parameters (course, aKey) passed in as parameters and uses single colon to inherit from Node (Course course)

DEFINE Node (Course course, unsigned int aKey) : Node(course)

{

INITIALIZE “key” to aKey;

}

};

DECLARE a vector of courses (uninitialized) // to store course data

DECLARE integer “tableSize” and SET to DEFAULT\_SIZE;

DECLARE integer method “hash” and with key as parameter;

DEFINE public data members

PUBLIC:

DECLARE method HashTable(); // method to define resizing of hash table

DECLARE method PRINTALL () // method will define how to print all course data

};

DEFINE method hash() that takes in key as a parameter // method hashes the passed in key and returns the hashedKey value.

{

DECLARE integer “hashedKey” ;

INITIALIZE “hashedKey” to EVALUATE expression key MODULO (%) tableSize;

RETURN hashedKey

}

**REQUIREMENT 3 - Print Course Information Pseudocode**

DEFINE method of type void HashTable::PrintAll() // defines class public data member to print courses

{

ITERATE through hash table

DEFINE FOR loop

DECLARE and INITIALIZE loop iterator “i” and SET to 0;

SET “i” LESS THAN size of the hash table;

INCREMENT “i” by one;

{

DECLARE pointer Node \*currNode

SET currNode equal to reference pointer courses.at(ith) position;

IF currNode key IS NOT equal to UINT\_MAX

{

PRINT index “i” << currNode courseID << currNode preReq1 << currNode preReq2

WHILE currNode next IS NOT null pointer

{

SET currNode to next node

PRINT currNode key << currNode courseID << currNode preReq1 << currNode preReq2

}

}

}

RETURN;

}