**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**



### ЗВІТ

До лабораторної роботи № 9

**На тему:** *“* *Принцип поліморфізму”*

**З дисципліни:** *“Об’єктно-орієнтоване програмування”*

**Лектор:**

доцент каф.ПЗ

Коротєєва Т. О.

**Виконав:**

ст. гр. ПЗ-16

Шеремета А.І.

**Прийняв:**

асист. каф. ПЗ

Дивак І.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** Принцип поліморфізму.

**Мета** **роботи:** Навчитись створювати списки об’єктів базового типу, що включають об’єкти похідних типів. Освоїти способи вирішення проблеми неоднозначності при множинному наслідуванні. Вивчити плюси заміщення функцій при множинному наслідуванні. Навчитись використовувати чисті віртуальні функції, знати коли варто використовувати абстрактні класи.

**Теоретичні відомості**

Поліморфізм – це здатність об'єкта використовувати методи похідного класу, який не існує на момент створення базового.

Множинне наслідування — це функція C ++, де клас може наслідувати більше ніж один клас.

Конструктори успадкованих класів викликаються в тому ж порядку, в якому вони успадковуються.

Деструктори викликаються в порядку, зворотному конструкторам.

Неоднозначність у множинному наслідуванні

Найбільш очевидна проблема з множинним успадкуванням виникає під час перевизначення функції.

Припустимо, що два базові класи мають одну і ту ж функцію, яка не змінюється в похідному класі.

Якщо ви спробуєте викликати функцію за допомогою об’єкта похідного класу, компілятор покаже помилку. Це тому, що компілятор не знає, яку функцію викликати.

Цю проблему можна вирішити за допомогою функції розділення області видимості, щоб вказати, яку функцію класифікувати.

Ієрархічне наслідування C++

Якщо від базового класу наслідується більше одного класу, це називається ієрархічним наслідуванням. При ієрархічному наслідуванні всі функції, які є загальними в дочірніх класах, включені в базовий клас.

Наприклад, фізика, хімія, біологія є похідними від уроку природознавства. Аналогічно, собака, кіт, кінь походять від класу тварин.

class base\_class {

... .. ...

}

class first\_derived\_class: public base\_class {

... .. ...

}

class second\_derived\_class: public base\_class {

... .. ...

}

class third\_derived\_class: public base\_class {

... .. ...

}

Щоб уникнути дубльованого підоб’єкта базового класу, який виникає з «страшним ромбом», ви повинні використовувати ключове слово virtual у спадковій частині класів, які походять безпосередньо з вершини ромба.

За визначенням, абстрактний клас у C++ — це клас, який має принаймні одну чисту віртуальну функцію (тобто функцію, яка не має визначення). Класи, що успадковують абстрактний клас, повинні давати визначення для чисто віртуальної функції; інакше підклас сам став би абстрактним класом.

Зазвичай чисті віртуальні функції оголошуються в абстрактному базовому класі і не виконуються. Оскільки неможливо створити об’єкт абстрактного базового класу, як правило, нема необхідності і в виконання чистої віртуальної функції.

**Лабораторне завдання**

1. Розробити ієрархію класів відповідно до варіанту.
2. Використати множинне наслідування, продемонструвати вирішення проблеми з неоднозначністю доступу до членів базових класів за допомогою віртуального наслідування, за допомогою явного звертання до членів класу та за допомогою заміщення функцій в похідному класі (при потребі).
3. Створити списки об’єктів базового типу, в них помістити об’єкти похідного типу. Продемонструвати виклик функцій з об’єктів – елементів списку. Використати опратор dynamic\_cast (при потребі).
4. Створити абстрактний клас, використати чисто віртуальну функцію, що містить реалізацію в базовому класі.
5. Для вивільнення динамічної пам’яті використовувати віртуальні деструктори.
6. Сформувати звіт до лабораторної роботи. Відобразити в ньому діаграму наслідування класів.

**Варіант**

10. Масиви (звичайний массив, оптимізований по пам’яті массив)

**Хід роботи**

1. Реалізовую програму:

**Файл array.h**

#ifndef ARRAY\_H

#define ARRAY\_H

#include <QTableWidget>

#include <iostream>

class Array

{

public:

virtual int getSize()const=0;

virtual int getByIndex(int index)=0;

virtual void display(QTableWidget& tabArray)=0;

virtual ~Array();

};

#endif // ARRAY\_H

**Файл arraydefault.h**

#ifndef ARRAYDEFAULT\_H

#define ARRAYDEFAULT\_H

#include <array.h>

#include<QTableWidget>

class ArrayDefault: virtual public Array

{

public:

ArrayDefault();

ArrayDefault(int arrsize);

int getSize()const override;

int getByIndex(int index)override;

void display(QTableWidget& tabArray)override;

void sortUp();

void sortDown();

int getMax();

int getMin();

double getAvg();

void changeSize(int n);

void clear();

void operator>>(QTableWidget \*tableWidget);

virtual ~ArrayDefault();

protected:

int size;

int numbers[5];

};

#endif // ARRAYDEFAULT\_H

**Файл arraydynamic.h**

#ifndef ARRAYDYNAMIC\_H

#define ARRAYDYNAMIC\_H

#include <arraydefault.h>

#include <QTableWidget>

#include <functionsdynamic.h>

class ArrayDynamic : public ArrayDefault, FunctionsDynamic

{

public:

ArrayDynamic();

ArrayDynamic(int arrsize);

int getSize()const override;

int getByIndex(int index)override;

void display(QTableWidget& tabArrayDynamic) override;

void sortUp();

void sortDown();

int getMax();

int getMin();

double getAvg();

void changeSize(int n);

void clear();

void fillZeros()override;

void fillN(int N)override;

void operator>>(QTableWidget \*tabArrayDynamic);

virtual ~ArrayDynamic();

protected:

int sizeD;

int \*numbersD;

};

#endif // ARRAYDYNAMIC\_H

**Файл functionsdynamic.h**

#ifndef FUNCTIONSDYNAMIC\_H

#define FUNCTIONSDYNAMIC\_H

#include <array.h>

class FunctionsDynamic: virtual public Array

{

public:

FunctionsDynamic();

virtual void fillZeros()=0;

virtual void fillN(int N)=0;

virtual ~FunctionsDynamic();

};

#endif // FUNCTIONSDYNAMIC\_H

**Файл mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QTableWidget>

#include <array.h>

#include <arraydefault.h>

#include <arraydynamic.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_SortUpButt\_clicked();

void on\_SortDownButt\_clicked();

void on\_pushButton\_clicked();

void on\_Min\_clicked();

void on\_Avg\_clicked();

void on\_Max\_clicked();

void on\_Avg\_2\_clicked();

void on\_Get\_Index\_Dyn\_clicked();

void on\_change\_Dyn\_clicked();

void on\_change\_clicked();

void on\_SortUpButt\_Dyn\_clicked();

void on\_SortDownButt\_Dyn\_clicked();

void on\_pushButton\_2\_clicked();

void on\_Min\_Dyn\_clicked();

void on\_Avg\_Dyn\_clicked();

void on\_Max\_Dyn\_clicked();

void on\_clear\_clicked();

void on\_clear\_Dyn\_clicked();

void on\_textEdit\_numberFill\_selectionChanged();

private:

Ui::MainWindow \*ui;

QTableWidget \*tabA;

QTableWidget \*tabD;

};

#endif // MAINWINDOW\_H

**Файл array.cpp**

#include "array.h"

Array::~***Array***(){

}

**Файл arraydefault.cpp**

#include "arraydefault.h"

ArrayDefault::**ArrayDefault**()

{

}

ArrayDefault::**ArrayDefault**(*int* arrsize){

**this**->size = arrsize;

}

*int* ArrayDefault::***getSize***()**const**{

**return** size;

}

*int* ArrayDefault::***getByIndex***(*int* index){

**return** numbers[index];

}

*int* ArrayDefault::**getMax**(){

*int* max = **this**->numbers[0];

**for**(*int* i = 0; i < size; i++){

**if**(**this**->numbers[i] > max)

max = **this**->numbers[i];

}

**return** max;

}

*int* ArrayDefault::**getMin**(){

*int* min = **this**->numbers[0];

**for**(*int* i = 0; i < size; i++){

**if**(**this**->numbers[i] < min)

min = **this**->numbers[i];

}

**return** min;

}

*void* ArrayDefault::**changeSize**(*int* n){

**for** (*int* i = size+1; i < n; i++){

numbers[i] = 0;

size = n;

}

}

*void* ArrayDefault::**clear**(){

**for**(*int* i = 0; i < size; i++){

numbers[i] = 0;

}

size = 1;

}

*double* ArrayDefault::**getAvg**(){

*double* avg = 0;

*int* sum = 0;

**for**(*int* i = 0; i < size; i++){

*int* elem = numbers[i];

sum += elem;

}

avg = sum/size;

**return** avg;

}

*void* ArrayDefault::***display***(QTableWidget& tabArray){

tabArray.setColumnCount(size);

**for**(*int* i=0;i < size; ++i){

QTableWidgetItem \*newitem = **new** QTableWidgetItem;

newitem->setText(QString::number(numbers[i]));

tabArray.setItem(0,i,*newitem*);

}

}

*void* ArrayDefault::**operator**>>(QTableWidget \*tabArray){

**//** **ArrayDefault** **A(tabArray->columnCount());**

**for**(*int* i = 0; i<tabArray->columnCount(); i++){

*int* element = tabArray->item(0, i)->text().toInt();

numbers[i] = element;

tabArray->item(0, i)->setTextAlignment(Qt::AlignHCenter);

tabArray->item(0, i)->setTextAlignment(Qt::AlignVCenter);

}

}

*void* ArrayDefault::**sortUp**(){

*int* i, j, temp;

**for** (i = 1; i < size; i++){

temp = numbers[i];

j = i - 1;

**while** (j >= 0 && numbers[j] > temp){

numbers[j + 1] = numbers[j];

j = j - 1;

}

numbers[j + 1] = temp;

}

}

*void* ArrayDefault::**sortDown**(){

**for** (*int* step = 0; step < (size-1); ++step) {

*int* swapped = 0;

**for** (*int* i = 0; i < (size-step-1); ++i) {

**if** (numbers[i] < numbers[i + 1]) {

*int* temp = numbers[i];

numbers[i] = numbers[i + 1];

numbers[i + 1] = temp;

swapped = 1;

}

}

**if** (swapped == 0)

**break**;

}

}

ArrayDefault::~***ArrayDefault***(){

}

**Файл arraydynamic.cpp**

#include "arraydynamic.h"

#include <algorithm>

ArrayDynamic::**ArrayDynamic**()

{

sizeD = 0;

numbersD = **nullptr**;

}

ArrayDynamic::**ArrayDynamic**(*int* arrsize){

sizeD = arrsize;

numbersD = **new** *int* [arrsize];

}

ArrayDynamic::~***ArrayDynamic***(){

**delete** [] numbersD;

}

*int* ArrayDynamic::***getSize***()**const**{

**return** sizeD;

}

*int* ArrayDynamic::***getByIndex***(*int* index){

**return** numbersD[index];

}

*void* ArrayDynamic::***display***(QTableWidget& tabArrayDynamic){

tabArrayDynamic.setColumnCount(sizeD);

**for**(*int* i=0;i < sizeD; ++i){

QTableWidgetItem \*newitem = **new** QTableWidgetItem;

newitem->setText(QString::number(numbersD[i]));

tabArrayDynamic.setItem(0,i,*newitem*);

}

}

*int* ArrayDynamic::**getMax**(){

*int* max = **this**->numbersD[0];

**for**(*int* i = 0; i < sizeD; i++){

**if**(**this**->numbers[i] > max)

max = **this**->numbersD[i];

}

**return** max;

}

*int* ArrayDynamic::**getMin**(){

*int* min = **this**->numbersD[0];

**for**(*int* i = 0; i < sizeD; i++){

**if**(**this**->numbers[i] < min)

min = **this**->numbersD[i];

}

**return** min;

}

*void* ArrayDynamic::**changeSize**(*int* n){

**delete** [] numbersD;

sizeD = n;

numbersD = **new** *int* [n];

}

*void* ArrayDynamic::**clear**(){

sizeD = 0;

**delete**[] numbersD;

numbersD = **nullptr**;

}

*void* ArrayDynamic::***fillZeros***()

{

**for**(*int* i = 0; i < sizeD; i++){

numbersD[i] = 0;

}

}

*void* ArrayDynamic::***fillN***(*int* N){

**for**(*int* i = 0; i < sizeD; i++){

numbersD[i] = N;

}

}

*double* ArrayDynamic::**getAvg**(){

*double* avg = 0;

*int* sum = 0;

**for**(*int* i = 0; i < sizeD; i++){

*int* elem = numbersD[i];

sum += elem;

}

avg = sum/sizeD;

**return** avg;

}

*void* ArrayDynamic::**operator**>>(QTableWidget \*tabArrayDynamic){

**//** **ArrayDynamic** **B(tabArrayDynamic->columnCount());**

**for**(*int* i = 0; i<tabArrayDynamic->columnCount(); i++){

*int* element = tabArrayDynamic->item(0, i)->text().toInt();

numbersD[i] = element;

tabArrayDynamic->item(0, i)->setTextAlignment(Qt::AlignHCenter);

tabArrayDynamic->item(0, i)->setTextAlignment(Qt::AlignVCenter);

}

}

*void* ArrayDynamic::**sortUp**(){

*int* i, j, temp;

**for** (i = 1; i < sizeD; i++){

temp = numbersD[i];

j = i - 1;

**while** (j >= 0 && numbersD[j] > temp){

numbersD[j + 1] = numbersD[j];

j = j - 1;

}

numbersD[j + 1] = temp;

}

}

*void* ArrayDynamic::**sortDown**(){

*int* i,j;

*int* temp;

**for**(i=0;i<sizeD;i++)

{

**for**(j=i+1;j<sizeD;j++)

{

**if**(numbersD[i]<numbersD[j])

{

temp =numbersD[i];

numbersD[i]=numbersD[j];

numbersD[j]=temp;

}

}

}

}

**Файл functionsdynamic.cpp**

#include "functionsdynamic.h"

FunctionsDynamic::**FunctionsDynamic**()

{

}

FunctionsDynamic::~***FunctionsDynamic***(){

}

**Файл mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <array.h>

#include <arraydefault.h>

#include <arraydynamic.h>

#include <QMessageBox>

#include <QPalette>

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(**new** Ui::MainWindow)

{

ui->setupUi(**this**);

tabD = **this**->ui->tabArrayDynamic;

tabA = **this**->ui->tabArray;

ui->textEdit\_numberFill->setAlignment(Qt::AlignHCenter);

}

MainWindow::~***MainWindow***()

{

**delete** ui;

}

*void* MainWindow::**on\_SortUpButt\_clicked**()

{

ArrayDefault A(tabA->columnCount());

A>>tabA;

A.sortUp();

ui->tabArray->clear();

A.*display*(*\*tabA*);

}

*void* MainWindow::**on\_SortDownButt\_clicked**()

{

ArrayDefault A(tabA->columnCount());

A>>tabA;

A.sortDown();

ui->tabArray->clear();

A.*display*(*\*tabA*);

}

*void* MainWindow::**on\_pushButton\_clicked**()

{

ArrayDynamic B(tabD->columnCount());

B.*fillZeros*();

B.*display*(*\*tabD*);

}

*void* MainWindow::**on\_Min\_clicked**()

{

ArrayDefault A(tabA->columnCount());

A>>tabA;

ui->lineEdit->setText(QString::number(A.getMin()));

ui->lineEdit->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}

*void* MainWindow::**on\_Avg\_clicked**()

{

ArrayDefault A(tabA->columnCount());

A>>tabA;

ui->lineEdit->setText(QString::number(A.getAvg()));

ui->lineEdit->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}

*void* MainWindow::**on\_Max\_clicked**()

{

ArrayDefault A(tabA->columnCount());

A>>tabA;

ui->lineEdit->setText(QString::number(A.getMax()));

ui->lineEdit->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}

*void* MainWindow::**on\_Avg\_2\_clicked**()

{

ui->lineEdit\_2->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

*int* index = ui->Index->text().toInt();

ArrayDefault A(tabA->columnCount());

A>>tabA;

*int* urindex = A.*getByIndex*(index);

ui->lineEdit\_2->setText(QString::number(urindex));

}

*void* MainWindow::**on\_Get\_Index\_Dyn\_clicked**()

{

ui->lineEdit\_3->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

*int* index = ui->Index\_Dyn->text().toInt();

ArrayDynamic B(tabD->columnCount());

B>>tabD;

*int* urindex = B.*getByIndex*(index);

ui->lineEdit\_3->setText(QString::number(urindex));

}

*void* MainWindow::**on\_change\_Dyn\_clicked**()

{

*int* newSize = ui->newSize\_Dyn->text().toInt();

ArrayDefault B(tabD->columnCount());

B.changeSize(newSize);

tabD->setColumnCount(newSize);

}

*void* MainWindow::**on\_change\_clicked**()

{

*int* newSize = ui->newSize->text().toInt();

ArrayDefault A(tabA->columnCount());

A.changeSize(newSize);

tabA->setColumnCount(newSize);

}

*void* MainWindow::**on\_SortUpButt\_Dyn\_clicked**()

{

ArrayDynamic B(tabD->columnCount());

B>>tabD;

B.sortUp();

B.*display*(*\*tabD*);

}

*void* MainWindow::**on\_SortDownButt\_Dyn\_clicked**()

{

ArrayDynamic B(tabD->columnCount());

B>>tabD;

B.sortDown();

B.*display*(*\*tabD*);

}

*void* MainWindow::**on\_pushButton\_2\_clicked**()

{

**if**(ui->textEdit\_numberFill->toPlainText().toInt()){

ArrayDynamic B(tabD->columnCount());

B.*fillN*(ui->textEdit\_numberFill->toPlainText().toInt());

B.*display*(*\*tabD*);

}

**else**{

QMessageBox::warning(**this**, "Error", "You forgor to enter number");

}

}

*void* MainWindow::**on\_Min\_Dyn\_clicked**()

{

**if**(ui->tabArrayDynamic->item(0,1))

{

ArrayDefault B(tabD->columnCount());

B>>tabD;

ui->lineEdit\_Dyn->setText(QString::number(B.getMin()));

ui->lineEdit\_Dyn->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}**else** {

QMessageBox::warning(**this**, "Error", "Fill dynamic array");

}

}

*void* MainWindow::**on\_Avg\_Dyn\_clicked**()

{

**if**(ui->tabArrayDynamic->item(0,1))

{

ArrayDefault B(tabD->columnCount());

B>>tabD;

ui->lineEdit\_Dyn->setText(QString::number(B.getAvg()));

ui->lineEdit\_Dyn->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}**else** {

QMessageBox::warning(**this**, "Error", "Fill dynamic array");

}

}

*void* MainWindow::**on\_Max\_Dyn\_clicked**()

{

**if**(ui->tabArrayDynamic->item(0,1))

{

ArrayDefault B(tabD->columnCount());

B>>tabD;

ui->lineEdit\_Dyn->setText(QString::number(B.getMax()));

ui->lineEdit\_Dyn->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}**else** {

QMessageBox::warning(**this**, "Error", "Fill dynamic array");

}

}

*void* MainWindow::**on\_clear\_clicked**()

{

ui->tabArray->clear();

ui->lineEdit->clear();

ui->Index->setValue(0);

ui->lineEdit\_2->clear();

ui->newSize->cleanText();

}

*void* MainWindow::**on\_clear\_Dyn\_clicked**()

{

ui->tabArrayDynamic->clear();

ui->lineEdit\_Dyn->clear();

ui->newSize\_Dyn->cleanText();

ui->Index\_Dyn->setValue(0);

ui->lineEdit\_3->clear();

ui->textEdit\_numberFill->clear();

}

*void* MainWindow::**on\_textEdit\_numberFill\_selectionChanged**()

{

ui->textEdit\_numberFill->setStyleSheet("border-radius: 10px;color:white;background-color:#333333;font-weight:525;");

}

**Виконання лабораторної роботи**

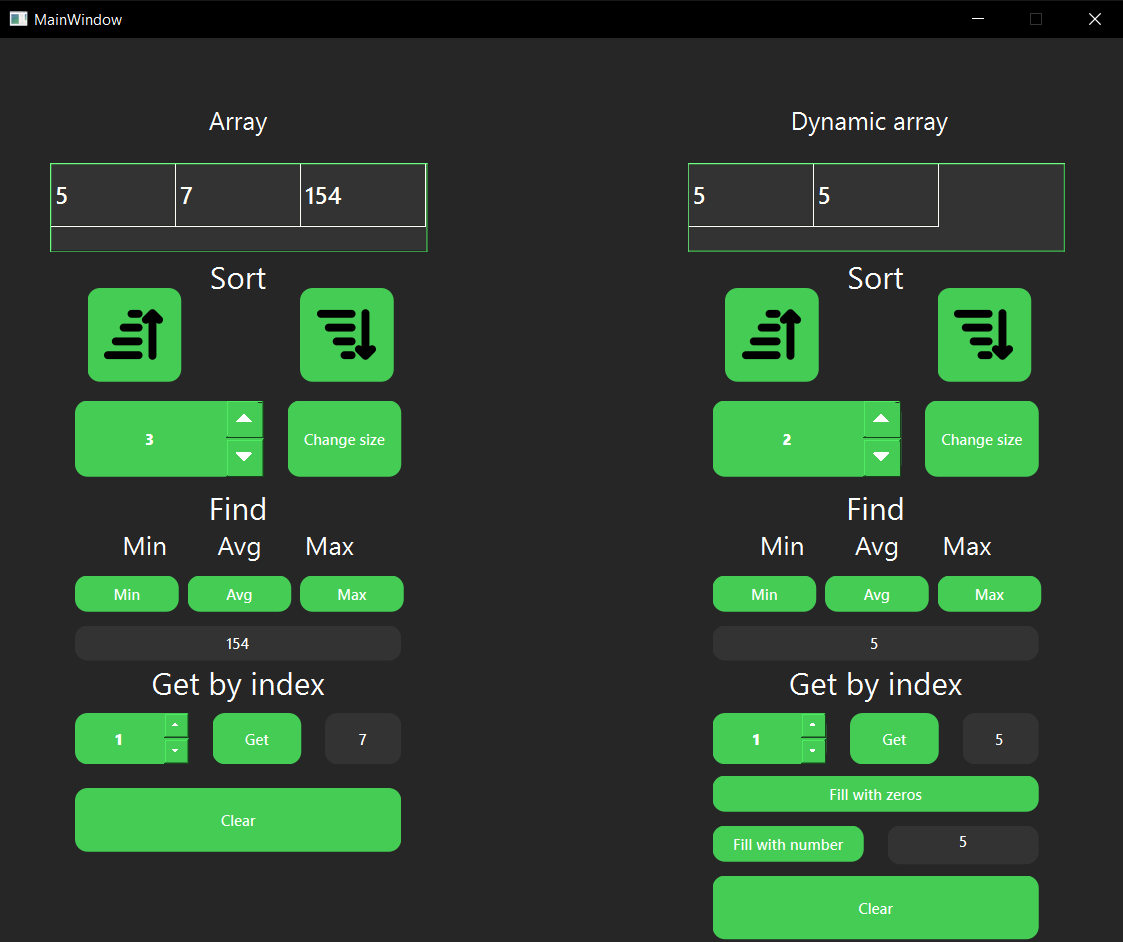


Рис. 1 Виконання програми

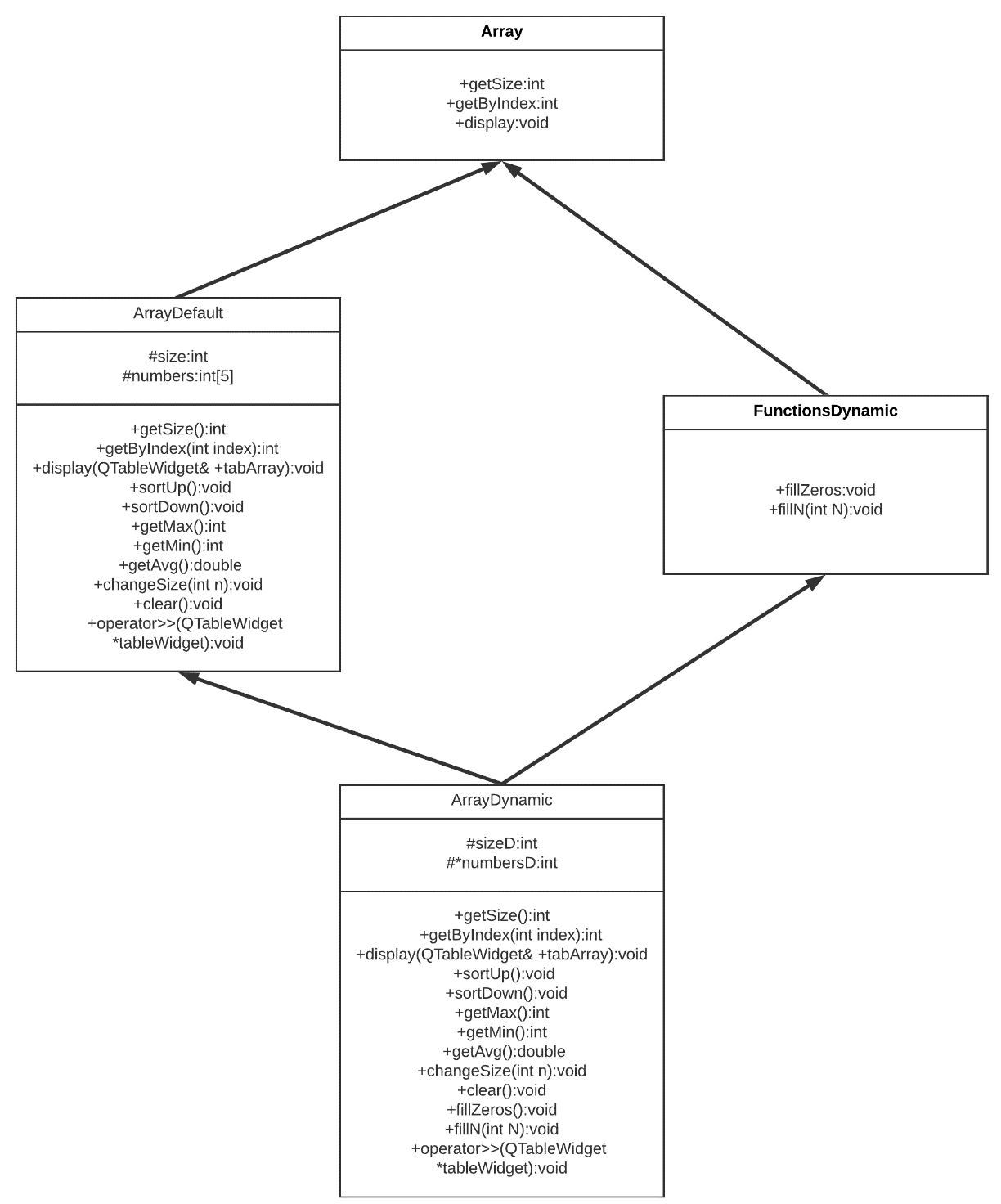


Рис.2 UML-діаграма

**Висновки**

На цій лабораторній роботі я навчився створювати чисто віртуальні функції, використовувати на практиці множинне наслідування, віртуальне наслідування, абстрактні класи, віртуальні деструктори та освоїв принципи поліморфізму.