CS 232 - P1

Programmer Manual

By: Jeffrey Marron for B. Streller’s Spring 2016 CS232 class

Contents:

I. cs232\_p1\_client\_driver.cpp

II. table.t

I. - cs232\_p1\_client\_driver.cpp

The driver first initializes two table storage containers. Table t stores the transition table data parsed from the transition table input file while table a stores the action table data parsed from the action table input file. A temporary pair pTemp is initialized which will be used to keep track of the current coordinates on the transition and action tables. The program is operated through a switch case command system via user input.

T. Calls cclear to clear the input buffer and then t.clear in order to ensure the table is empty before attempting to input data into it. t.getFileName then asks the user to specify a file path for the transition table input file. The data is then parsed from that specified file into table t via t.getAtInfo.

A. Calls cclear to clear the input buffer and then a.clear in order to ensure the table is empty before attempting to input data into it. a.getFileName then asks the user to specify a file path for the action table input file. The data is then parsed from that specified file into table a via a.getAtInfo.

M. Calls the driver.cpp function displayMenu which displays the main menu to the user.

E. The data input from the user goes into char tState. This is then converted into a 0-4 coordinate int by processing its ascii value by modulo 97. This value is put into pTemp.second which keeps track of the row in the tables. t.lookup is called using pTemp as an input parameter which checks its value against the coordinates in the table. The transition coordinates are then returned to int nState. Ptemp’s row coordinates are then updated based on the date returned from the table. If the coordinates call for an alarm or unlock action, then appropriate actions and performed which signal the user.

II. - table.t

This file specifies the templated implementations of the table container class.

string Table<Key, T, J>::getFileName(ifstream &, string & inFilename)

Opens the ifstream inFile based on the input file name specified by the user.

void Table<Key, T, J>::getAtInfo()

Creates a temporary pair kvpair. It then scans the input file so that it omits empty lines and lines starting with the tab space (the column header A-E). It then parses the input file line by line using the getline into string str which it then parses into string stream iss. The first piece of data is skipped (row header) and then the data values are converted from characters into integer x by adding together their cumulative ascii values. The row is modified by counter int j which increments after each row is processed. The column is modified by counter int k which increments after each insertion. J, k, and x are then stored as the first, second, and third pair values respectively into kvpair which is then inserted into the table.

void Table<Key, T, J>::print()

Prints the table data into the console window. Used for debugging.

Table<Key, T, J>::Table(int(\*map)(Key k))

The table constructor fills the multidimensional table with empty pair values by using nested for loops.

void Table<Key, T, J>::clear()

Same functionality as the Table constructor however its used to modify an existing table so that all coordinates are filled with empty pair values.

bool Table<Key, T, J>::insert(Pair<Key, T, J> kvpair)

Takes a pair kvpair as input. Examines kvpair’s first and second values to determine its table coordinates. It then stores the pair in those coordinates accordingly. Otherwise, a message displays if there is an invalid input.

bool Table<Key, T, J>::remove(const Key aKey)

Sets all the data points at a specified coordinate on the table as NULL.

int Table<Key, T, J>::lookUp(Pair<Key, T, J> kvpair, int &temp)

Takes a pair kvpair as input. Examines kvpair’s first and second values to look up the proper table coordinates. It then compares the first pair value at those table coordinates to the first value of kvpair. If the coordinates match, it assigns int temp the value from the third piece of data from the specified table coordinates. It then runs through an if else statement which evaluates the temp value (the addition of the ascii conversions from the original table input) and assigns temp a new value which will become a new table coordinate once its returned to int main. If temp is assigned 302 then the alarm state has been reached. If temp is assigned 335 then the unlock state has been reached.