

Додаток 1
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»
«Успадкування та поліморфізм»

Варіант 12

Виконав студент: ІП-13 Дойчев Костянтин Миколайович

Перевірила: Вечерковська Анастасія Сергіївна

Київ 2021

Лабораторна робота №4

Тема: Успадкування та поліморфізм

Постановка задачі

12. Спроектувати клас TEquation, який представляє рівняння і містить віртуальні методи для знаходження коренів рівняння та перевірки, чи є деяке значення коренем рівняння. На основі цього класу створити класи-нащадки, які представляють лінійні та квадратні рівняння. Створити n лінійних рівнянь та m квадратних рівнянь, згенерувавши дані для них випадковим чином. Знайти суму коренів для кожного із видів рівнянь (за умови, що вони існують). Перевірити, чи є задане значення коренем вказаного рівняння

Код:

Файл - main.cpp

```
#include <iostream>
#include <vector>
#include "LinearEquation.h"
#include "QuadraticEquation.h"

using namespace std;

int main() {
    srand(time(NULL));
    int n, m;
    cout << "Number of linear equations: ";
    cin >> n;

    cout << "Number of quadratic equations: ";
    cin >> m;

    int minCoeff, maxCoeff;

    cout << "Minimal coefficient: ";
    cin >> minCoeff;
```

```

cout << "Maximal coefficient: ";
cin >> maxCoeff;

vector<LinearEquation> linearEquations =
LinearEquation::generateRandomEquation(minCoeff, maxCoeff, n);
vector<QuadraticEquation> quadraticEquations =
QuadraticEquation::generateRandomEquation(minCoeff, maxCoeff, m);

for (int i = 0; i < n; i++) {
    cout << "\nLinear equation " << i + 1 << ": " << endl;
    linearEquations[i].print();
}
cout << '\n';

for (int i = 0; i < m; i++) {
    cout << "\nQuadratic equation " << i + 1 << ": " << endl;
    quadraticEquations[i].print();
}

double linearSum = 0;
for (int i = 0; i < linearEquations.size(); ++i) {
    linearSum += linearEquations[i].calculateRootsSum();
}

double quadraticSum = 0;
for (int i = 0; i < quadraticEquations.size(); ++i) {
    quadraticSum += quadraticEquations[i].calculateRootsSum();
}

cout << endl;

cout << "Linear equations sum: " << linearSum << endl;
cout << "Quadratic equations sum: " << quadraticSum << endl;

double possibleRoot = 0;
cout << "Possible root: ";
cin >> possibleRoot;

char answer;
cout << "Check linear equation? (y/n): ";
cin >> answer;

bool isLinear = answer == 'y';
int equationIndex = 0;
cout << "Equation index: ";
cin >> equationIndex;

bool isRootCorrect = false;

```

```

    if (isLinear) {
        for (int i = 0; i < linearEquations.size(); ++i) {
            if (i == equationIndex && linearEquations[i].isRoot(possibleRoot))
            {
                isRootCorrect = true;
                break;
            }
        }
    } else {
        for (int i = 0; i < quadraticEquations.size(); ++i) {
            if (i == equationIndex &&
quadraticEquations[i].isRoot(possibleRoot)) {
                isRootCorrect = true;
                break;
            }
        }
    }

    cout << (isRootCorrect ? "Root is correct" : "Root is incorrect") << endl;

    return 0;
}

```

Файл: TEquation.h

```
1  #ifndef LAB_4_TEQUATION_H
2  #define LAB_4_TEQUATION_H
3
4  #include <iostream>
5  #include <vector>
6
7  class TEquation {
8  public:
9      TEquation();
10     TEquation(std::vector<double> coefficients, int coeffsNum);
11     std::vector<double> getCoefficients();
12     virtual std::vector<double> findRoots();
13     std::vector<double> getRoots();
14     int getNumberOfCoefficients();
15     bool isRoot(double root);
16     double calculateRootsSum();
17     void print();
18 protected:
19     std::vector<double> roots;
20     int numberOfCoefficients{0};
21     bool areCoefficientsValid(std::vector<double> coefficients);
22     bool areCoefficientsValid(std::vector<double> coeffs, int coeffsNum);
23 private:
24     std::vector<double> coefficients;
25
26 };
27
28
29 #endif //LAB_4_TEQUATION_H
```

TEquation.cpp

```
#include "TEquation.h"

TEquation::TEquation(): coefficients(std::vector<double>()) {}

TEquation::TEquation(std::vector<double> coeff, int coeffsNum) {
    if (this->areCoefficientsValid(coeff, coeffsNum)) {
        this->coefficients = coeff;
    } else {
        throw "Invalid coefficients";
    }
}

bool TEquation::areCoefficientsValid(std::vector<double> coeffs) {
```

```

    bool isValidLength = coeffs.size() == this->getNumberOfCoefficients();
    return isValidLength && coefficients[0] != 0;
}

bool TEquation::areCoefficientsValid(std::vector<double> coeffs, int coeffsNum)
{
    bool isFirstValid = coeffs[0] != 0;
    bool isValidLength = coeffs.size() == coeffsNum;
    return isValidLength && isFirstValid;
}

std::vector<double> TEquation::findRoots() {
    return std::vector<double>();
}

std::vector<double> TEquation::getRoots() {
    return this->roots;
}

std::vector<double> TEquation::getCoefficients() {
    return this->coefficients;
}

int TEquation::getNumberOfCoefficients() {
    return this->numberOfCoefficients;
}

bool TEquation::isRoot(double root) {
    if (this->roots.size() == 0) {
        return false;
    }

    for (auto r : this->roots) {
        if (r == root) {
            return true;
        }
    }

    return false;
}

double TEquation::calculateRootsSum() {
    double sum = 0;
    for (auto r : this->roots) {
        sum += r;
    }
    return sum;
}

void TEquation::print() {

```

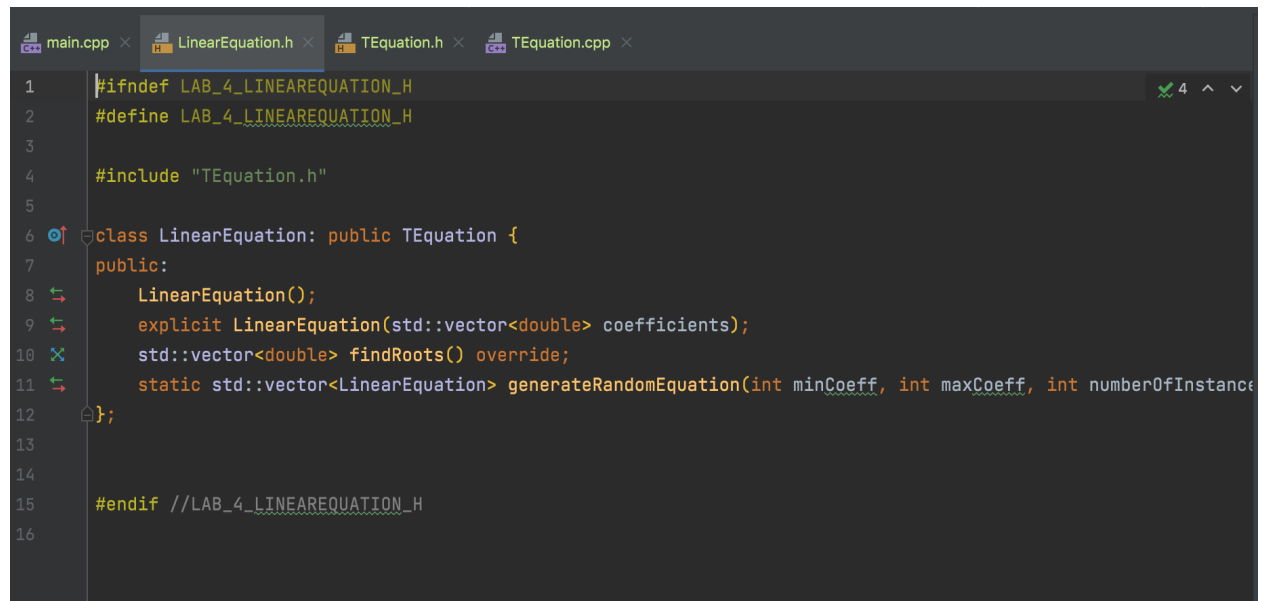
```

std::cout << "Coefficients: [";
for (auto c: this->coefficients) {
    std::cout << c << " ";
}
std::cout << "]" << std::endl;

std::cout << "Roots: [";
for (auto r: this->roots) {
    std::cout << r << " ";
}
std::cout << "]" << std::endl;
};

```

LinearEquation.h



```

1  #ifndef LAB_4_LINEAREQUATION_H
2  #define LAB_4_LINEAREQUATION_H
3
4  #include "TEquation.h"
5
6  class LinearEquation: public TEquation {
7  public:
8      LinearEquation();
9      explicit LinearEquation(std::vector<double> coefficients);
10     std::vector<double> findRoots() override;
11     static std::vector<LinearEquation> generateRandomEquation(int minCoeff, int maxCoeff, int numberOfInstances);
12 };
13
14
15 #endif //LAB_4_LINEAREQUATION_H
16

```

LinearEquation.cpp

```

#include "LinearEquation.h"

LinearEquation::LinearEquation() {
    this->numberOfCoefficients = 2;
};

LinearEquation::LinearEquation(std::vector<double> coeffs) : TEquation(coeffs,
2) {
    this->numberOfCoefficients = 2;
    this->findRoots();
};

std::vector<double> LinearEquation::findRoots() {

```

```

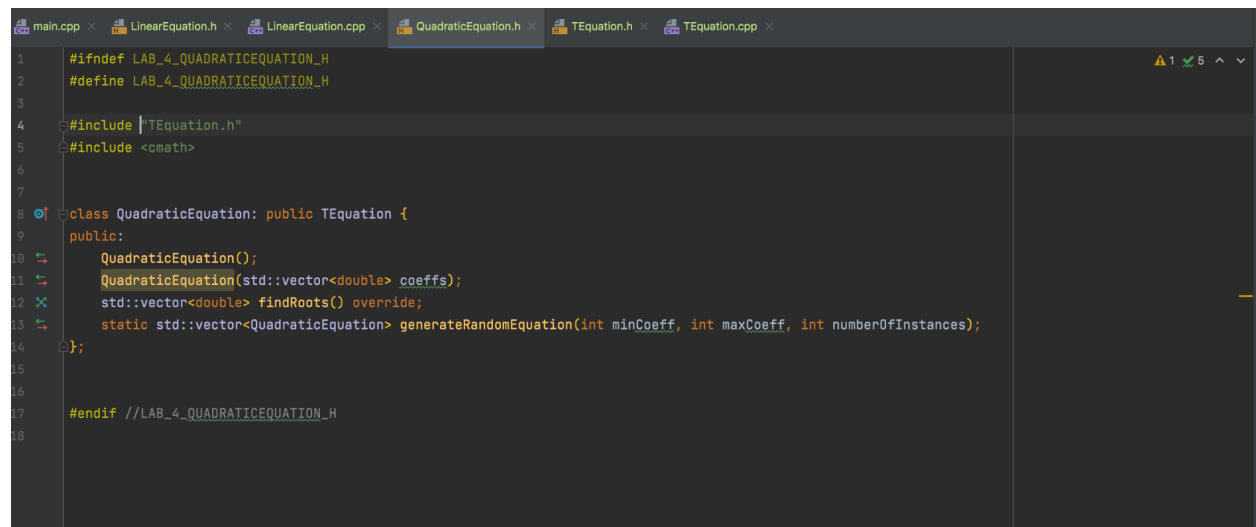
    auto coeffs = this->getCoefficients();
    double a = coeffs[0];
    double b = coeffs[1];
    std::vector<double> root = {-b / a};
    this->roots = root;
    return this->roots;
}

std::vector<LinearEquation> LinearEquation::generateRandomEquation(int
minCoeff, int maxCoeff, int numberOfInstances) {
    std::vector<LinearEquation> equations;
    int numberOfCoefficients = 2;
    for (int i = 0; i < numberOfInstances; i++) {
        try {
            std::vector<double> coefficients;
            for (int j = 0; j < numberOfCoefficients; j++) {
                coefficients.push_back((rand() % (maxCoeff - minCoeff + 1) +
minCoeff));
            }
            auto eq = LinearEquation(coefficients);
            equations.push_back(eq);
        } catch (const char *msg) {
            std::cout << msg << std::endl;
        }
    }

    return equations;
}

```

QuadraticEquation.h



```

1  #ifndef LAB_4_QUADRATICEQUATION_H
2  #define LAB_4_QUADRATICEQUATION_H
3
4  #include "TEquation.h"
5  #include <cmath>
6
7
8  class QuadraticEquation: public TEquation {
9  public:
10     QuadraticEquation();
11     QuadraticEquation(std::vector<double> coeffs);
12     std::vector<double> findRoots() override;
13     static std::vector<QuadraticEquation> generateRandomEquation(int minCoeff, int maxCoeff, int numberOfInstances);
14 };
15
16
17 #endif //LAB_4_QUADRATICEQUATION_H
18

```


QuadraticEquation.cpp

```
#include "QuadraticEquation.h"

QuadraticEquation::QuadraticEquation() {
    this->numberOfCoefficients = 3;
}

QuadraticEquation::QuadraticEquation(std::vector<double> coeffs) :
TEquation(coeffs, 3) {
    this->numberOfCoefficients = 3;
    this->findRoots();
};

std::vector<double> QuadraticEquation::findRoots() {
    auto coeffs = this->getCoefficients();
    double a = coeffs[0];
    double b = coeffs[1];
    double c = coeffs[2];
    double discriminant = pow(b, 2) - 4 * a * c;

    if (discriminant < 0) {
        std::vector<double> root = {};
        this->roots = root;
    } else if (discriminant == 0) {
        std::vector<double> root = {-b / (2 * a)};
        this->roots = root;
    } else {
        double root1 = (-b + sqrt(discriminant)) / (2 * a);
        double root2 = (-b - sqrt(discriminant)) / (2 * a);
        std::vector<double> root = {root1, root2};
        this->roots = root;
    }
    return this->roots;
}

std::vector<QuadraticEquation>
QuadraticEquation::generateRandomEquation(int minCoeff, int maxCoeff, int
numberOfInstances) {
    std::vector<QuadraticEquation> equations;
    int numberOfCoefficients = 3;
    for (int i = 0; i < numberOfInstances; i++) {
        try {
            std::vector<double> coefficients;
            for (int j = 0; j < numberOfCoefficients; j++) {
                coefficients.push_back((rand() % (maxCoeff - minCoeff + 1) +
minCoeff));
            }
            auto eq = QuadraticEquation(coefficients);
        }
    }
}
```

```

        equations.push_back(eq);
    } catch (const char *msg) {
        std::cout << msg << std::endl;
    }
}

return equations;
}

```

Дані і консоль:

Number of linear equations: 3
 Number of quadratic equations: 3
 Minimal coefficient: -20
 Maximal coefficient: 20

Linear equation 1:
 Coefficients: [2 1]
 Roots: [-0.5]

Linear equation 2:
 Coefficients: [16 3]
 Roots: [-0.1875]

Linear equation 3:
 Coefficients: [-9 -10]
 Roots: [-1.11111]

Quadratic equation 1:
 Coefficients: [-11 6 -19]
 Roots: []

Quadratic equation 2:
 Coefficients: [-12 14 3]
 Roots: [-0.184962 1.35163]

Quadratic equation 3:

Coefficients: [-14 -3 5]

Roots: [-0.714286 0.5]

Linear equations sum: -1.79861

Quadratic equations sum: 0.952381

Possible root: -0.5

Check linear equation? (y/n): y

Equation index: 0

Root is correct

Process finished with exit code 0