# **Звіт про виконання лабораторної роботи № 1.2**

## «Оголошення та будова класу» з дисципліни «Об’єктно-орієнтоване програмування»

## Студента групи «**ІТ-12**» - **Степанчука Сергія**

## Мета роботи

Освоїти використання класів та об’єктів

## Умова завдання

Потрібно реалізувати клас з приватними полями, вказаний в завданні свого варіанту.

Обов’язково потрібно реалізувати:

 методи доступу (константні методи зчитування та методи запису) значення кожного

поля;

 метод ініціалізації Init( ); метод має контролювати значення аргументів на

коректність;

 метод введення з клавіатури Read( );

 метод виведення на екран Display( ).

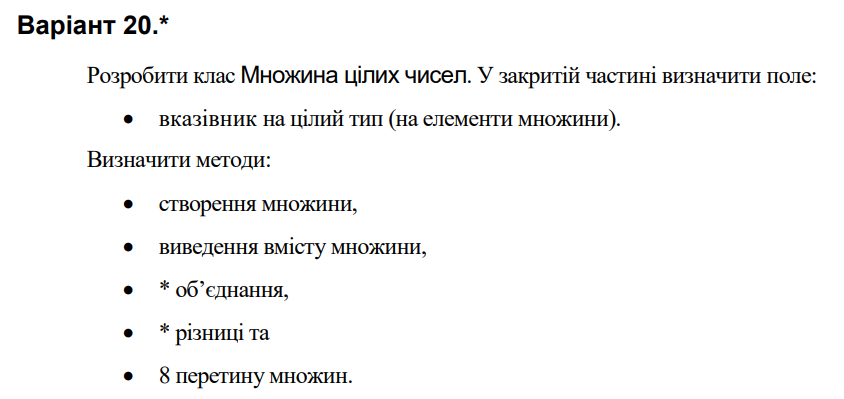
Реалізувати зовнішню функцію з ім’ям makeКлас( ), де Клас – ім’я класу, об’єкт

якого вона створює. Функція має отримувати як аргументи значення для полів класу і

повертати об’єкт необхідного класу. При передачі помилкових параметрів слід виводити

повідомлення і закінчувати роботу.

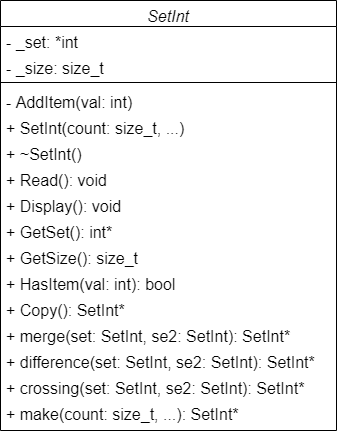
Визначення класу та реалізацію його методів слід розмістити в окремих модулях



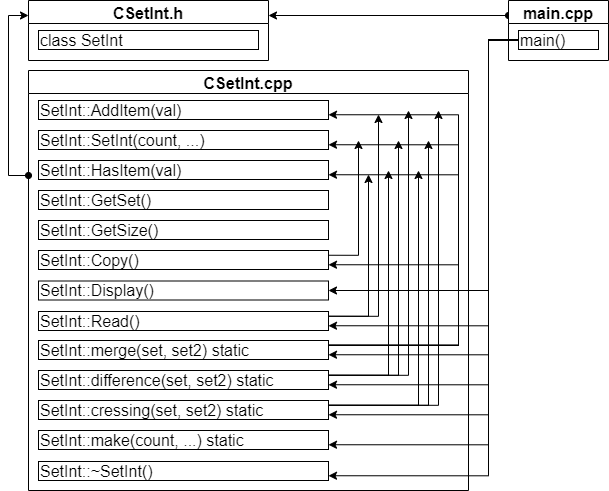
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_1.2>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // CSetInt.h

#pragma once

#include <iostream>

class SetInt

{

int\* \_set = nullptr;

size\_t \_size = 0;

void AddItem(const int& val);

public:

SetInt(size\_t count = 0, ...);

bool HasItem(const int& val) const;

const int \* GetSet() const;

size\_t GetSize() const;

SetInt\* Copy() const;

void Display() const;

void Read();

static SetInt\* merge(const SetInt& set, const SetInt& set2);

static SetInt\* difference(const SetInt& set, const SetInt& set2);

static SetInt\* crossing(const SetInt& set, const SetInt& set2);

static SetInt\* make(size\_t count = 0, ...);

~SetInt();

};

### // CSetInt.cpp

#include "CSetInt.h"

using namespace std;

void SetInt::AddItem(const int& val)

{

int\* tmp = new int[\_size+1];

if (\_size > 0)

memcpy(tmp, \_set, \_size\*sizeof(int));

\_set = tmp;

\_set[\_size] = val;

++\_size;

}

SetInt::SetInt(size\_t count, ...) {

if (count == 0)

return;

\_set = new int[\_size = count];

for (size\_t i = 0; i < \_size; i++)

\_set[i] = (&count + 1)[i];

}

bool SetInt::HasItem(const int& val) const

{

for (size\_t i = 0; i < \_size; i++)

if (val == \_set[i])

return true;

return false;

}

const int\* SetInt::GetSet() const

{

return \_set;

}

size\_t SetInt::GetSize() const

{

return \_size;

}

SetInt\* SetInt::Copy() const

{

SetInt \*tmp = new SetInt();

tmp->\_size = \_size;

tmp->\_set = new int[\_size];

memcpy(tmp->\_set, \_set, \_size\*sizeof(int));

return tmp;

}

void SetInt::Display() const

{

for (size\_t i = 0; i < \_size; i++)

cout << \_set[i] << ' ';

cout << endl;

}

void SetInt::Read()

{

int tmp;

char is\_cnt;

do

{

cout << "Input " << \_size + 1 << " item of set: ";

cin >> tmp;

if (!HasItem(tmp))

AddItem(tmp);

else

cout << "This num already in set!" << endl;

cout << "continue(y/n): ";

cin >> is\_cnt;

} while (is\_cnt != 'n');

}

SetInt\* SetInt::merge(const SetInt& set, const SetInt& set2)

{

SetInt \*tmp = set.Copy();

for (size\_t i = 0; i < set2.\_size; i++)

if (!tmp->HasItem(set2.\_set[i]))

tmp->AddItem(set2.\_set[i]);

return tmp;

}

SetInt\* SetInt::difference(const SetInt& set, const SetInt& set2)

{

SetInt \*tmp = new SetInt();

for (unsigned int i = 0; i < set.\_size; i++)

if (!set2.HasItem(set.\_set[i]))

tmp->AddItem(set.\_set[i]);

return tmp;

}

SetInt\* SetInt::crossing(const SetInt& set, const SetInt& set2)

{

SetInt \*tmp = new SetInt();

for (unsigned int i = 0; i < set.\_size; i++)

if (set2.HasItem(set.\_set[i]))

tmp->AddItem(set.\_set[i]);

return tmp;

}

SetInt\* SetInt::make(size\_t count, ...)

{

SetInt\* obj = new SetInt();

if (count == 0)

return obj;

obj->\_set = new int[obj->\_size = count--];

for (; count != UINT\_MAX; count--)

obj->\_set[count] = (&count + 1)[count];

return obj;

}

SetInt::~SetInt()

{

if (\_set != nullptr)

delete[] \_set;

}

### // main.cpp

#include "CSetInt.h"

using namespace std;

int main()

{

SetInt set[2];

for (size\_t i = 0; i < 2; i++)

{

cout << "Input " << i+1 << " set:" << endl;

set[i].Read();

set[i].Display();

}

/\*SetInt set[2] = {

SetInt(5, 1, 2, 3, 4, 5),

SetInt(3, 1, 2, 6)

};

for (size\_t i = 0; i < 2; i++)

{

set[i].Display();

cout << endl;

}\*/

SetInt \*merge = SetInt::merge(set[0], set[1]),

\*difference = SetInt::difference(set[0], set[1]),

\*crossing = SetInt::crossing(set[0], set[1]);

cout << "Merge (1, 2): " << endl;

merge->Display();

cout << endl;

cout << "Difference (1, 2): " << endl;

difference->Display();

cout << endl;

cout << "Crossing (1, 2): " << endl;

crossing->Display();

cout << endl;

delete merge, difference, crossing;

cout << endl << "Make func: " << endl;

SetInt\* mset = SetInt::make(2, 5, 10);

mset->Display();

delete mset;

return 0;

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Program/CSetInt.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(TestCSetInt)

{

public:

TEST\_METHOD(merge)

{

SetInt set[2] = {

SetInt(5, 1, 2, 3, 4, 5),

SetInt(3, 1, 2, 6)

};

SetInt \*merge = SetInt::Merge(set[0], set[1]);

const int out\_merge[] = { 1, 2, 3, 4, 5, 6 };

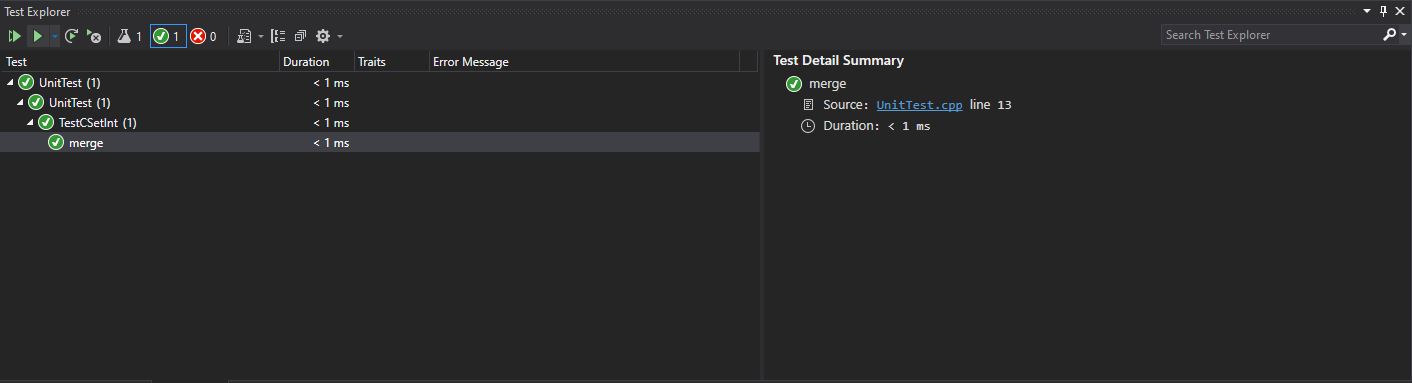
Assert::AreEqual(memcmp(out\_merge, merge->GetSet(), merge->GetSize() \* sizeof(int)), 0);

}

};

}

### Результат



## Висновок

Навчився використовувати класи.