Створити клас TPair, який представляє пару чисел і містить методи для їх збільшення/зменшення на 1. Реалізувати класи нащадки TTime (“години.хвилини”) та TMoney (“гривні.копійки”). Згенерувати поступово випадковим чином  пар (час, гроші), де час – тривалість виконання роботи, а гроші – вартість однієї хвилини роботи працівників. Обчислити витрати на виконання кожної із робіт

#pragma once

# include <string>

class Number

{

private:

int numberValue;

int maxNumberValue;

int minNumberValue;

public:

int get\_numberValue();

void set\_numberValue(int value);

int get\_maxNumberValue();

void set\_maxNumberValue(int value);

int get\_minNumberValue();

void set\_minNumberValue(int value);

Number(int numberValue, int maxNumberValue, int minNumberValue);

Number(int maxNumberValue, int minNumberValue);

Number(int numberValue);

Number();

bool increment();

bool decrement();

std::string toString();

};

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

#include "stdafx.h"

#include "Number.h"

int Number::get\_numberValue()

{

return numberValue;

}

void Number::set\_numberValue(int value)

{

if (value>=minNumberValue && value<maxNumberValue)

{

numberValue = value;

}

else

{

throw "error";

}

}

int Number::get\_maxNumberValue()

{

return maxNumberValue;

}

void Number::set\_maxNumberValue(int value)

{

if (value >= minNumberValue)

{

maxNumberValue = value;

}

else

throw "error";

}

int Number::get\_minNumberValue()

{

return minNumberValue;

}

void Number::set\_minNumberValue(int value)

{

minNumberValue = value;

}

Number::Number(int numberValue, int maxNumberValue, int minNumberValue)

{

set\_minNumberValue(minNumberValue);

set\_maxNumberValue(maxNumberValue);

set\_numberValue(numberValue);

}

Number::Number(int maxNumberValue, int minNumberValue):Number(minNumberValue,maxNumberValue,minNumberValue)

{

}

Number::Number(int numberValue):Number(numberValue,0,INT32\_MAX)

{

}

Number::Number():Number(0)

{

}

bool Number::increment()

{

numberValue++;

if (numberValue>maxNumberValue)

{

numberValue = minNumberValue;

return false; //Було порушення діапазону

}

return true; //Не було порушень діапазону

}

bool Number::decrement()

{

numberValue--;

if (numberValue<minNumberValue)

{

numberValue = maxNumberValue;

return false;

}

return true;

}

std::string Number::toString()

{

return std::to\_string(numberValue);

}

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

#pragma once

#include "Number.h"

#include <string>

class Pair

{

protected:

Number\* firstNumber;

Number\* secondNumber;

public:

Pair(int value1, int min1, int max1, int value2, int min2, int max2);

Pair(int min1, int max1, int min2, int max2);

Pair(int max1, int max2);

Pair();

void increment();

void decrement();

std::string toString(char delimeter);

int toNumber();

~Pair();

};

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

#include "stdafx.h"

#include "Pair.h"

Pair::Pair(int value1, int min1, int max1, int value2, int min2, int max2)

{

firstNumber = new Number(value1, max1, min1 );

secondNumber = new Number(value2, max2, min2);

}

Pair::Pair(int min1, int max1, int min2, int max2)

{

firstNumber = new Number(max1, min1);

secondNumber = new Number(max2, min2);

}

Pair::Pair(int max1, int max2)

{

firstNumber = new Number(max1, 0);

secondNumber = new Number(max2, 0);

}

Pair::Pair()

{

firstNumber = new Number();

secondNumber = new Number();

}

void Pair::increment()

{

if (firstNumber->increment()==false)

{

secondNumber->increment();

}

}

void Pair::decrement()

{

if (firstNumber->decrement()==false)

{

secondNumber->decrement();

}

}

std::string Pair::toString(char delimeter)

{

return secondNumber->toString()+delimeter+firstNumber->toString();

}

int Pair::toNumber()

{

return firstNumber->get\_numberValue()+

(secondNumber->get\_numberValue())\*((firstNumber->get\_maxNumberValue())+1);

}

Pair::~Pair()

{

delete firstNumber;

delete secondNumber;

}

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

#pragma once

#include "Pair.h"

#include <string>

class Time :

public Pair

{

public:

int get\_hours();

void set\_hours(int hours);

int get\_minutes();

void set\_minutes(int minutes);

Time(int hours, int minutes);

Time();

int toMinutes();

std::string toString();

~Time();

};

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

#include "stdafx.h"

#include "Time.h"

int Time::get\_hours()

{

return secondNumber->get\_numberValue();

}

void Time::set\_hours(int hours)

{

secondNumber->set\_numberValue(hours);

}

int Time::get\_minutes()

{

return firstNumber->get\_numberValue();

}

void Time::set\_minutes(int minutes)

{

firstNumber->set\_numberValue(minutes);

}

Time::Time(int hours, int minutes):Pair(minutes,0,59,hours,0,23)

{

}

Time::Time():Time(0,0)

{

}

int Time::toMinutes()

{

return toNumber();

}

std::string Time::toString()

{

return Pair::toString(':');

}

Time::~Time()

{

}

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

#pragma once

#include "Pair.h"

#include <string>

class Money :

public Pair

{

public:

int get\_uah();

void set\_uah(int uah);

int get\_coins();

void set\_coins(int coins);

Money(int uah, int coins);

Money(int coins);

Money();

int toCoins();

std::string toString();

~Money();

};

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

#include "stdafx.h"

#include "Money.h"

#include<stdlib.h>

int Money::get\_uah()

{

return secondNumber->get\_numberValue();

}

void Money::set\_uah(int uah)

{

secondNumber->set\_numberValue(uah);

}

int Money::get\_coins()

{

return firstNumber->get\_numberValue();

}

void Money::set\_coins(int coins)

{

firstNumber->set\_numberValue(coins);

}

Money::Money(int uah, int coins):Pair(coins,0,99,uah,0,INT32\_MAX)

{

}

Money::Money(int coins) : Pair(0, 99, 0, INT32\_MAX)

{

if (coins>=0)

{

set\_uah(coins / 100);

set\_coins(coins % 100);

}

else

{

throw "error";

}

}

Money::Money():Money(0,0)

{

}

int Money::toCoins()

{

return toNumber();

}

std::string Money::toString()

{

return Pair::toString('.');

}

Money::~Money()

{

}

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

#pragma once

#include "Money.h"

#include "Time.h"

#include <string>

using namespace std;

class Order

{

string title; //Назва роботи

Time\* timeOfOrder; //Тривалсть замовлення

Money\* minutePriceOfOrder; //Вартість однієї хвилини

public:

Order(string title, int h,int m,int uah, int coins);

Money\* get\_totalCost();

string toString();

~Order();

};

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

#include "stdafx.h"

#include "Order.h"

Order::Order(string title, int h, int m, int uah, int coins)

{

this->title = title;

timeOfOrder = new Time(h,m);

minutePriceOfOrder = new Money(uah,coins);

}

Money\* Order::get\_totalCost()

{

return new Money( timeOfOrder->toMinutes() \* minutePriceOfOrder->toCoins() );

}

string Order::toString()

{

return title +", time:"+timeOfOrder->toString()+", price:"+ minutePriceOfOrder->toString();

}

Order::~Order()

{

delete timeOfOrder;

delete minutePriceOfOrder;

}

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

// ConsoleApplication43.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "Order.h"

#include <iostream>

using namespace std;

int main()

{

Order\*\* orders = new Order\*[5];

//------------------ Випадковим чином генеруємо

for (int i = 0; i < 5; i++)

{

orders[i] = new Order("Order\_" + to\_string(i + 1), rand() % 24, rand() % 60, rand() % 200, rand() % 100);

}

//------------------ Виводимо вартість для кожного замовлення

for (int i = 0; i < 5; i++)

{

Money\* orderTotalCost = orders[i]->get\_totalCost();

printf("Order %d cost:%s\n", i + 1, orderTotalCost->toString().data());

delete orderTotalCost;

}

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

for (int i = 0; i < 5; i++)

{

delete orders[i];

}

delete[] orders;

system("pause");

return 0;

}

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

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