Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4 з дисципліни
«Основи програмування 2. Модульне програмування»

«Успадкування та поліморфізм»

Варіант 9

Виконав студент ІП-15, Дзюбенко Даниїл Дмитрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 4

Успадкування та поліморфізм

Мета – вивчити механізми створення і використання класів та об’єктів.

Варіант 9

Задача

9. Створити клас TMatrix, який представляє матрицю і містить методи для обчислення детермінанта та суми елементів матриці. На основі цього класу створити класи-нащадки, які представляють квадратні матриці 2-го та 3-го порядку. За допомогою цих класів обчислити вираз

$$S = \left(\sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij} \right) + |A| + |B|,$$

де $A = \|a_{ij}\|_1^3$ – матриця 3-го порядку, а $B = \|b_{ij}\|_1^2$ – матриця 2-го порядку.

Код

C++

```cpp
#include "Classes.h"

int main() {
    srand(time(NULL));

    double a, b, sum, s;
    SquareMatrix2Order matrixB;
    SquareMatrix3Order matrixA;

    cout << "Matrix A:\n";
    matrixA.printMatrix();

    cout << "\nMatrix B:\n";
    matrixB.printMatrix();

    a = matrixA.getDeterminant();
    b = matrixB.getDeterminant();
    sum = matrixA.getSumOfElements();
    s = a + b + sum;

    cout << "\nDet A: " << a << endl;
    cout << "Det B: " << b << endl;
    cout << "Sum of elements Matrix A: " << sum << endl;
    cout << "\nS = " << s << endl;

    system("pause");
    return 0;
}
```

```h
#pragma once
#include <iostream>
#include <iomanip>
using namespace std;

class TMatrix {
public:
    TMatrix(int n, int m);
    virtual double getDeterminant() = 0;
    double getSumOfElements();
    void printMatrix();

protected:
    int n, m;
    double** matr;
};

class SquareMatrix2Order : public TMatrix {
public:
    SquareMatrix2Order():TMatrix(2, 2){};
    ~SquareMatrix2Order();
    double getDeterminant() override;
};

class SquareMatrix3Order : public TMatrix {
public:
    SquareMatrix3Order():TMatrix(3, 3){};
    ~SquareMatrix3Order();
    double getDeterminant() override;
};
```

```cpp
#include "Classes.h"

TMatrix::TMatrix(int n, int m) {
    this->n = n;
    this->m = m;
    matr = new double* [n];
    for (int i = 0; i < n; i++)
    {
        matr[i] = new double[m];
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            matr[i][j] = rand() % 101 - 50;
        }
    }
}

void TMatrix::printMatrix() {
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << setw(6) << matr[i][j];
        }
        cout << endl;
    }
}
```

```cpp
double TMatrix::getSumOfElements() {
    double sum = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            sum += matr[i][j];
        }
    }
    return sum;
}

SquareMatrix2Order::~SquareMatrix2Order() {
    delete[] matr;
};

double SquareMatrix2Order::getDeterminant() {
    return matr[0][0] * matr[1][1] - matr[0][1] * matr[1][0];
}

SquareMatrix3Order::~SquareMatrix3Order() {
    delete[] matr;
};

double SquareMatrix3Order::getDeterminant() {
    return matr[0][0] * matr[1][1] * matr[2][2] + matr[0][1] * matr[1][2] * matr[2][0] + matr[0][2] * matr[1][0]
      * matr[2][1] - matr[0][2] * matr[1][1] * matr[2][0] - matr[0][1] * matr[1][0] * matr[2][2] - matr[0][0] *
      matr[1][2] * matr[2][1];
}
```

```
Matrix A:
    -9   -43   -49
    -4     7    20
    29   -46    14

Matrix B:
   -21    10
    18     5

Det A: -35579
Det B: -285
Sum of elements Matrix A: -81

S = -35945
Для продолжения нажмите любую клавишу . . .
```

Python

```python
import classes

matrixA = classes.SquareMatrix3Order()
matrixB = classes.SquareMatrix2Order()

print("\nMatrix A:")
matrixA.printMatrix()
print("\nMatrix B:")
matrixB.printMatrix()

a = matrixA.getDeterminant()
b = matrixB.getDeterminant()
sumA = matrixA.getSumOfElements()
s = a + b + sumA

print(f"\nDeterminant Matrix A = {a}")
print(f"Determinant Matrix B = {b}")
print(f"Sum of elements Matrix A: {sumA}")
print(f"\nS = {s}")
```

```python
from random import randint


class TMatrix:

    _matr = []

    def __init__(self, n, m):
        self._n = n
        self._m = m
        for i in range(n):
            line = []
            for j in range(m):
                line.append(randint(-50, 50))
            self._matr.append(line)

    def getSumOfElements(self):
        sumElem = 0
        for i in range(self._n):
            for j in range(self._m):
                sumElem += self._matr[i][j]
        return sumElem

    def printMatrix(self):
        for i in range(self._n):
            for j in range(self._m):
                print('{:4d}'.format(self._matr[i][j]), end=' ')
            print()

    def getDeterminant(self):
        raise NotImplementedError("In subclass method getDeterminant() must be redefine!")
```

```python
class SquareMatrix2Order(TMatrix):
    def __init__(self):
        super().__init__(2, 2)

    def getDeterminant(self):
        return super()._matr[0][0] * super()._matr[1][1] - super()._matr[0][1] * super()._matr[1][0]


class SquareMatrix3Order(TMatrix):
    def __init__(self):
        super().__init__(3, 3)

    def getDeterminant(self):
        return super()._matr[0][0] * super()._matr[1][1] * super()._matr[2][2] + super()._matr[0][1] * super()._matr[1][2] * super()._matr[2][0] + super()._matr[0][2] * super()._matr[1][0] * super()._matr[2][1] - super()._matr[0][2] * super()._matr[1][1] * super()._matr[2][0] - super()._matr[0][1] * super()._matr[1][0] * super()._matr[2][2] - super()._matr[0][0] * super()._matr[1][2] * super()._matr[2][1]
```

```
Matrix A:
 -49   -2    0
  -2  -29   -3
  10   34  -28

Matrix B:
 -49   -2
  -2  -29

Determinant Matrix A = -44614
Determinant Matrix B = 1417
Sum of elements Matrix A: -69

S = -43266

Process finished with exit code 0
```