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.

Виконала: ст.гр. ПЗ-23

Кохман О.В.

Прийняла: доцент каф. ПЗ

Коротєєва Т.О.

«_____»_____2022р.

∑ ____ _____.

Львів – 2022

Тема: поліморфізм класів.

Мета: 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

Віртуальна функція в мові C++ — це особливий тип функції, яка, при її виклику, виконує «найдочірніший» метод, який існує між батьківським і дочірніми класами. Ця властивість відома як **поліморфізм**. Дочірній метод викликається тоді, коли збігається **сигнатура** (ім’я, типи параметрів і чи є метод константним) і тип повернення дочірнього методу з сигнатурою і типом повернення методу батьківського класу. Такі методи називаються **перевизначеннями** (або ***“перевизначеними методами”***).

Щоб зробити функцію віртуальною, потрібно просто вказати **ключове слово virtual** перед оголошенням функції.

Якщо функція позначена як віртуальна, то всі відповідні перевизначення теж вважаються віртуальними, навіть якщо біля них явно не вказано ключове слова virtual. Однак, наявність ключового слова virtual біля методів дочірніх класів послужить корисним нагадуванням про те, що ці методи є віртуальними, а не звичайними. Отже, хорошою практикою є вказування ключового слова virtual біля перевизначень в дочірніх класах, навіть якщо це не є строго необхідним.

«Якщо все так добре з віртуальними функціями, то чому б не зробити всі методи віртуальними?» — запитаєте Ви. Відповідь: “Це неефективно!”. Обробка і виконання виклику віртуального методу займає більше часу, ніж обробка і виконання виклику звичайного методу. Крім того, компілятор також повинен виділяти один додатковий вказівник для кожного об’єкта класу, який має одну або кілька віртуальних функцій.

Індивідуальне 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 оператор 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.

2. Лабіринт Будівельник

Тут те саме. Базовий клас-інтерфейс **MazeBuilder**. Його реалізації: **SimpleMazeBuilder**, **MiddleMazeBuilder**, **ComplexMazeBuilder**. Можна зробити, щоб сам об’єкт **Maze** міг виводити себе якось на форму. Його елементами можуть бути, до прикладу, **Entrance**, **Core**, **Exit**.

Код програми

Назва файлу: Builder.h

```
#ifndef BUILDER_H
#define BUILDER_H
#pragma once
#include "Maze.h"

class Builder { // abstract class
protected:
	string colour;
	string shape;
	string style;
	string complexity;
	int height;
	int width;
public:
	virtual void setShape(string shape) = 0;
	virtual void setColour(string colour) = 0;
	virtual void setStyle(string style) = 0;
	virtual void setHeight() = 0;
	virtual void setWidth() = 0;
	virtual void setComplexity() = 0;
};
#endif
```

Назва файлу: Builder.cpp

```
#include "Builder.h"

void Builder::setComplexity() {
	this->complexity = "Classic";
}
```

Назва файлу: MazeBuilder.h

```
#ifndef MAZEBUILDER_H
#define MAZEBUILDER_H
#pragma once
#include "Builder.h"

class MazeBuilder : public Builder {
protected:
	string colour;
	string shape;
	string style;
	string complexity;
	int height;
	int width;
public:
	virtual void setShape(string shape) override;
	virtual void setColour(string colour) override;
	virtual void setStyle(string style) override;
	virtual void setHeight() override;
	virtual void setWidth() override;
	virtual void setComplexity() override {
		this->complexity = "Classic";
	}
	virtual Maze* getMaze();
	virtual ~MazeBuilder() {};
};
#endif
```

Назва файлу: MazeBuilder.cpp

```
#include "MazeBuilder.h"

void MazeBuilder::setShape(string shape) {
	this->shape = shape;
}
void MazeBuilder::setColour(string colour) {
	this->colour = colour;
}
void MazeBuilder::setStyle(string style) {
	this->style = style;
}
void MazeBuilder::setHeight() {
	this->height = 0;
}
void MazeBuilder::setWidth() {
	this->width = 0;
}
Maze* MazeBuilder::getMaze() {
	return new Maze(this->shape, this->colour, this->style, this->height,
this->width, this->complexity);
}
```

Назва файлу: SimpleMazeBuilder.h

```
#ifndef SIMPLEMAZEBUILDER_H
#define SIMPLEMAZEBUILDER_H
#pragma once
#include "MazeBuilder.h"
class SimpleMazeBuilder : public MazeBuilder {
public:
	virtual void setHeight() override;
	virtual void setWidth() override;
	virtual void setComplexity() override;
	virtual ~SimpleMazeBuilder() {};
```

```
};
#endif
```
Назва файлу: SimpleMazeBuilder.cpp
```
#include "SimpleMazeBuilder.h"

void SimpleMazeBuilder::setHeight() {
	this->height = 100;
}
void SimpleMazeBuilder::setWidth() {
	this->width = 100;
}
void SimpleMazeBuilder::setComplexity() {
	this->complexity = "Simple";
}
```
Назва файлу: MiddleMazeBuilder.h
```
#ifndef MIDDLEMAZEBUILDER_H
#define MIDDLEMAZEBUILDER_H
#pragma once
#include "MazeBuilder.h"
class MiddleMazeBuilder : public MazeBuilder {
public:
	virtual void setHeight() override;
	virtual void setWidth() override;
	virtual void setComplexity() override;
	virtual ~MiddleMazeBuilder() {};
};
#endif
```
Назва файлу: MiddleMazeBuilder.cpp
```
#include "MiddleMazeBuilder.h"

void MiddleMazeBuilder::setHeight() {
	this->height = 200;
}
void MiddleMazeBuilder::setWidth() {
	this->width = 200;
}
void MiddleMazeBuilder::setComplexity() {
	this->complexity = "Middle";
}
```
Назва файлу: ComplexMazeBuilder.h
```
#ifndef COMPLEXMAZEBUILDER_H
#define COMPLEXMAZEBUILDER_H
#pragma once
#include "MazeBuilder.h"
class ComplexMazeBuilder : public MazeBuilder {
public:
	virtual void setHeight() override;
	virtual void setWidth() override;
	virtual void setComplexity() override;
	virtual ~ComplexMazeBuilder() {};
};
#endif
```
Назва файлу: ComplexMazeBuilder.cpp
```
#include "ComplexMazeBuilder.h"

void ComplexMazeBuilder::setHeight() {
```

```
	this->height = 300;
}
void ComplexMazeBuilder::setWidth() {
	this->width = 300;
}
void ComplexMazeBuilder::setComplexity() {
	this->complexity = "Complex";
}
```

Назва файлу: Logger.h

```
#ifndef LOGGER_H
#define LOGGER_H
#pragma once
#include <iostream>
using namespace std;
class Logger {
protected:
	void log(string identifier) {
		cout << "There was executed action: " << identifier << endl;
	}
};
#endif
```

Назва файлу: Logger.cpp

```
#include "Logger.h"
```

Назва файлу: Finisher.h

```
#ifndef FINISHER_H
#define FINISHER_H
#pragma once
#include "Logger.h"
class Finisher : public virtual Logger {
	virtual void finish() = 0;
};
#endif
```

Назва файлу: Finisher.cpp

```
#include "Finisher.h"
```

Назва файлу: Printer.h

```
#ifndef PRINTER_H
#define PRINTER_H
#pragma once
#include "Logger.h"
class Printer : public virtual Logger {
	virtual void print(System::Windows::Forms::Label^ label) = 0;
};
#endif
```

Назва файлу: Printer.cpp

```
#include "Printer.h"
```

Назва файлу: MyForm.h

```
#pragma once
#include "standardString.h"
#include "SimpleMazeBuilder.h"
#include "MiddleMazeBuilder.h"
#include "ComplexMazeBuilder.h"
#include "Maze.h"
#include <list>
namespace Main {
```

```
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::ComboBox^ comboBox1;
	protected:
	private: System::Windows::Forms::ComboBox^ comboBox2;
	private: System::Windows::Forms::ComboBox^ comboBox3;
	private: System::Windows::Forms::Button^ button1;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::Label^ label6;

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
			this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());
			this->comboBox3 = (gcnew System::Windows::Forms::ComboBox());
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->label1 = (gcnew System::Windows::Forms::Label());
```

```
                    this->label2 = (gcnew System::Windows::Forms::Label());
                    this->label3 = (gcnew System::Windows::Forms::Label());
                    this->label4 = (gcnew System::Windows::Forms::Label());
                    this->label5 = (gcnew System::Windows::Forms::Label());
                    this->label6 = (gcnew System::Windows::Forms::Label());
                    this->SuspendLayout();
                    //
                    // comboBox1
                    //
                    this->comboBox1->FormattingEnabled = true;
                    this->comboBox1->Items->AddRange(gcnew cli::array<
System::Object^  >(4) { L"Square", L"Circular", L"Triangular", L"Hexagonal" });
                    this->comboBox1->Location = System::Drawing::Point(84, 51);
                    this->comboBox1->Name = L"comboBox1";
                    this->comboBox1->Size = System::Drawing::Size(121, 24);
                    this->comboBox1->TabIndex = 0;
                    //
                    // comboBox2
                    //
                    this->comboBox2->FormattingEnabled = true;
                    this->comboBox2->Items->AddRange(gcnew cli::array<
System::Object^  >(3) { L"Red", L"Black", L"White" });
                    this->comboBox2->Location = System::Drawing::Point(250, 51);
                    this->comboBox2->Name = L"comboBox2";
                    this->comboBox2->Size = System::Drawing::Size(121, 24);
                    this->comboBox2->TabIndex = 1;
                    //
                    // comboBox3
                    //
                    this->comboBox3->FormattingEnabled = true;
                    this->comboBox3->Items->AddRange(gcnew cli::array<
System::Object^  >(3) { L"Orthogonal", L"Sigma", L"Delta" });
                    this->comboBox3->Location = System::Drawing::Point(420, 51);
                    this->comboBox3->Name = L"comboBox3";
                    this->comboBox3->Size = System::Drawing::Size(121, 24);
                    this->comboBox3->TabIndex = 2;
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(250, 159);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(121, 23);
                    this->button1->TabIndex = 3;
                    this->button1->Text = L"Build all mazes";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->Location = System::Drawing::Point(81, 276);
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(0, 16);
                    this->label1->TabIndex = 4;
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
                    this->label2->Location = System::Drawing::Point(247, 276);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(0, 16);
                    this->label2->TabIndex = 5;
```

```
                    //
                    // label3
                    //
                    this->label3->AutoSize = true;
                    this->label3->Location = System::Drawing::Point(417, 276);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(0, 16);
                    this->label3->TabIndex = 6;
                    //
                    // label4
                    //
                    this->label4->AutoSize = true;
                    this->label4->Location = System::Drawing::Point(96, 32);
                    this->label4->Name = L"label4";
                    this->label4->Size = System::Drawing::Size(96, 16);
                    this->label4->TabIndex = 7;
                    this->label4->Text = L"choose shape:";
                    //
                    // label5
                    //
                    this->label5->AutoSize = true;
                    this->label5->Location = System::Drawing::Point(263, 32);
                    this->label5->Name = L"label5";
                    this->label5->Size = System::Drawing::Size(95, 16);
                    this->label5->TabIndex = 8;
                    this->label5->Text = L"choose colour:";
                    //
                    // label6
                    //
                    this->label6->AutoSize = true;
                    this->label6->Location = System::Drawing::Point(432, 32);
                    this->label6->Name = L"label6";
                    this->label6->Size = System::Drawing::Size(86, 16);
                    this->label6->TabIndex = 9;
                    this->label6->Text = L"choose style:";
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(604, 488);
                    this->Controls->Add(this->label6);
                    this->Controls->Add(this->label5);
                    this->Controls->Add(this->label4);
                    this->Controls->Add(this->label3);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->label1);
                    this->Controls->Add(this->button1);
                    this->Controls->Add(this->comboBox3);
                    this->Controls->Add(this->comboBox2);
                    this->Controls->Add(this->comboBox1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this,
&MyForm::MyForm_Load);
                    this->ResumeLayout(false);
                    this->PerformLayout();

                }
#pragma endregion
        private: System::Void MyForm_Load(System::Object^ sender,
System::EventArgs^ e) {
```

```
	}
		template <typename T>
		void callMazeOperations(T* builder,
System::Windows::Forms::Label^ label) {
			builder->setShape(toStandardString(comboBox1->Text));
			builder->setColour(toStandardString(comboBox2->Text));
			builder->setStyle(toStandardString(comboBox3->Text));
			builder->setHeight();
			builder->setWidth();
			builder->setComplexity();
			Maze maze = builder->getMaze();
			maze.finish();
			maze.print(label);
		}
	private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
		SimpleMazeBuilder* simpleBuilder = new SimpleMazeBuilder();
		MiddleMazeBuilder* middleBuilder = new MiddleMazeBuilder();
		ComplexMazeBuilder* complexBuilder = new ComplexMazeBuilder();
		MazeBuilder* mazeBuilder = new MazeBuilder();
		list<MazeBuilder*> myList;
		myList.push_back(simpleBuilder);
		myList.push_back(middleBuilder);
		myList.push_back(complexBuilder);
		callMazeOperations(dynamic_cast<SimpleMazeBuilder*>(myList.front()),
label1);
		myList.pop_front();
		callMazeOperations(dynamic_cast<MiddleMazeBuilder*>(myList.front()),
label2);
		myList.pop_front();

	callMazeOperations(dynamic_cast<ComplexMazeBuilder*>(myList.front()),
label3);
		myList.pop_front();
	}


	};
}
```

Назва файлу: MyForm.cpp

```
#include "MyForm.h"
using namespace Main;
int main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}
```

Назва файлу: standardString.h

```
#ifndef STANDARDSTRING_H
#define STANDARDSTRING_H
#include <iostream>
static std::string toStandardString(System::String^ string) {
    using System::Runtime::InteropServices::Marshal;
    System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);
    char* charPointer = reinterpret_cast<char*>(pointer.ToPointer());
    std::string returnString(charPointer, string->Length);
    Marshal::FreeHGlobal(pointer);
    return returnString;
}
```

```
#endif
```

Протокол роботи

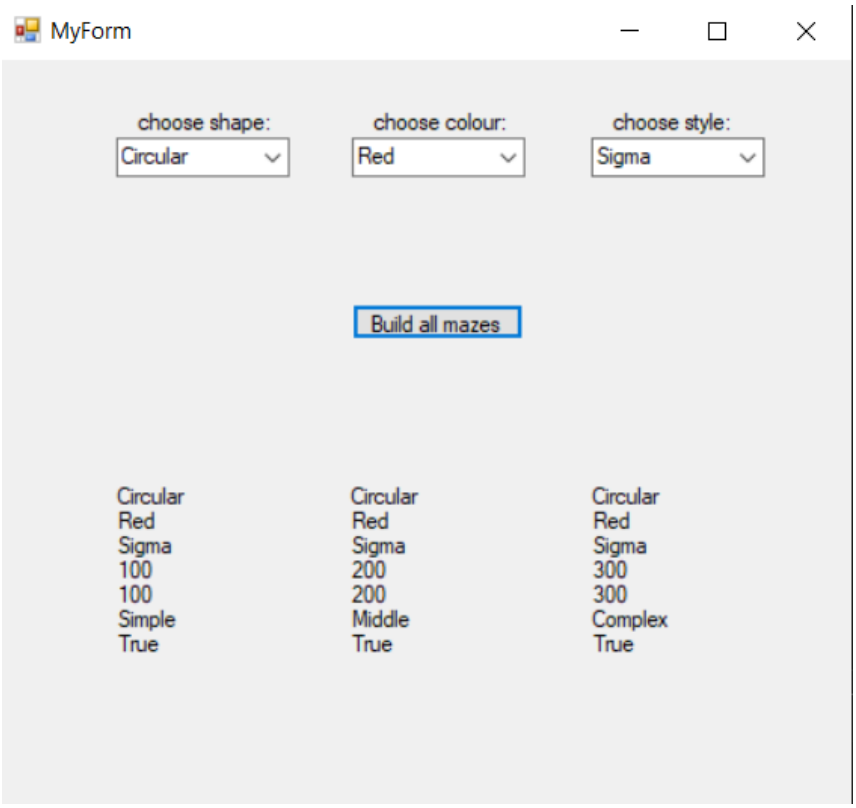

Рис. 1 Результат виконання програми.

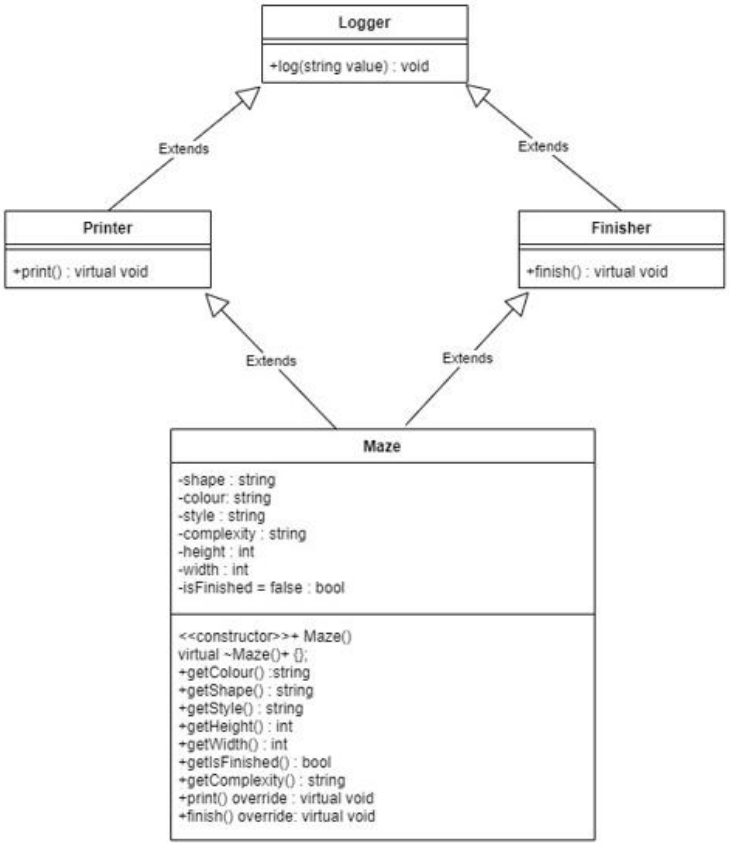

Рис. 2 Діаграма наслідування класів.

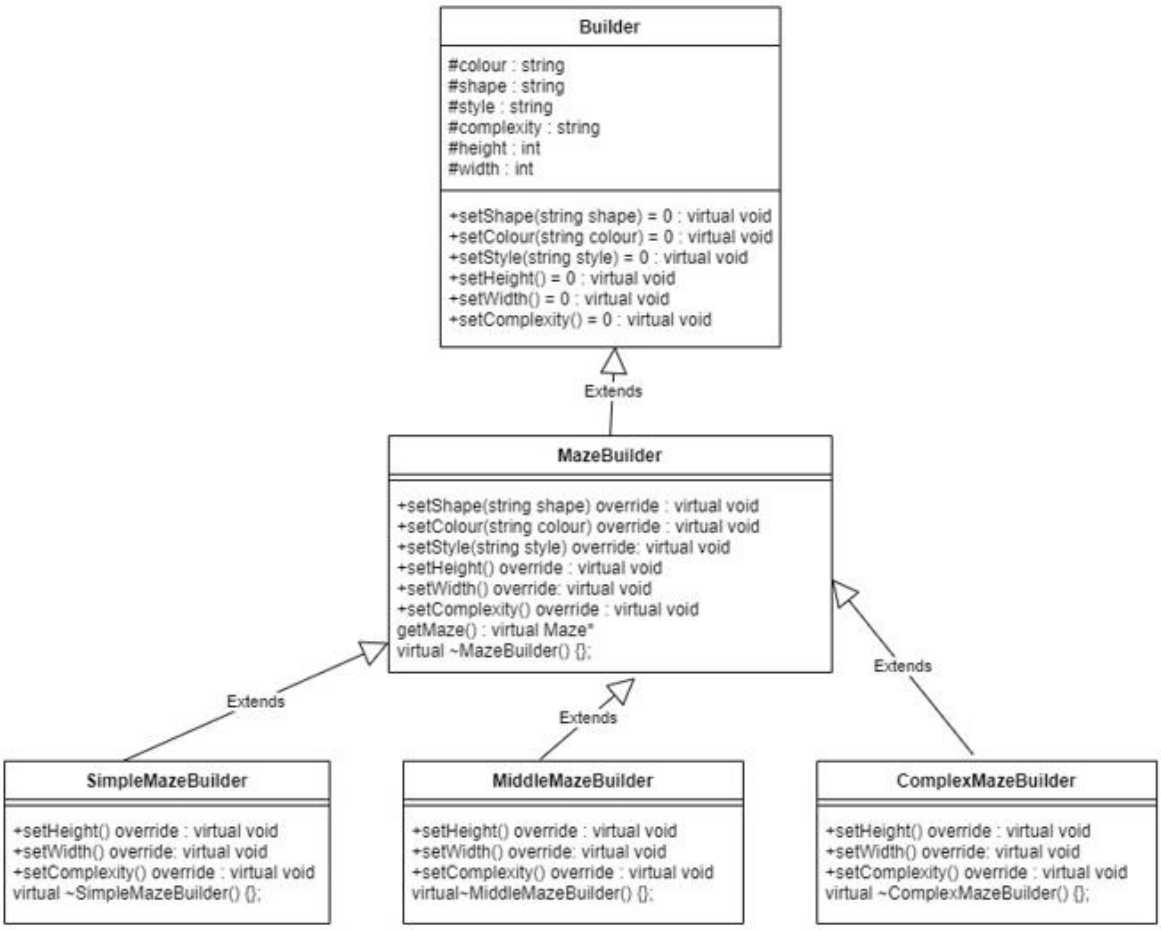

Рис. 3 Діаграма наслідування класів.

Висновок

На цій лабораторній роботі я ознайомилась і поліморфімом в c++, а саме із віртуальними функціями , віртуальним наслідуванням, множинним наслідуванням , віртуальними деструкторами , а також із оператором dynamic_cast, реалізувала програму та продемонструвала результати на формі у Visual Studio 2022.