# **Звіт про виконання лабораторної роботи № 4.5**

## «Освоїти використання віртуальних та абстрактних функцій та інтерфейсів» з дисципліни «Об’єктно-орієнтоване програмування»

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

Зміст

[**Звіт про виконання лабораторної роботи № 4.5** 0](#_Toc72661744)

[«Освоїти використання віртуальних та абстрактних функцій та інтерфейсів» з дисципліни «Об’єктно-орієнтоване програмування» 0](#_Toc72661745)

[Студента групи «**ІТ-12**» - **Степанчука Сергія** 0](#_Toc72661746)

[Мета роботи 2](#_Toc72661747)

[Умова завдання 2](#_Toc72661748)

[Посилання на git-репозиторій з проектом: 2](#_Toc72661749)

[UML-діаграма класів 3](#_Toc72661750)

[Структурна схема 4](#_Toc72661751)

[Текст програми 5](#_Toc72661752)

[// IArray.h 5](#_Toc72661753)

[// CArray.h 5](#_Toc72661754)

[// CArray.cpp 5](#_Toc72661755)

[// CAndArray.h 6](#_Toc72661756)

[// CAndArray.cpp 6](#_Toc72661757)

[// COrArray.h 7](#_Toc72661758)

[// COrArray.cpp 7](#_Toc72661759)

[// main.cpp 8](#_Toc72661760)

[UnitTest 9](#_Toc72661761)

[Код 9](#_Toc72661762)

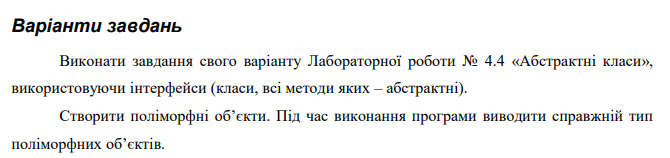
[Результат 9](#_Toc72661763)

[Висновок 9](#_Toc72661764)

## Мета роботи

Освоїти використання віртуальних та абстрактних функцій та інтерфейсів.

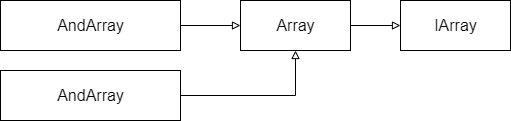
# Умова завдання



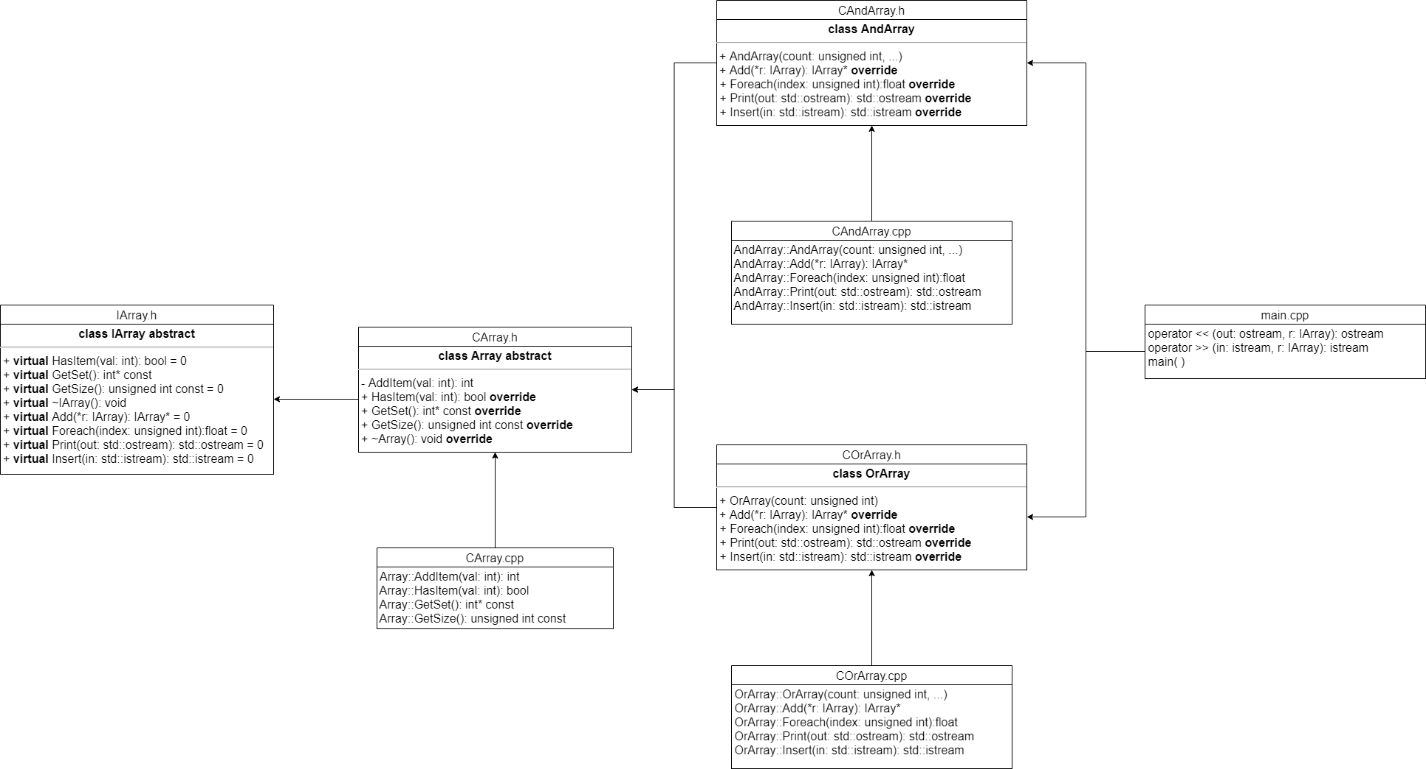
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_4.5>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // IArray.h

#pragma once

#include <iostream>

class IArray abstract

{

public:

virtual bool HasItem(const int& val) const = 0;

virtual const int\* GetSet() const = 0;

virtual unsigned int GetSize() const = 0;

virtual ~IArray() {};

virtual IArray\* Add(IArray\* r) = 0;

virtual float Foreach(const unsigned int& index) = 0;

virtual std::ostream& Print(std::ostream& out) const = 0;

virtual std::istream& Insert(std::istream& in) = 0;

};

### // CArray.h

#pragma once

#include "IArray.h"

class Array abstract : public IArray

{

protected:

int\* \_set = nullptr;

unsigned int \_size = 0;

void AddItem(const int& val);

public:

bool HasItem(const int& val) const override;

const int\* GetSet() const override;

unsigned int GetSize() const override;

~Array() override;

};

### // CArray.cpp

#include "CArray.h"

void Array::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;

}

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

{

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

if (val == \_set[i])

return true;

return false;

}

const int\* Array::GetSet() const

{

return \_set;

}

unsigned int Array::GetSize() const

{

return \_size;

}

Array::~Array()

{

if (\_set != nullptr)

delete[] \_set;

}

### // CAndArray.h

#pragma once

#include "CArray.h"

class AndArray: public Array

{

public:

AndArray(unsigned int count = 0, ...);

IArray\* Add(IArray\* r) override;

float Foreach(const unsigned int& index) override;

std::ostream& Print(std::ostream& out) const override;

std::istream& Insert(std::istream& in) override;

};

### // CAndArray.cpp

#include "CAndArray.h"

AndArray::AndArray(unsigned int count, ...) {

if (count == 0)

return;

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

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

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

}

IArray\* AndArray::Add(IArray\* r)

{

AndArray\* tmp = new AndArray();

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

if (r->HasItem(\_set[i]))

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

return tmp;

}

float AndArray::Foreach(const unsigned int& index)

{

return sqrt(\_set[index]);

}

std::ostream& AndArray::Print(std::ostream& out) const

{

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

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

out << std::endl;

return out;

}

std::istream& AndArray::Insert(std::istream& in)

{

int tmp;

char is\_cnt;

do

{

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

std::cin >> tmp;

if (!HasItem(tmp))

AddItem(tmp);

else

std::cout << "This num already in array!" << std::endl;

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

std::cin >> is\_cnt;

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

return in;

}

### // COrArray.h

#pragma once

#include "CArray.h"

class OrArray :public Array

{

public:

OrArray(unsigned int count = 0, ...);

IArray\* Add(IArray\* r) override;

float Foreach(const unsigned int& index) override;

std::ostream& Print(std::ostream& out) const override;

std::istream& Insert(std::istream& in) override;

};

### // COrArray.cpp

#include "COrArray.h"

OrArray::OrArray(unsigned int count, ...) {

if (count == 0)

return;

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

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

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

}

IArray\* OrArray::Add(IArray\* r)

{

OrArray\* tmp = new OrArray();

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

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

for (unsigned int i = 0; i < r->GetSize(); i++)

if (!tmp->HasItem(r->GetSet()[i]))

tmp->AddItem(r->GetSet()[i]);

return tmp;

}

float OrArray::Foreach(const unsigned int& index)

{

return log(\_set[index]);

}

std::ostream& OrArray::Print(std::ostream& out) const

{

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

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

out << std::endl;

return out;

}

std::istream& OrArray::Insert(std::istream& in)

{

int tmp;

char is\_cnt;

do

{

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

in >> tmp;

if (!HasItem(tmp))

AddItem(tmp);

else

std::cout << "This num already in array!" << std::endl;

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

in >> is\_cnt;

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

return in;

}

### // main.cpp

#include "CAndArray.h"

#include "COrArray.h"

using namespace std;

ostream& operator << (ostream& out, const IArray& r)

{

return r.Print(out);

}

istream& operator >> (istream& in, IArray& r)

{

return r.Insert(in);

}

void main()

{

cout << "AndArray" << endl;

{

IArray\* A = new AndArray();

IArray\* B = new AndArray();

cin >> \*A;

cout << "\*A = " << \*A << endl;

cin >> \*B;

cout << "\*B = " << \*B << endl;

IArray\* C = A->Add(B);

cout << "\*C = " << \*C << endl;

cout << "\*A[0](foreach): " << A->Foreach(0) << endl;

cout << typeid(A).name() << endl;

cout << typeid(\*A).name() << endl << endl;

delete A;

delete B;

delete C;

}

cout << "OrArray" << endl;

{

IArray\* A = new OrArray();

IArray\* B = new OrArray();

cin >> \*A;

cout << "\*A = " << \*A << endl;

cin >> \*B;

cout << "\*B = " << \*B << endl;

IArray\* C = A->Add(B);

cout << "\*C = " << \*C << endl;

cout << "\*A[0](foreach): " << A->Foreach(0) << endl;

cout << typeid(A).name() << endl;

cout << typeid(\*A).name() << endl << endl;

delete A;

delete B;

delete C;

}

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Program/IArray.h"

#include "../Program/CArray.h"

#include "../Program/CArray.cpp"

#include "../Program/CAndArray.h"

#include "../Program/CAndArray.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

TEST\_METHOD(TestMethod1)

{

Array \*arr = new AndArray(3, 1, 2, 3);

Assert::AreEqual(arr->GetSize(), (unsigned int)3);

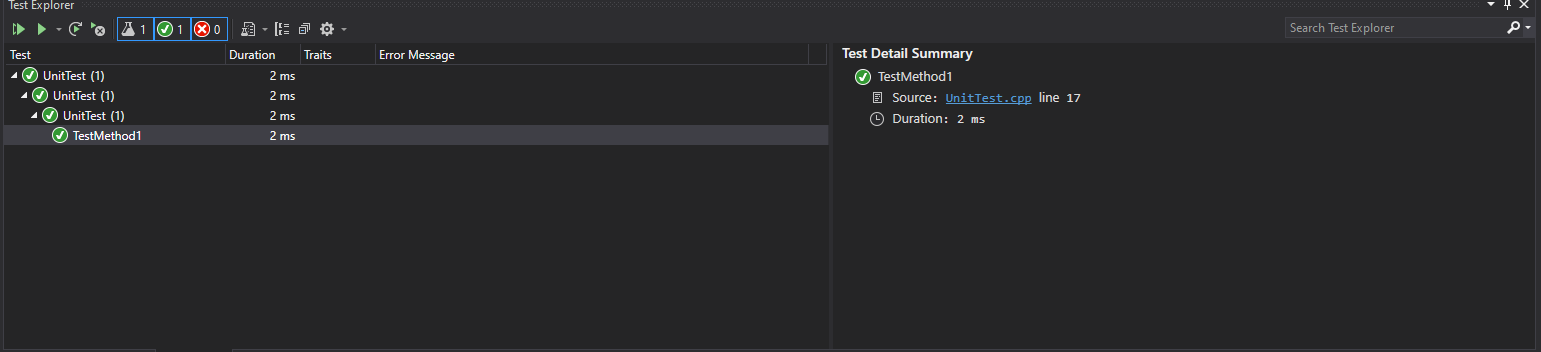
delete arr;

}

};

}

### Результат



## Висновок

Освоїв використання віртуальних та абстрактних функцій та інтерфейсів.