//Name-Tasfique Enam

//Student ID- 5886429

//main class

#include <iostream>

#include <set>

#include <iterator>

#include <map>

#include <fstream>

#include <string>

#include <cstdlib>

#include <queue>

#include "SportFacility.h"

#include "Member.h"

using namespace std;

template <typename T, typename U>

void print\_set(multiset<T, U>& s)

{

for (const auto& elem : s)

{

cout << elem << endl;

}

}

template <typename K, typename V>

void print\_map(map <K, V> & m)

{

cout << "FACILITY BOOKING INFROMATION" << endl;

for (const auto& kv : m)

{

cout << "FACILITY ID : " << kv.first << endl << kv.second;

}

}

struct compareId

{

using is\_transparent = void;

bool operator()(SportFacility const& sf1, SportFacility const& sf2) const

{

return sf1.getId() < sf2.getId();

}

bool operator()(int id, SportFacility const& sf) const

{

return id < sf.getId();

}

bool operator()(SportFacility const& sf, int id) const

{

return sf.getId() < id;

}

};

int main()

{

multiset<SportFacility, compareId> sfList;

queue<Member> memberQueue;

map<int, Member> bookingMap;

int input;

string filename;

SportFacility facility; ///object

//int id, string desc, int capacity, string indicator, int duration, string startTime, double rentalFee

int fId=0;

string fDescription="";

int fCapacity=0;

string fOccupied="";

int fDuration=0;

string fStartTime="";

double fRentalFee=0.0;

bool check = false;

bool check2=false;

bool check3=false;

bool check4=false;

cout << "Enter the File Name: " << endl;

cin >> filename;

ifstream myFile(filename);

//myFile.open(filename.c\_str(), ios::in);

while(!myFile.good())

{

cout << "Invalid File, Try Again: " << endl;

cin >> filename;

myFile.open(filename.c\_str(), ios::in);

}

while(!myFile.eof())

{

myFile >> fId >> fDescription >> fCapacity >> fOccupied >> fDuration >> fStartTime >> fRentalFee;

/\*myFile >> fId;

myFile >> fDescription;

myFile >> fCapacity;

myFile >> fOccupied;

myFile >> fDuration;

myFile >> fStartTime;

myFile >> fRentalFee;\*/

facility.setId(fId);

facility.setDescription(fDescription);

facility.setCapacity(fCapacity);

facility.setIndicator(fOccupied);

facility.setDuration(fDuration);

facility.setStarttime(fStartTime);

facility.setRentalFee(fRentalFee);

sfList.insert(facility);

}

myFile.close();

cout << "File Read Successfully!" << endl << endl;

cout << "DISPLAY MULTISET " << endl;

print\_set(sfList);

/\*sfList.insert(SportFacility(1000, "BEDMINTON COURT", 30, "EMPTY", 12, "08:00 AM", 150.25));

sfList.insert(SportFacility(1001, "FUTSAL COURT", 30, "EMPTY", 12, "08:00 AM", 150.25));

sfList.insert(SportFacility(1002, "SWIMMING POOL", 100, "EMPTY", 14, "06:00 AM", 50.5));

cout << "DISPLAY MULTISET " << endl;

print\_set(sfList);\*/

do

{

cout << "Choose a menu, please." << endl;

cout << "1) MEMBER ARRIVAL" << endl;

cout << "2) BOOKING OF SPORT FACILITY" << endl;

cout << "3) LEAVING THE SPORT FACILITY" << endl;

cout << "4) DISPLAY ALL SPORT FACILITIES" << endl;

cout << "5) DISPLAY ALL BOOKING DETAILS" << endl;

cout << "6) EXIT THE PROGRAM" << endl;

cout << "> ";

cin >> input;

switch (input)

{

case 1: // MEMBER ARRIVAL

{

// INPUT MEMBER'S INFORMATION

string memberName;

int memberId;

cout << "Input a member name : ";

cin >> memberName;

cout << "Input a member ID : ";

//cin >> memberId;

while(!(cin >> memberId)) {

cout << "Error Please Try Again. "<< endl;

cin.clear();

cin.ignore(123, '\n');

}

// CREATE A MEMBER OBJECT

Member member(memberName, memberId);

// PUSH IT INTO THE QUEUE

memberQueue.push(member);

check = false;

check2 = true;

break;

}

case 2: // BOOKING OF SPORT FACILITY

{

if(check2==true) {

int searchValue;

cout << "Input a sport facility ID : ";

while(!(cin >> searchValue)) {

cout << "Error Please Try Again. "<< endl;

cin.clear();

cin.ignore(123, '\n');

}

//cout << "Input a sport facility ID : ";

//cin >> searchValue;

// SEARCH FOR THE SPORT FACILITY RECORD ACCORDING TO USER INPUT

// CHECK WHETHER THIS FACILITY IS OCCUPIED OR NOT

// IF IT IS EMPTY, THE INDICATOR IS CHANGED INTO 'OCCUPIED'

if (sfList.find(searchValue)->getIndicator().compare("EMPTY") == 0)

{

int id = sfList.find(searchValue)->getId();

string desc = sfList.find(searchValue)->getDesc();

int capacity = sfList.find(searchValue)->getCapacity();

int duration = sfList.find(searchValue)->getDuration();

string startTime = sfList.find(searchValue)->getStartTime();

double rentalFee = sfList.find(searchValue)->getRentalFee();

sfList.erase(sfList.find(searchValue));

sfList.insert(SportFacility(id, desc, capacity, "OCCUPIED", duration, startTime, rentalFee));

// AND REMOVE THE MEMBER FROM THE QUEUE AND ADD A NEW BOOKING RECORD TO MAP

Member insertMember = memberQueue.front();

memberQueue.pop();

bookingMap.insert(pair<int, Member>(id, insertMember));

cout << "BOOKING COMPLETE." << endl;

}

else

{

cout << "DATA IS NOT FOUND OR OCCUPIED" << endl;

}

} else{

cout << "Select Option 2 First. " << endl;

}

break;

}

case 3: // LEAVING THE SPORT FACILITY

{

if(check4==true) {

int searchValue;

cout << "Input a sport facility ID : ";

//cin >> searchValue;

while(!(cin >> searchValue)) {

cout << "Error Please Try Again. "<< endl;

cin.clear();

cin.ignore(123, '\n');

}

if (sfList.find(searchValue)->getIndicator().compare("OCCUPIED") == 0)

{

int id = sfList.find(searchValue)->getId();

string desc = sfList.find(searchValue)->getDesc();

int capacity = sfList.find(searchValue)->getCapacity();

int duration = sfList.find(searchValue)->getDuration();

string startTime = sfList.find(searchValue)->getStartTime();

double rentalFee = sfList.find(searchValue)->getRentalFee();

sfList.erase(sfList.find(searchValue));

sfList.insert(SportFacility(id, desc, capacity, "EMPTY", duration, startTime, rentalFee));

// AND REMOVE THE MEMBER FROM THE QUEUE AND ADD A NEW BOOKING RECORD TO MAP

bookingMap.erase(bookingMap.find(searchValue));

cout << "BOOKING COMPLETE." << endl;

}

else

{

cout << "DATA IS NOT FOUND OR OCCUPIED" << endl;

}

} else{

cout << "Select Option 5 First. " << endl;

}

break;

}

case 4: // DISPLAY ALL SPORT FACILITIES

{

print\_set(sfList);

break;

}

case 5: // DISPLAY ALL BOOKING DETAILS

{

print\_map(bookingMap);

check3 = false;

check4 = true;

break;

}

case 6: // EXIT THE PROGRAM

{

cout << "Thank you for using our program. End the program." << endl;

break;

}

default: // VALIDATE THE USER INPUT

{

cout << "Your input is wrong!" << endl;

}

}

} while (input != 6);

return 0;

}

#include "Member.h"

#include <iostream>

#include <set>

#include <iterator>

#include <map>

#include <fstream>

#include <string>

#include <cstdlib>

#include <queue>

using namespace std;

Member::Member()

{

name = "";

id = 0;

}

Member::Member(string name, int id) : name(name), id(id) {}

void Member::setAll(string name, int id)

{

this->name = name;

this->id = id;

}

string Member::getName()

{

return name;

}

int Member::getId()

{

return id;

}

ostream& operator<<(ostream& os, const Member& m)

{

os << "MEMBER INFORMATION" << endl << "MEMBER NAME : " << m.name << endl << "MEMBER ID : " << m.id << endl << endl;

return os;

}

#include "SportFacility.h"

#include <iostream>

#include <set>

#include <iterator>

#include <map>

#include <fstream>

#include <string>

#include <cstdlib>

#include <queue>

using namespace std;

SportFacility::SportFacility()

{

id = 0;

desc = "";

capacity = 0;

indicator = "EMPTY";

duration = 0;

startTime = "";

rentalFee = 0.0;

}

SportFacility::SportFacility(int id, string desc, int capacity, string indicator, int duration, string startTime, double rentalFee) : id(id), desc(desc), capacity(capacity), indicator(indicator), duration(duration), startTime(startTime), rentalFee(rentalFee) {}

//SportFacility(int, string, int, string, int, string, double);

void SportFacility::setId(int id)

{

this->id = id;

}

void SportFacility::setDescription(string desc)

{

this->desc = desc;

}

void SportFacility::setCapacity(int capacity)

{

this->capacity = capacity;

}

void SportFacility::setIndicator(string indicator)

{

this->indicator = indicator;

}

void SportFacility::setDuration(int duration)

{

this->duration = duration;

}

void SportFacility::setStarttime(string startTime)

{

this->startTime = startTime;

}

void SportFacility::setRentalFee(double rentalFee)

{

this->rentalFee = rentalFee;

}

int SportFacility::getId() const

{

return id;

}

string SportFacility::getDesc() const

{

return desc;

}

int SportFacility::getCapacity() const

{

return capacity;

}

string SportFacility::getIndicator() const

{

return indicator;

}

int SportFacility::getDuration() const

{

return duration;

}

string SportFacility::getStartTime() const

{

return startTime;

}

double SportFacility::getRentalFee() const

{

return rentalFee;

}

ostream& operator<<(ostream& os, const SportFacility& sf)

{

os << "FACILITY INFORMATION" << endl << "FACILITY ID : " << sf.id << endl << "FACILITY DESCRIPTION : " << sf.desc << endl << "FACILITY INDICATOR : " << sf.indicator << endl << "FACILITY DURATION : " << sf.duration << " hours" << endl << "FACILITY START TIME : " << sf.startTime << endl << "FACILITY RENTAL FEE : " << sf.rentalFee << endl << endl;

return os;

}