#pragma once

#ifndef SAMPLEHEAD

#define SAMPLEHEAD

#include <iostream>

#include <algorithm>

#include <string>

using namespace std;

const int FOUR = 4;

template <class T, class G>

class HASHIN

{

private:

struct node

{

T data1;

G data2;

node \*next;

};//end struct node

node \*H[FOUR];

public:

void clearHashTable()

{

for (int i = 0; i < FOUR; ++i)

{

H[i] = NULL;

}//endfor

}//end clearHashTable

int hashFun(int x)

{

return x % 4;

}//end hashFun

void pushHash(T x, G y)

{

int hashed = hashFun(x[0] + x[1] + x[2]);

node \*p = new node;

p->data1 = x;

p->data2 = y;

p->next = H[hashed];

H[hashed] = p;

}//end pushHash

void searchHash(T x)

{

int hashed = hashFun(x[0] + x[1] + x[2]);

node \*p = H[hashed];

while (p != NULL)

{

if (p->data1 == x) { break; }

p = p->next;

}//endwhile

if (p == NULL) { cout << "Your data does not exist. . .\n"; }

else { cout << p->data1 << "is found and informaion is " << p->data2 << endl; }

}//end searchHash

void displayHash()

{

node \*p;

for (int i = 0; i < FOUR; ++i)

{

p = H[i];

cout << "H[" << i << "]-> ";

while (p != NULL)

{

cout << "(" << p->data1 << ", " << p->data2 << ")->";

p = p->next;

}//endwhile

cout << "NULL\n";

}//endfor

}//end displayHash

};//end HASHIN

template <class S>

class theStack

{

private:

struct node

{

S info;

node \*next;

};//end struct node

node \*list;

public:

theStack() { list = NULL; }

void pushs(S x)

{

node \*p = new node;

p->info = x;

p->next = NULL;

if (list == NULL)

{

list = p;

}

else

{

p->next = list;

list = p;

}

}//endpushs

bool emptys()

{

return(list == NULL) ? true : false;

}

S pops()

{

S x;

node \*p = list;

x = p->info;

list = p->next;

delete(p);

return x;

}

};//end class

template<class Q>

class theQueue

{

private:

struct node

{

Q info;

node \*next;

};

node \*front, \*rear;

public:

theQueue() { front = NULL; rear = NULL; }

void pushq(Q x)

{

node \*p = new node;

p->info = x;

p->next = NULL;

if (front == NULL)

{

front = p;

rear = p;

}//endif

else

{

rear->next = p;

rear = p;

}//endelse

}//end psuhq

bool emptyq()

{

return(front == NULL) ? true : false;

}

Q popQ()

{

Q x;

node \*p = front;

x = front->info;

front = p->next;

delete(p);

return x;

}//qnd popQ

void makeCircular()

{

rear->next = front;

}//end makecircular

int total = 0;

void displayCircular()

{

node \*p = front;

do {

total++;

cout << p->info << "->";

p = p->next;

} while (p != front);

cout << "RESTART->";

}

int giveTotal()

{

return total;

}

};

//------------------------------------------------

template <class A, class B, class C>

class BST

{

private:

struct node

{

A info1;

B info2;

C info3;

node \*left;

node \*right;

};//end struct node

node \*root;

public:

BST() { root = NULL; }

void insert(node \* &r, A x, B y, C z)

{

node \*p = new node;

p->info1 = x;

p->info2 = y;

p->info3 = z;

p->left = NULL;

p->right = NULL;

if (r == NULL)

{

r = p;

}//endif

else

{

if (x < r->info1) insert(r->left, x, y, z);

if (x > r->info1) insert(r->right, x, y, z);

}//endelse

}//end insert

void insert(A x, B y, C z)

{

insert(root, x, y, z);

}//end overridden insert

void displayInorder(node \*r)

{

if (r != NULL)

{

displayInorder(r->left);

cout << "ID:" << r->info1 << ", NAME:" << r->info2 << ", BAL:" << r->info3 << " ";

displayInorder(r->right);

}//endif

}//end displayInorder

void displayInorder()

{

displayInorder(root);

}//end overridden displayInorder

void displayPreorder(node \*r)

{

if (r != NULL)

{

cout << "ID:" << r->info1 << ", NAME:" << r->info2 << ", BAL:" << r->info3 << " ";

displayPreorder(r->left);

displayPreorder(r->right);

}//endif

}//end displayPreorder

void displayPreorder()

{

displayPreorder(root);

}//end overridden displayPreorder

void displayPostorder(node \*r)

{

if (r != NULL)

{

displayPostorder(r->left);

displayPostorder(r->right);

cout << "ID:" << r->info1 << ", NAME:" << r->info2 << ", BAL:" << r->info3 << " ";

}//endif

}//end displayPostorder

void displayPostorder()

{

displayPostorder(root);

}//end overridden displayPostorder

void search(node \*r, int x)

{

if (r == NULL) { cout << "Item was not found\n"; }

else if (r->info1 == x) { cout << "ID found with name:" << r->info2 << " and BALANCE of " << r->info3 << endl; }

else if (x < r->info1) { search(r->left, x); }

else if (x > r->info1) { search(r->right, x); }

}//end search

void search(int x)

{

search(root, x);

}//end overridden search

};//end class BST

#endif

#include "SAMPLEHEAD.h"

#include <fstream>

#include <ctime>

const int NUMOFMONTHS = 12;

int f(int n)

{

if (n == 1) return 3;

else return (n + f(n - 1));

}//end f

int main()

{

srand(time(0));

//----------------------------------------

int randomNumbers[10];

for (int i = 0; i < 10; ++i)

{

randomNumbers[i] = rand() % 50 + 1;

cout << randomNumbers[i] << " ";

}//endfor

cout << endl;

theStack <int> ss;

ss.pushs(11);

ss.pushs(22);

ss.pushs(33);

while (!ss.emptys())

{

int x = ss.pops();

cout << x << "->";

}

cout << "NULL\n";

cout << "Now use queue---------------------\n";

theQueue <int> qq;

qq.pushq(44);

qq.pushq(55);

qq.pushq(66);

while (!qq.emptyq())

{

int x = qq.popQ();

cout << x << "->";

}

cout << "NULL\n";

cout << "Now use circular---------------------\n";

qq.pushq(77);

qq.pushq(88);

qq.pushq(99);

qq.makeCircular();

qq.displayCircular();

cout << "total nodes = " << qq.giveTotal() << endl;;

while(!ss.emptys())

{

int x = ss.pops();

cout << x << ", ";

}

cout << endl;

cout << "----------------------------------------------------\n";

struct monthDays

{

string month;

int days;

};

cout << "calling statement 5 = " << f(5) << endl;

HASHIN <string, int> sampleHash;

sampleHash.clearHashTable();

monthDays theMonthDays[NUMOFMONTHS] = { {"JAN",31},{"FEB",28},{"MAR", 31},{"APR",30},{"MAY",31},{"JUN",30},

{"JUL",31},{"AUG",31},{"SEP",30},{"OCT",31},{"NOV",30},{"DEC",31} };

for (int i = 0; i < NUMOFMONTHS; ++i)

{

sampleHash.pushHash(theMonthDays[i].month, theMonthDays[i].days);

}//endfor

sampleHash.displayHash();

/\*

char again;

string toBeSearched;

do {

cout << "Search an item: ";

cin >> toBeSearched;

sampleHash.searchHash(toBeSearched);

cout << "Search again? (y/n): ";

cin >> again;

} while (again=='y' || again=='Y');

\*/

cout << "################################################\n";

//-----------------FOR THE BST-----------------

struct CUSTOMERS

{

int id;

string name;

double balance;

};//end CUSTOMERS

CUSTOMERS customer1;

BST <int, string, double> theTree;

CUSTOMERS sampleCustomers[3]; //= { {20, "John", 20.30},{10, "Mary", 10.10},{30, "Bill", 30.30} };

ifstream inData("mytext.txt");

if (inData.is\_open())

{

cout << "TEST:\n";

//cout << "your textfile does not exist\n";

for (int i = 0; i <3; ++i)

{

while (inData >> sampleCustomers[i].id >> sampleCustomers[i].name >> sampleCustomers[i].balance)

{

cout << sampleCustomers[i].id << ", " << sampleCustomers[i].name << ", " << sampleCustomers[i].balance << endl;

theTree.insert(sampleCustomers[i].id, sampleCustomers[i].name, sampleCustomers[i].balance);

}

}

}

else { cout << "your file does not exist. . .\n"; }

cout << "STREAM ABOVE---------------------\n";

/\*for (int i = 0; i < 3; ++i)

{

theTree.insert(sampleCustomers[i].id, sampleCustomers[i].name, sampleCustomers[i].balance);

}//endfor\*/

cout << "------------------------------\n";

theTree.displayInorder();

cout << endl;

theTree.displayPreorder();

cout << endl;

theTree.displayPostorder();

cout << endl;

int item;

char again = 'y';

do{

cout << "Search for an item: ";

cin >> item;

theTree.search(item);

cout << "Again?";

cin >> again;

} while (again == 'y' || again == 'Y');

system("pause");

return 0;

}//end main