Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра ІУС

Дисципліна: «Технології об’єктно орієнтованого програмування»

**ПРАКТИЧНА РОБОТА № 4**

**«** **СПАДКУВАННЯ ТА ВІРТУАЛЬНИЙ ПОЛІМОРФІЗМ »**

|  |  |
| --- | --- |
|  | Прийняла:  Білова Т. Г.  з оцінкою «\_\_»  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2020р. |

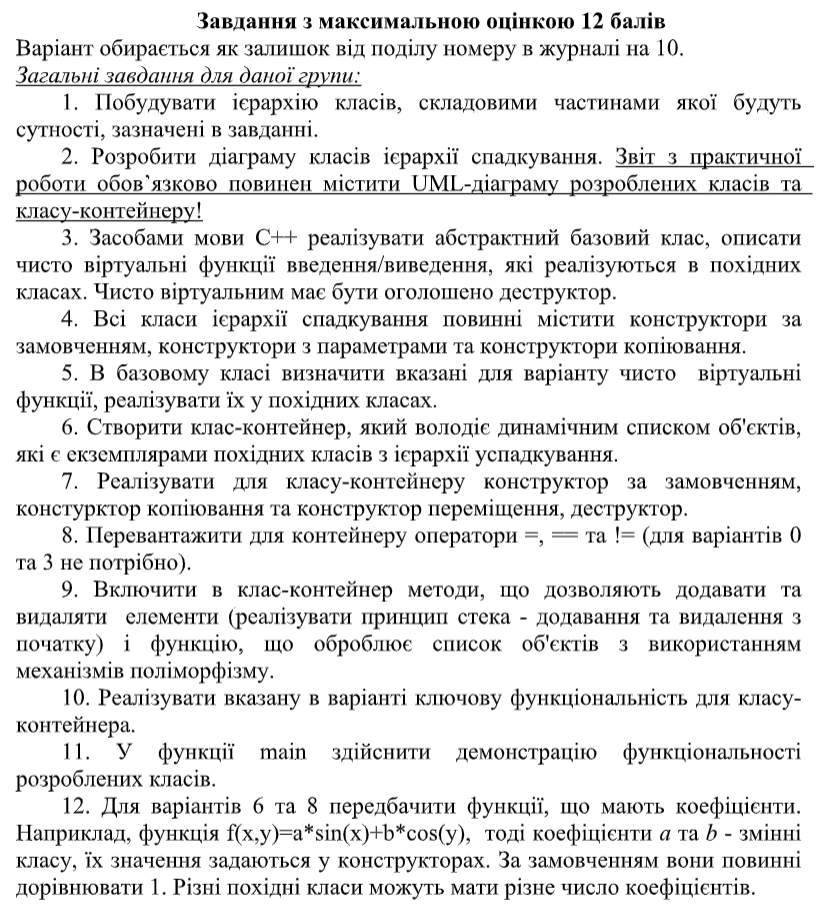
|  |  |
| --- | --- |
|  | Виконав:  Ст. гр. ІТУ-19-2  Куренков Б.М. |

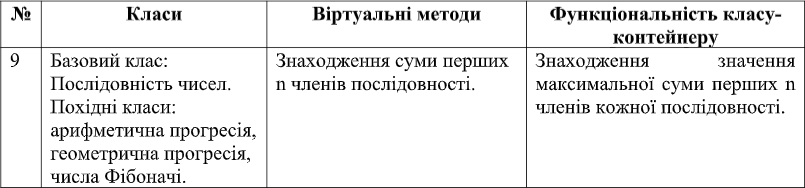
**МЕТА РОБОТИ**

Засвоєння поняття успадкування у класах, набуття навичок організації ієрархії успадкування. Засвоєння поняття поліморфізму, набуття навичок програмування віртуальних методів та використання поліморфних контейнерів.

**ПОСТАНОВКА ЗАДАЧІ**

**Варіант 9**





**ХІД РОБОТИ**

#pragma once

#include <iostream>

#include <cmath>

#include <utility>

class Num\_seq {

public:

virtual float Calculate(int) = 0;

virtual void setValue(float, float) = 0;

virtual float get\_a0() = 0;

virtual float get\_d() = 0;

virtual void Show(int) = 0;

virtual ~Num\_seq() { std::cout << "Num\_seq is deleted!\n\n"; }

};

class Arithmetic : public Num\_seq {

float a0;

float d;

public:

~Arithmetic() override { std::cout << "Arithmetic is deleted!\n"; };

virtual void Show(int n) override {

float result = a0, tmp = a0;

std::cout << "Arithmetic\nS" << n << " = " << result;

for (int i = 0; i < n; ++i) {

tmp += d;

result += tmp;

std::cout << " + " << tmp;

}

std::cout << " = " << result << "\na0 = " << a0 << " , d = " << d << " , n = " << n << "\n\n";

}

virtual float get\_a0() override { return a0; };

virtual float get\_d() override { return d; };

void setValue(float a0, float d) override { this->a0 = a0; this->d = d; }

Arithmetic() :a0(0), d(0) {}

Arithmetic(float a0, float d) { this->a0 = a0; this->d = d; }

Arithmetic(Arithmetic& num\_seq) {

a0 = num\_seq.a0;

d = num\_seq.d;

}

Arithmetic& operator=(Arithmetic& num\_seq) {

if (this == &num\_seq) return \*this;

this->a0 = num\_seq.a0;

this->d = num\_seq.d;

return \*this;

}

float Calculate(int n) override {

return a0 + (n +1- 1) \* d;

}

};

class Geometric : public Num\_seq {

float a0;

float d;

public:

~Geometric() override { std::cout << "Geometric is deleted!\n"; };

virtual void Show(int n) override {

float result = a0, tmp = a0;

std::cout << "Geometric\nS" << n << " = " << result;

for (int i = 0; i < n; ++i) {

tmp \*= d;

result += tmp;

std::cout << " + " << tmp;

}

std::cout << " = " << result << "\na0 = " << a0 << " , d = " << d << " , n = " << n << "\n\n";

}

virtual float get\_a0() override { return a0; };

virtual float get\_d() override { return d; };

void setValue(float a0, float d) override { this->a0 = a0; this->d = d; }

Geometric() : a0(0), d(0) {}

Geometric(float a0, float d) { this->a0 = a0; this->d = d; }

Geometric(Geometric& num\_seq) {

a0 = num\_seq.a0;

d = num\_seq.d;

}

Geometric& operator=(Geometric& num\_seq) {

if (this == &num\_seq) return \*this;

this->a0 = num\_seq.a0;

this->d = num\_seq.d;

return \*this;

}

float Calculate(int n) override {

return a0 \* pow(d, n + 1 - 1);

}

};

class Fibonacci : public Num\_seq {

float a0;

float d;

public:

~Fibonacci() override { std::cout << "Fibonacci is deleted!\n"; };

virtual void Show(int n) override {

float a1 = a0 + d, result = a0, tmp = a0, tmpp = result;

std::cout << "Fibonacci\nS" << n << " = " << result;

for (int i = 0; i < n; ++i) {

tmpp = tmp + a1;

result += tmpp;

tmp = a1;

a1 = tmpp;

std::cout << " + " << tmpp;

}

std::cout << " = " << result << "\na0 = " << a0 << " , d = " << d << " , n = " << n << "\n\n";

}

virtual float get\_a0() override { return a0; };

virtual float get\_d() override { return d; };

void setValue(float a0, float d) override { this->a0 = a0; this->d = d; }

Fibonacci() : a0(0), d(0) {}

Fibonacci(float a0, float d) { this->a0 = a0; this->d = d; }

Fibonacci(Fibonacci& num\_seq) {

this->a0 = num\_seq.a0;

this->d = num\_seq.d;

}

Fibonacci& operator=(Fibonacci& num\_seq) {

if (this == &num\_seq) return \*this;

this->a0 = num\_seq.a0;

this->d = num\_seq.d;

return \*this;

}

float Calculate(int n) override {

float result = a0, tmp = result, a1 = a0 + d;

for (int i = 0; i < n; ++i) {

result = tmp + a1;

tmp = a1;

a1 = result;

}

return result;

}

};

struct Element {

Num\_seq\* item;

Element\* next;

};

class Container {

Element\* head;

public:

bool operator==(Container& cont) {

if (head == NULL && cont.head == NULL) return 1;

if (head == cont.head) return 1;

if (typeid(\*this->head->item).name() != typeid(\*cont.head->item).name()) return 0;

if (this->head->item->get\_a0() == cont.head->item->get\_a0() && this->head->item->get\_d() == cont.head->item->get\_d()) return 1;

return 0;

}

bool operator!=(Container& cont) {

return !(\*this == cont);

}

Container& operator=(Container&& container) {

if (&container == this) return \*this;

if (!container.head) { head = NULL; }

else {

head = container.head;

container.head = NULL;

}

return \*this;

}

Container(Container&& container) {

if (!container.head) { head = NULL; }

else {

head = container.head;

container.head = NULL;

}

}

~Container() {

if (head) {

Element\* tmp = head;

while (tmp) {

delete[] tmp->item;

tmp = tmp->next;

}

}

head = NULL;

std::cout << "Container's content is deleted!\n\n";

}

Container() : head(NULL) {}

Container(Container& container) {

if (!container.head) { head->item = NULL; head->next = NULL; }

else {

this->~Container();

Element\* tmp\_cont = container.head;

Element\* tmp;

while (tmp\_cont) {

tmp = NULL;

tmp = new Element;

if (!strcmp(typeid(\*tmp\_cont->item).name(), "class Fibonacci"))

tmp->item = new Fibonacci;

else if (!strcmp(typeid(\*tmp\_cont->item).name(), "class Geometric"))

tmp->item = new Geometric;

else

tmp->item = new Arithmetic;

tmp->item->setValue(tmp\_cont->item->get\_a0(), tmp\_cont->item->get\_d());

tmp->next = head;

head = tmp;

tmp\_cont = tmp\_cont->next;

}

}

}

Container& operator=(Container& container) {

if (this == &container) return \*this;

if (!container.head) { head->item = NULL; head->next = NULL; }

else {

this->~Container();

Element\* tmp\_cont = container.head;

Element\* tmp;

while (tmp\_cont) {

tmp = NULL;

tmp = new Element;

if (!strcmp(typeid(\*tmp\_cont->item).name(), "class Fibonacci"))

tmp->item = new Fibonacci;

else if (!strcmp(typeid(\*tmp\_cont->item).name(), "class Geometric"))

tmp->item = new Geometric;

else

tmp->item = new Arithmetic;

tmp->item->setValue(tmp\_cont->item->get\_a0(), tmp\_cont->item->get\_d());

tmp->next = head;

head = tmp;

tmp\_cont = tmp\_cont->next;

}

}

return \*this;

}

void Push(Num\_seq& num\_seq) {

Element\* tmp = new Element;

if (!strcmp(typeid(num\_seq).name(), "class Fibonacci"))

tmp->item = new Fibonacci;

else if (!strcmp(typeid(num\_seq).name(), "class Geometric"))

tmp->item = new Geometric;

else

tmp->item = new Arithmetic;

tmp->item->setValue(num\_seq.get\_a0(), num\_seq.get\_d());

tmp->next = head;

head = tmp;

}

void Pop() {

if (head != NULL) {

Element\* tmp = head;

head = head->next;

tmp->item->~Num\_seq();

tmp = NULL;

}

}

void Show(int n) {

if (head == NULL) { std::cout << "Container is empty!\n\n"; }

else {

std::cout << "Container contain:\n\n";

Element\* tmp = head;

while (tmp != NULL) {

tmp->item->Show(n);

tmp = tmp->next;

}

std::cout << "\n";

}

}

void FindMaxValue(int n) {

if (!head) { std::cout << "Container is empty!\n\n"; }

else {

Element\* tmp = head;

std::cout << "List of values with n = " << n << ":\n";

for (int i = 0; tmp; ++i) {

std::cout << "container[" << i << "] = ";

std::cout << tmp->item->Calculate(n) << "\n";

tmp = tmp->next;

}

tmp = head;

std::cout << "\n";

float max\_i = 0, max\_value = tmp->item->Calculate(n);

for (int i = 0; tmp; i++) {

if (max\_value < tmp->item->Calculate(n)) {

max\_i = i + 1;

max\_value = tmp->item->Calculate(n);

}

tmp = tmp->next;

}

std::cout << "container[" << max\_i << "] has the biggest value = " << max\_value << "\n\n";

}

}

};

-----------------------------------------------------------------------------------------

#include "Numerical\_sequences.h"

int main() {

const int N = 3;

Container\* container = new Container;

for (int i = 0; i < N; i++) {

if (i % 3 == 2) {

container->Push(\*(new Arithmetic(1, 0.1)));

}

else if (i % 3 == 1) {

container->Push(\*(new Geometric(2, 0.1)));

}

else

container->Push(\*(new Fibonacci(3, 0.1)));

}

container->Show(5);

container->FindMaxValue(6);

Fibonacci\* arith1 = new Fibonacci;

container->Push(\*arith1);

container->Show(5);

container->FindMaxValue(6);

arith1->setValue(1, 0.1);

container->Show(5);

container->FindMaxValue(6);

Fibonacci\* arith2 = new Fibonacci;

\*arith2 = \*arith1;

arith2->Show(5);

std::cout << arith2->Calculate(3) << "\n\n";

arith1->~Fibonacci();

arith2->Show(5);

std::cout << arith2->Calculate(3) << "\n\n";

Container\* container1 = new Container;

Container\* container2 = container;

Container\* container3(container1);

container1->Show(3);

container2->Show(3);

container3->Show(3);

Container\* container4 = new Container(std::move(\*container2));

Container\* container5 = new Container;

\*container5 = std::move(\*container2);

container1->Show(3);

container4->Show(3);

container5->Show(3);

std::cout << "\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";

Container\* container6 = new Container;

\*container6 = \*container4;// = new Container(\*container4);

container4->Show(3);

container6->Show(3);

std::cout << (\*container4 == \*container6);

std::cout << "\n\n";

std::cout << (\*container4 != \*container6);

std::cout << "\n\n";

std::cout << "\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

container4->Pop();

container4->Show(3);

std::cout << "\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";

container1->~Container();

container2->~Container();

container3->~Container();

container4->~Container();

container5->~Container();

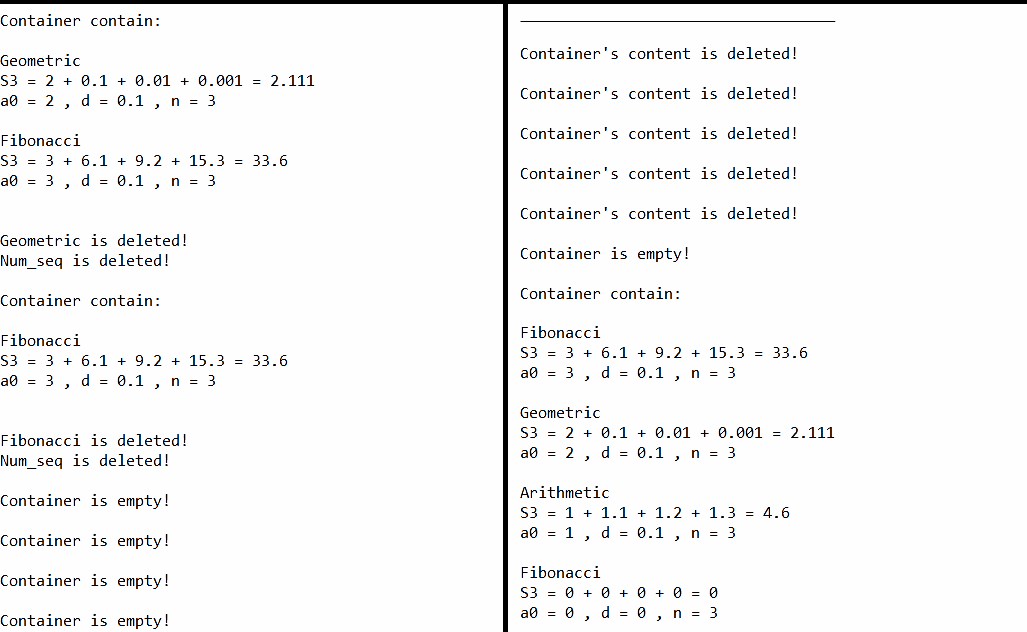
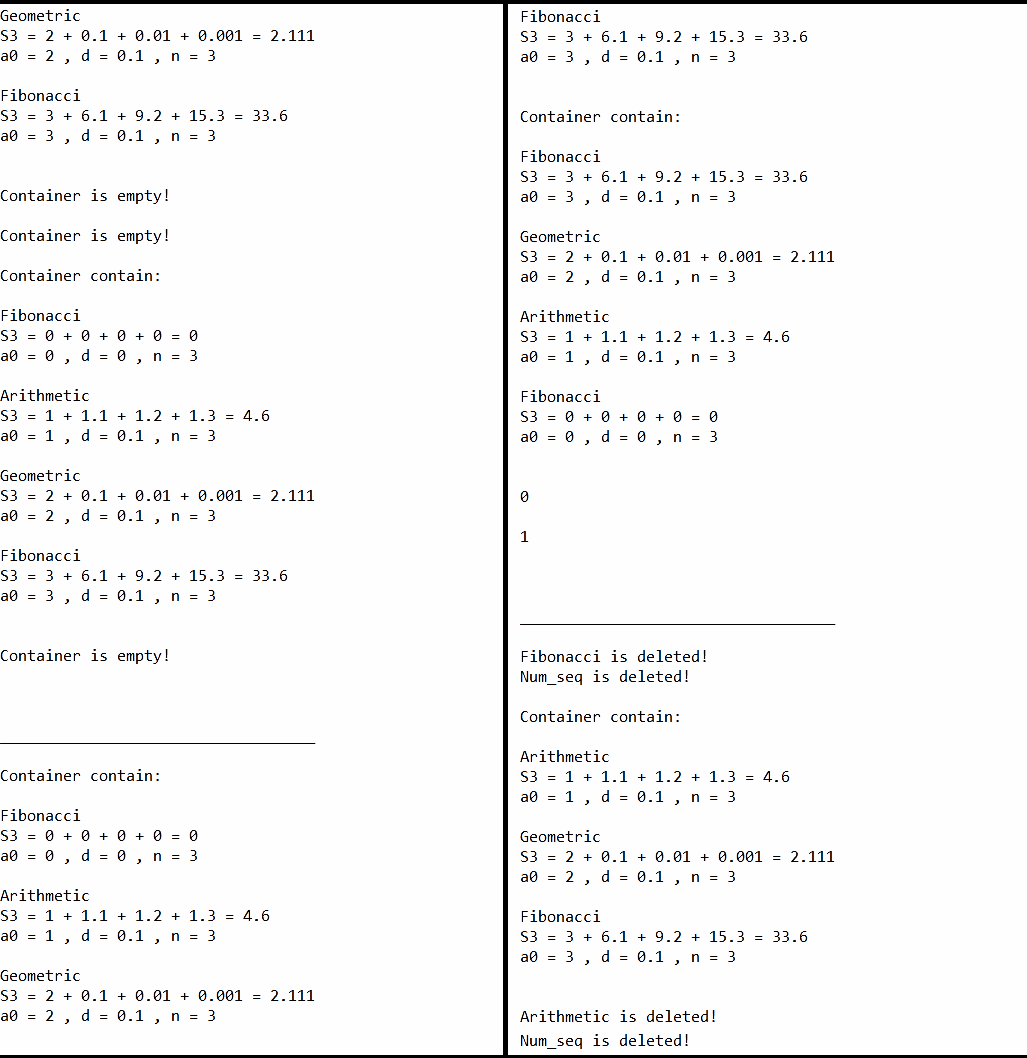
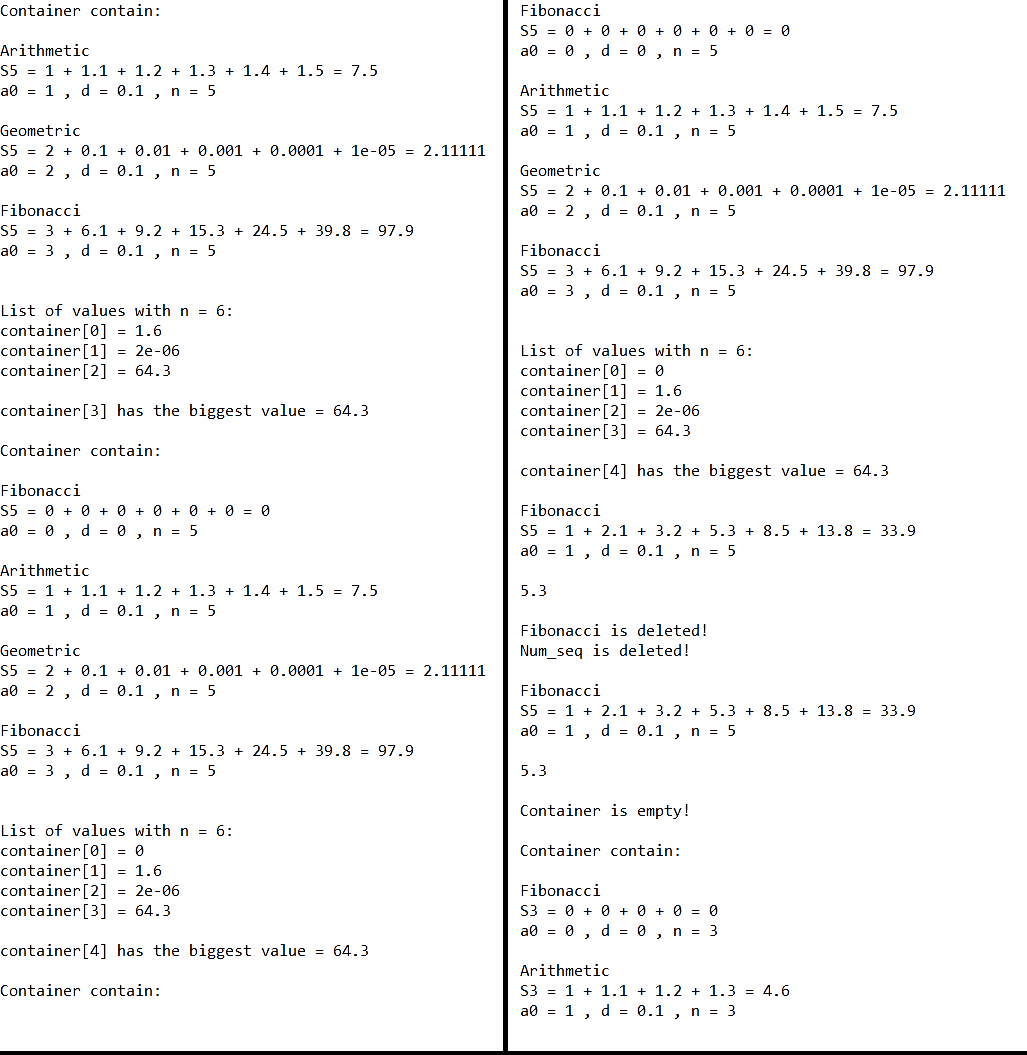
container4->Pop();

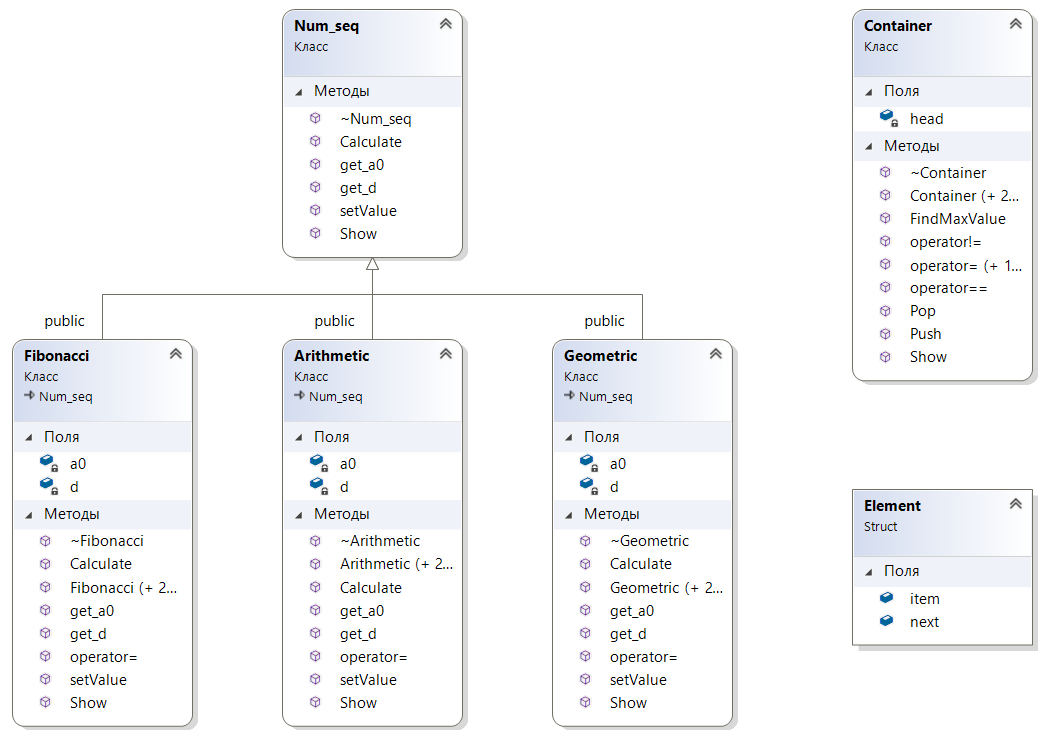
container4->Show(3);

container6->Show(3);

return 0;

}

**ВИКОНАННЯ ЕКСПЕРИМЕНТІВ ПО НАЛАГОДЖЕННЮ ПРОГРАМИ**

****

Діаграма классів

**Висновки.**

Протягом лабораторної роботи було розроблено программу, яка використовує принципи успадкування класів, з застосуванням ієрархії успадкування. Створено та продемонстровано роботоздатність віртуального класу та його нащадків. Створено класс-контейнер для зберігання, додавання та видалення єлементів класів – числових послідовностей.