ЗВІТ

про виконання лабораторної роботи № 1.5

« Композиція класів та об’єктів »

з дисципліни

«Об’єктно-орієнтоване програмування»

студента групи ІТ-12

М’ягких Владислава Юрійовича

**Мета роботи:**

Освоїти використання композитних класів та об’єктів.

**Умова завдання:**

У всіх завданнях, крім вказаних в завданні операцій, обов’язково мають бути реалізовані наступні методи:

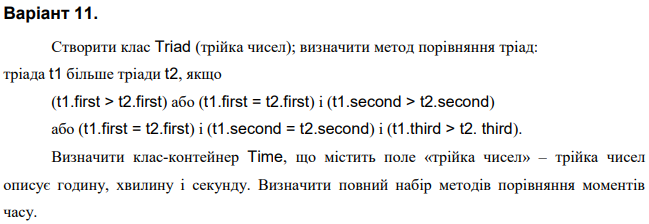
* методи доступу (константні методи зчитування та методи запису) значення кожного поля;
* метод ініціалізації Init( );
* метод введення з клавіатури Read( );
* метод виведення на екран Displау( );
* метод перетворення до літерного рядку toString( ).

Всі завдання мають бути реалізовані як класи із закритими полями, де операції реалізуються як методи класу.

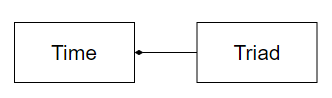
Визначення кожного класу та реалізацію його методів слід розмістити в окремих модулях.

Для демонстрації роботи з об’єктами нового типу у всіх завданнях потрібно написати головну функцію. У програмі обов’язково мають бути продемонстровані різні способи створення об’єктів і масивів об’єктів. Програма має демонструвати використання всіх функцій і методів.

У всіх завданнях потрібно реалізувати по два-три класи. Один клас є «контейнером», всі решту – описують поля, які містяться в «контейнері». Класи, що описують поля класу- «контейнера», мають бути визначені як незалежні.



**UML-діаграма класів:**



**Структурна схема програми:**

**Текст програми:**

///////////////

// Lab\_1.5.cpp

///////////////

#include <iostream>

#include "Time.h"

int main() {

Triad t1, t2;

t1.Read();

t2.Read();

t1.Display();

cout << t2.toString() << endl;

if (t1.h1(t2))

{

cout << "t1 < t2" << endl;

}

else if (t1.m1(t2))

{

cout << "t1 = t2" << endl;

}

else

{

cout << "t1 > t2" << endl;

}

Time d1, d2;

d1.Init(t1);

d2.Init(t2);

d1.Display();

cout << d2.toString() << endl;

if (d1.h1(d2))

{

cout << "d1 < d2" << endl;

}

else if (d1.m1(d2))

{

cout << "d1 = d2" << endl;

}

else

{

cout << "d1 > d2" << endl;

}

return 0;

}

///////////////

// Triad.h

///////////////

#pragma once

#include <string>

#include <sstream>

using namespace std;

class Triad

{

int first;

int second;

int third;

public:

int getFirst() const { return first; }

int getSecond() const { return second; }

int getThird() const { return third; }

bool setFirst(int);

bool setSecond(int);

bool setThird(int);

void Init(int firstValue, int secondValue, int thirdValue);

void Display() const;

void Read();

const string toString();

friend bool G(const Triad& t1, const Triad& t2);

bool h1(Triad t) const;

bool h2(Triad t) const;

bool m1(Triad t) const;

bool m2(Triad t) const;

bool s1(Triad t) const;

bool s2(Triad t) const;

double check();

};

///////////////

// Triad.cpp

///////////////

#include "Triad.h"

#include <iostream>

using namespace std;

bool Triad::setFirst(int value)

{

if (value > -1 && value < 24)

{

first = value;

return true;

}

else

{

return false;

}

}

bool Triad::setSecond(int value)

{

if (value > -1 && value < 60)

{

second = value;

return true;

}

else

{

return false;

}

}

bool Triad::setThird(int value)

{

if (value > -1 && value < 60)

{

third = value;

return true;

}

else

{

return false;

}

}

void Triad::Init(int firstValue, int secondValue, int thirdValue)

{

setFirst(firstValue);

setSecond(secondValue);

setThird(thirdValue);

}

void Triad::Display() const

{

cout << first << "." << second << "." << third << endl;

}

void Triad::Read()

{

int firstValue;

int secondValue;

int thirdValue;

cout << "Enter triad " << endl << endl;

cout << "first = "; cin >> firstValue;

cout << "second = "; cin >> secondValue;

cout << "third = "; cin >> thirdValue;

Init(firstValue, secondValue, thirdValue);

}

bool G(const Triad& t1, const Triad& t2)

{

return (t1.first > t2.first) || (t1.first == t2.first && t1.second > t2.second) || (t1.first == t2.first && t1.second == t2.second && t1.third > t2.third);

}

bool Triad::h1(Triad t) const

{

if (first < t.first)

{

return true;

}

if (first > t.first)

{

return false;

}

if (second < t.second)

{

return true;

}

if (second > t.second)

{

return false;

}

if (third < t.third)

{

return true;

}

if (third > t.third)

{

return false;

}

return false;

}

bool Triad::h2(Triad t) const

{

if (first < t.first)

{

return true;

}

if (first > t.first)

{

return false;

}

if (second < t.second)

{

return true;

}

if (second > t.second)

{

return false;

}

if (third < t.third)

{

return true;

}

if (third > t.third)

{

return false;

}

return true;

}

bool Triad::m1(Triad t) const

{

if ((first == t.first) && (second == t.second) && (third == t.third))

{

return true;

}

return false;

}

bool Triad::m2(Triad t) const

{

if ((first == t.first) && (second == t.second) && (third == t.third))

{

return false;

}

return true;

}

bool Triad::s1(Triad t) const

{

if (first < t.first)

{

return false;

}

if (first > t.first)

{

return true;

}

if (second < t.second)

{

return false;

}

if (second > t.second)

{

return true;

}

if (third < t.third)

{

return false;

}

if (third > t.third)

{

return true;

}

return false;

}

bool Triad::s2(Triad t) const

{

if (first < t.first)

{

return false;

}

if (first > t.first)

{

return true;

}

if (second < t.second)

{

return false;

}

if (second > t.second)

{

return true;

}

if (third < t.third)

{

return false;

}

if (third > t.third)

{

return true;

}

return true;

}

const string Triad::toString()

{

stringstream sout;

sout << first << "." << second << "." << third;

return sout.str();

}

double Triad::check()

{

return first > second;

}

///////////////

// Time.h

///////////////

#pragma once

#include <string>

#include "Triad.h"

using namespace std;

class Time {

Triad t;

public:

Triad getTriad() const { return t; }

void setTriad(Triad tValue) { t = tValue; }

void Init(Triad t);

void Display() const;

void Read();

const string toString();

bool h1(Time d) const;

bool h2(Time d) const;

bool m1(Time d) const;

bool m2(Time d) const;

bool s1(Time d) const;

bool s2(Time d) const;

};

///////////////

// Time.cpp

///////////////

#include "Time.h"

#include <iostream>

using namespace std;

void Time::Init(Triad dValue)

{

setTriad(dValue);

}

void Time::Display() const

{

t.Display();

}

void Time::Read()

{

Triad tValue;

tValue.Read();

Init(tValue);

}

bool Time::h1(Time d) const

{

return t.h1(d.getTriad());

}

bool Time::h2(Time d) const

{

return t.h2(d.getTriad());

}

bool Time::m1(Time d) const

{

return t.m1(d.getTriad());

}

bool Time::m2(Time d) const

{

return t.m2(d.getTriad());

}

bool Time::s1(Time d) const

{

return t.s1(d.getTriad());

}

bool Time::s2(Time d) const

{

return t.s2(d.getTriad());

}

const string Time::toString()

{

stringstream sout;

sout << t.toString();

return sout.str();

}

**Посилання на git-репозиторій з проектом:**

**Висновки:**

На даній лабораторній роботі, я освоїв використання композитних класів та об’єктів.