#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include "WorkDay.h"

#include "Employee.h"

using namespace std;

enum Week { sunday, monday, tuesday, wednesday, thursday, friday, saturday };

int main()

{

//This should not be in the other files I copied this from

//Seed the random number generator only once

unsigned seed = time(0);

srand(seed);

WorkDay sun(sunday);

WorkDay mon(monday);

WorkDay tue(tuesday);

WorkDay wed(wednesday);

WorkDay thu(thursday);

WorkDay fri(friday);

WorkDay sat(saturday);

//Set an array of all employees

const int SIZE = 10;

Employee employees[SIZE];

employees[0].setName("John Williams");

employees[1].setName("Carl Junior");

employees[2].setName("Matt Miller");

employees[3].setName("Amy Mack");

employees[4].setName("Paul Vargus");

employees[5].setName("Carol Watts");

employees[6].setName("Justin Stegus");

employees[7].setName("Brad Daniels");

employees[8].setName("Amanda Arnold");

employees[9].setName("Joe Donald");

sun.setEmployees(employees, SIZE);

mon.setEmployees(employees, SIZE);

tue.setEmployees(employees, SIZE);

wed.setEmployees(employees, SIZE);

thu.setEmployees(employees, SIZE);

fri.setEmployees(employees, SIZE);

sat.setEmployees(employees, SIZE);

//find way to return the data for each shift

sun.displayShift(4);

mon.displayShift(6);

tue.displayShift(6);

wed.displayShift(6);

thu.displayShift(6);

fri.displayShift(7);

sat.displayShift(7);

cout << "=================================================================\n";

for (int count = 0; count < SIZE; count++)

{

cout << employees[count].getName() << " has worked "

<< employees[count].getHoursToDate() << " hours this week.\n";

}

return 0;

}

#ifndef WORKDAY\_H

#define WORKDAY\_H

#include "Employee.h"

#include <string>

#include <iostream>

using namespace std;

struct Shift{

string name;

double shiftStart;

double shiftEnd;

};

class WorkDay

{

public:

WorkDay(int);

virtual ~WorkDay();

//void daySettings(int);

//void shiftArrSetter();

void setEmployees(Employee\*, int);

void displayShift(int);

string getDayName();

protected:

private:

int numShifts;

double dayStart;

double dayEnd;

Shift\* shiftArrPTR;

string dayName;

//Employee\* empArrPTR;

};

#endif // WORKDAY\_H

#include "WorkDay.h"

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <iomanip>

using namespace std;

enum Week { sunday, monday, tuesday, wednesday, thursday, friday, saturday };

WorkDay::WorkDay(int dayType)

{

switch(dayType)

{

case sunday: dayName = "Sunday";

break;

case monday: dayName = "Monday";

break;

case tuesday: dayName = "Tuesday";

break;

case wednesday: dayName = "Wednesday";

break;

case thursday: dayName = "Thursday";

break;

case friday: dayName = "Friday";

break;

case saturday: dayName = "Saturday";

break;

}

if (dayType == sunday)

{

dayStart = 10.00;

dayEnd = 18.00;

numShifts = 4;

shiftArrPTR = new Shift[4];

shiftArrPTR->shiftStart = 09.00;

shiftArrPTR->shiftEnd = 17.00;

(shiftArrPTR+1)->shiftStart = 09.00;

(shiftArrPTR+1)->shiftEnd = 17.00;

(shiftArrPTR+2)->shiftStart = 11.00;

(shiftArrPTR+2)->shiftEnd = 19.00;

(shiftArrPTR+3)->shiftStart = 11.00;

(shiftArrPTR+3)->shiftEnd = 19.00;

}

else if (dayType > sunday && dayType < friday)

{

dayStart = 10.00;

dayEnd = 21.50;

numShifts = 6;

shiftArrPTR = new Shift[6];

shiftArrPTR->shiftStart = 09.00;

shiftArrPTR->shiftEnd = 17.00;

(shiftArrPTR+1)->shiftStart = 09.00;

(shiftArrPTR+1)->shiftEnd = 17.00;

(shiftArrPTR+2)->shiftStart = 11.00;

(shiftArrPTR+2)->shiftEnd = 19.00;

(shiftArrPTR+3)->shiftStart = 13.00;

(shiftArrPTR+3)->shiftEnd = 21.00;

(shiftArrPTR+4)->shiftStart = 14.50;

(shiftArrPTR+4)->shiftEnd = 22.50;

(shiftArrPTR+5)->shiftStart = 14.50;

(shiftArrPTR+5)->shiftEnd = 22.50;

}

else if (dayType > thursday && dayType <= saturday)

{

dayStart = 10.00;

dayEnd = 21.50;

numShifts = 7;

shiftArrPTR = new Shift[7];

shiftArrPTR->shiftStart = 09.00;

shiftArrPTR->shiftEnd = 17.00;

(shiftArrPTR+1)->shiftStart = 09.00;

(shiftArrPTR+1)->shiftEnd = 17.00;

(shiftArrPTR+2)->shiftStart = 11.00;

(shiftArrPTR+2)->shiftEnd = 19.00;

(shiftArrPTR+3)->shiftStart = 12.00;

(shiftArrPTR+3)->shiftEnd = 20.00;

(shiftArrPTR+4)->shiftStart = 13.00;

(shiftArrPTR+4)->shiftEnd = 21.00;

(shiftArrPTR+5)->shiftStart = 14.50;

(shiftArrPTR+5)->shiftEnd = 22.50;

(shiftArrPTR+6)->shiftStart = 14.50;

(shiftArrPTR+6)->shiftEnd = 22.50;

}

else{

cout << "You dun' goofed. Try putting in valid"

<< " responses next time, retard.\n";

}

}

WorkDay::~WorkDay()

{

//dtor

}

void WorkDay::setEmployees(Employee\* emp, int numEmps)

{

//to keep a log of numbers already used

int\* logDynArr = new int[numShifts];

int randNum;

for (int count = 0; count < numShifts; count++)

{

do{

randNum = (rand() % (numEmps));

} while ((randNum == \*logDynArr || randNum == \*(logDynArr+1) || randNum == \*(logDynArr+2)

|| randNum == \*(logDynArr+3) || randNum == \*(logDynArr+4) || randNum == \*(logDynArr+5))

|| (emp+randNum)->getHoursToDate() >= 40);

(shiftArrPTR+count)->name = (emp+randNum)->getName();

\*(logDynArr+count) = randNum;

(emp+randNum)->addHours();

}

delete logDynArr;

logDynArr = NULL;

}

void WorkDay::displayShift(int shiftNum)

{

cout << "\n\tShift Schedule\n";

cout << "\t" << setw(10) << internal << getDayName() << endl;

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

{

cout << setw(14) << left << (shiftArrPTR+i)->name << ": ";

cout << (shiftArrPTR+i)->shiftStart << "-";

cout << (shiftArrPTR+i)->shiftEnd << endl;

}

}

string WorkDay::getDayName()

{

return dayName;

}

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <iostream>

#include <string>

using namespace std;

class Employee

{

public:

Employee();

Employee(int);

virtual ~Employee();

string getName();

void setName(string);

int getHoursMax();

void addHours();

int getHoursToDate();

protected:

private:

string name;

double hoursToDate;

int hoursMax;

static int overTimeEmps;

};

#endif // EMPLOYEE\_H

#include "Employee.h"

#include <string>

Employee::Employee()

{

hoursMax = 40;

hoursToDate = 0;

}

Employee::Employee(int hoursDesired)

{

hoursMax = hoursDesired;

}

Employee::~Employee()

{

//dtor

}

string Employee::getName()

{

return name;

}

void Employee::setName(string n)

{

name = n;

}

int Employee::getHoursMax()

{

return hoursMax;

}

void Employee::addHours()

{

hoursToDate += 8;

}

int Employee::getHoursToDate()

{

return hoursToDate;

}