

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра информационных систем**

**ОТЧЕТ**  
**по практической работе №1**  
**по дисциплине «Объектно-ориентированное программирование»**

Студенты гр. 8363

\_\_\_\_\_

Нерсисян А.С.

\_\_\_\_\_

Панфилович А.И.

Преподаватель

\_\_\_\_\_

Егоров С.С.

Санкт-Петербург

2021

## Задание на практическую работу

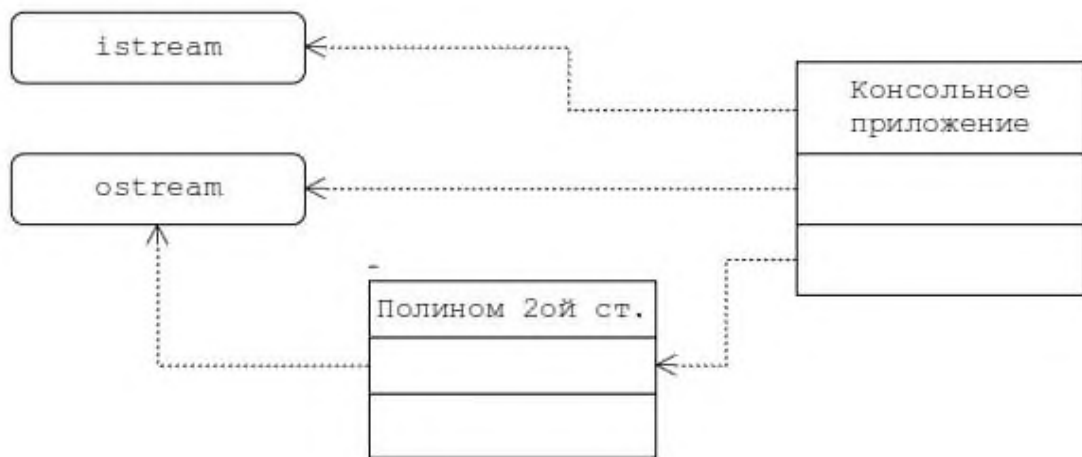


Рис.1. Диаграмма классов работы №1

Создать консольное приложение согласно представленной на рис.1 диаграмме классов, предназначенное для вычисления корней полинома 2-ой степени  $p(x) = a \cdot x^2 + b \cdot x + c$  ( $a \neq 0$ ) и его значения для заданного аргумента  $x$  на **множестве целых чисел**. Для этого необходимо специфицировать пользовательские классы "Консольное приложение" и "Полином 2ой степени". Т.е. задать атрибуты и методы указанных классов и распределить их по существующим областям видимости. Спецификация классов и реализация их методов должна обеспечивать реализацию отношений, указанных на диаграмме классов. **В отчете представить аргументированное обоснование своего выбора.**

Приложение должно включать основной модуль (функция `main`), модуль «`application`» и модуль «`polinom`».

В **основном модуле** консольного приложения (для языка C++ — это модуль с функцией `main`) должен создаваться объект класса "Консольное приложение" и вызываться его метод, который предоставляет пользователю **меню команд** приложения.

Модуль «**`application`**» должен содержать спецификацию класса "Консольное приложение" и реализацию его методов. Один из методов должен реализовывать меню команд приложения, включающее:

- команду, инициирующую ввод коэффициентов  $a$ ,  $b$ ,  $c$  (до ввода должны быть заданы значения по умолчанию);
- команду, инициирующую расчета корней полинома и вывод результатов расчета;
- команду, инициирующую ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод;
- команду, инициирующую вывод текстового представления полинома в указанной форме  $p(x)$ ;
- команду, инициирующую вывод текстового представления полинома в канонической форме;
- команду выхода из приложения.

Модуль «**polinom**» должен содержать спецификацию класса "Полином 2ой степени" и реализацию его методов, необходимых для реализации цели разрабатываемого приложения. Описание класса должно использовать вместо типа `double` (вещественное число, заданное в условии) абстрактный тип ***number***, описание которого должно задаваться в отдельном заголовочном файле `number.h` с помощью оператора **`typedef int number`** (для C++).

Требуется реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и протестировать на них программу. Оформить отчет, сделать выводы по работе.

\*) Следует обратить внимание на то, что класс "Полином 2ой степени", используемый в работах №1-№4, один и тот же, что является одним из правил ООП - понятие, описываемое как класс должно использоваться многократно и без изменений, если не требуется **дополнительной** детализации понятия и/или **усиления** его функциональности.

## Спецификации классов

### Класс **Application**

Атрибуты: нет атрибутов.

Модуль «**Application**» содержит спецификацию класса "Консольное приложение" и реализацию его методов.

Метод **Menu()** имеет тип возвращаемого значения `int`, не имеет формальных параметров, область видимости `private`, реализовывает меню команд приложения, включающее:

- команду, инициирующую ввод коэффициентов  $a$ ,  $b$ ,  $c$  (до ввода заданы значения по умолчанию 1);
- команду, инициирующую расчета корней полинома и вывод результатов расчета;
- команду, инициирующую ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод;
- команду, инициирующую вывод текстового представления полинома в указанной форме  $p(x)$ ;
- команду, инициирующую вывод текстового представления полинома в канонической форме;
- команду выхода из приложения

Метод **exec()** имеет тип возвращаемого значения `int`, не имеет формальных параметров, область видимости `public`, реализовывает обработку команд полученных из метода **Menu()**.

## Класс **Polinom**

Атрибуты:

Тип	Наименование	Область видимости
number	a	private
number	b	private
number	c	private
enum EPrintMode	printMode	private

Атрибуты a, b, c используется для хранения коэффициентов; printMode для хранения вида вывода на экран полинома  $P(x)$ .

Модуль «**Polinom**» содержит спецификацию класса "Полином 2-ой степени" и реализацию его методов.

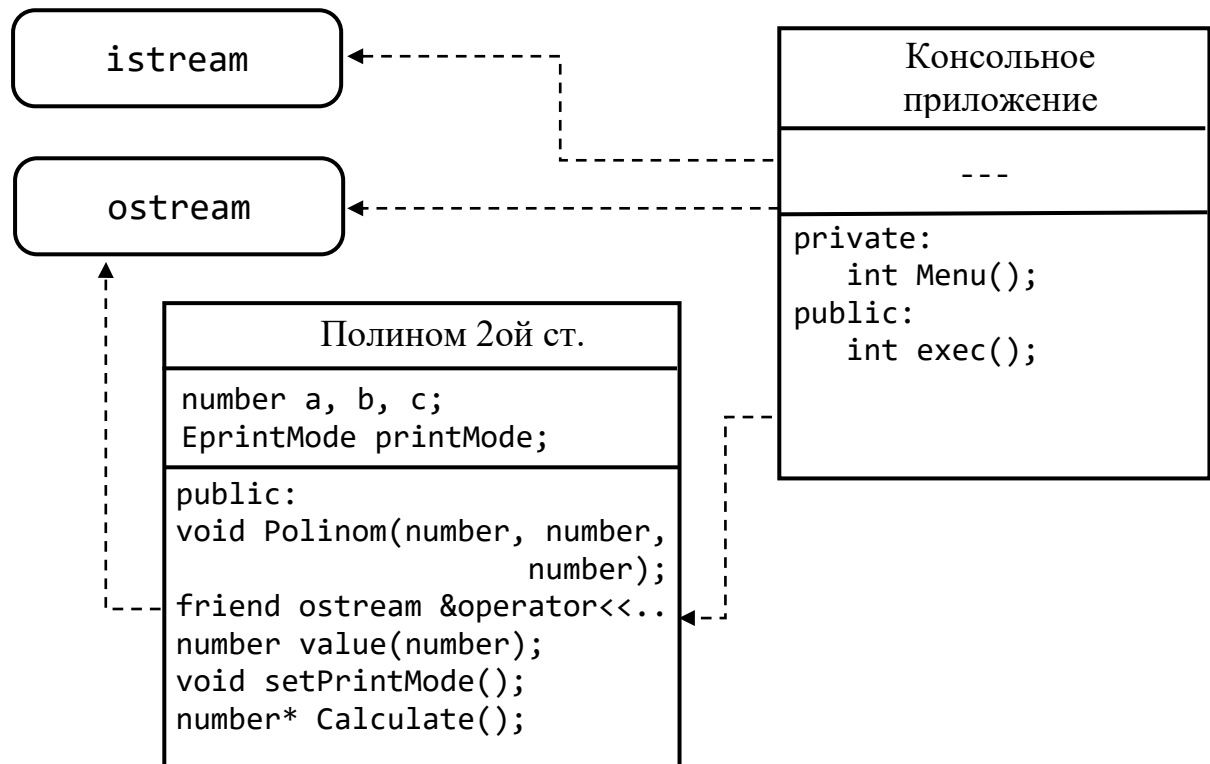
Конструктор **Polinom(number, number, number)** имеет 3 формальных параметров типа number, область видимости public, реализовывает создание объекта и задание значений коэффициентов полинома.

Метод **value(number)** имеет тип возвращаемого значения number, имеет 1 формальный параметр типа number, область видимости public, реализовывает расчет значения полинома по аргументу x.

Метод **Calculate()** имеет тип возвращаемого значения number\*, не имеет формальных параметров, область видимости public, реализовывает расчет и вывод корней полинома, а также возвращает массив корней.

Метод **setPrintMode(EPrintMode)** не возвращает никакие параметры (void), имеет один формальный параметр EPrintMode, область видимости public, задает вид полинома выводимого на экран.

## Диаграмма классов, дополненная атрибутами и методами



## Описание контрольного примера с исходными и ожидаемыми (расчетными) данными

Пример:  $P(x) = x^2 - 4x + 3$   $x_0 = 8$

$$a = 1, \quad b = -4, \quad c = 3, \quad x = 8$$

$$D = 16 - 12 = 4$$

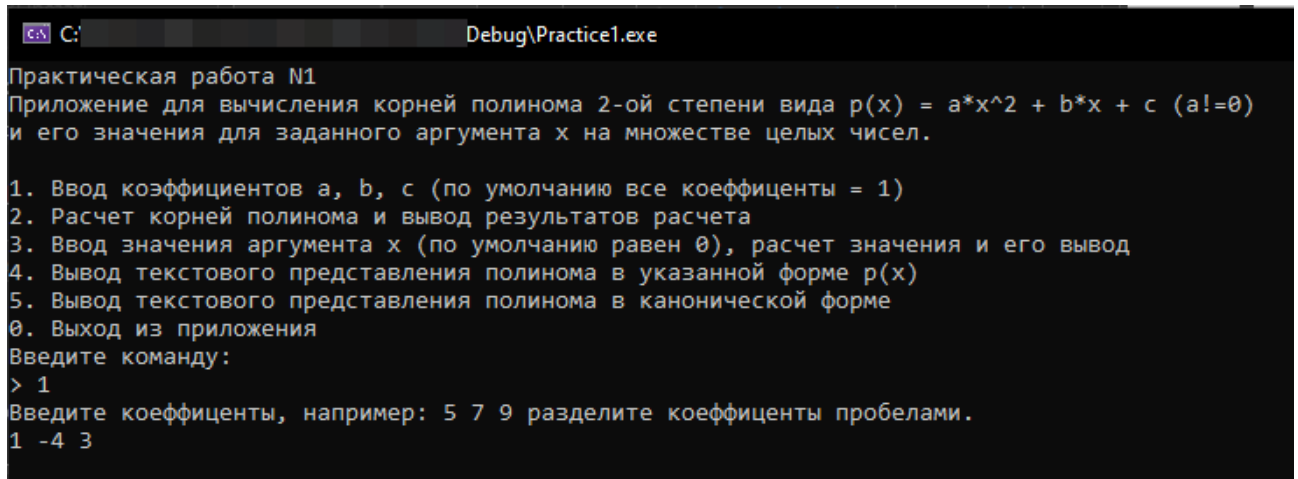
$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

$$x_1 = \frac{4 + 2}{2} = 3$$

$$x_2 = \frac{4 - 2}{2} = 1$$

$$P(9) = 8^2 - 4 * 8 + 3 = 64 - 32 + 3 = 35$$

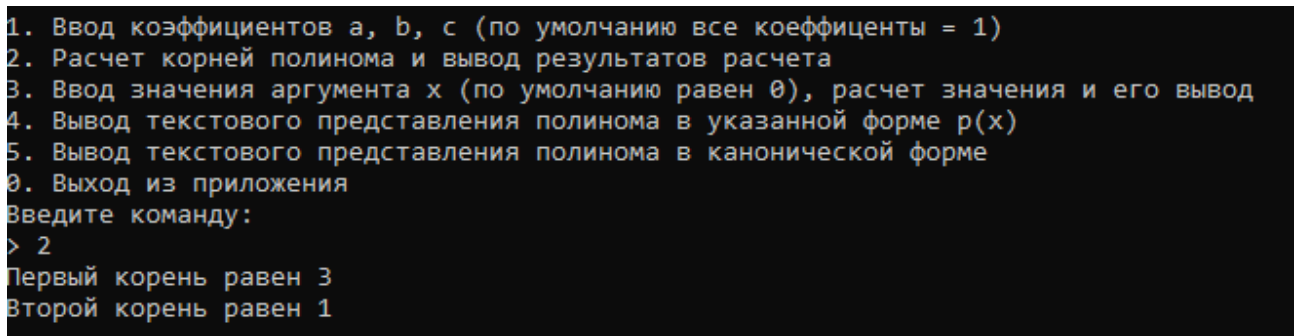
## Скриншоты программы на контрольных примерах



```
C:\ Debug\Practice1.exe
Практическая работа N1
Приложение для вычисления корней полинома 2-ой степени вида  $p(x) = a \cdot x^2 + b \cdot x + c$  ( $a \neq 0$ )
и его значения для заданного аргумента  $x$  на множестве целых чисел.

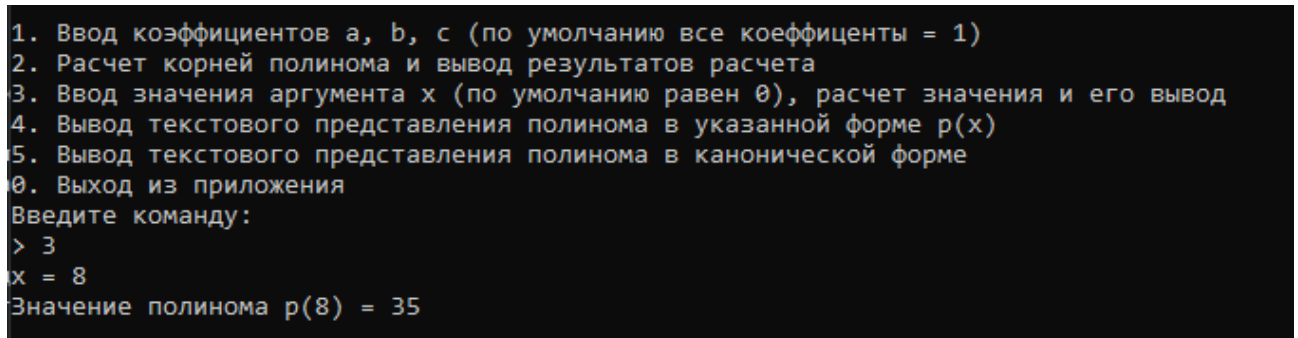
1. Ввод коэффициентов  $a, b, c$  (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме  $p(x)$ 
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 1
Введите коэффициенты, например: 5 7 9 разделите коэффициенты пробелами.
1 -4 3
```

Рисунок 1. Ввод значений коэффициентов



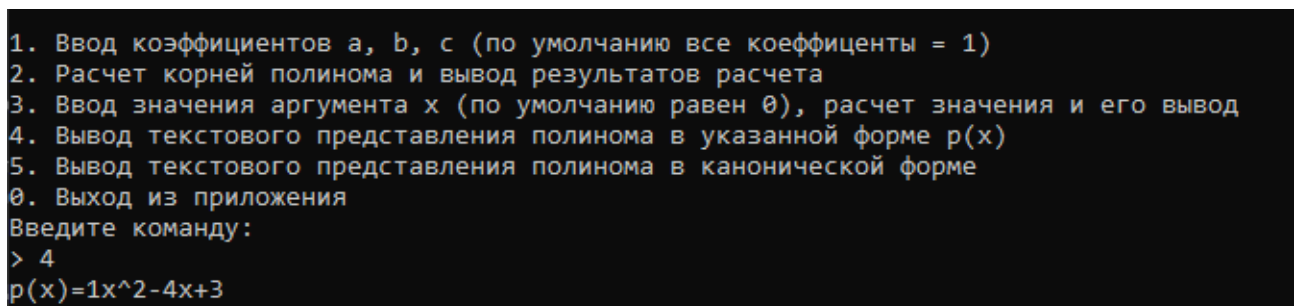
```
1. Ввод коэффициентов  $a, b, c$  (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме  $p(x)$ 
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 2
Первый корень равен 3
Второй корень равен 1
```

Рисунок 2. Расчет корней и вывод результатов расчета



```
1. Ввод коэффициентов  $a, b, c$  (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме  $p(x)$ 
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 3
 $x = 8$ 
Значение полинома  $p(8) = 35$ 
```

Рисунок 3. Ввод значения аргумента  $x$ , расчет значения и его вывод



```
1. Ввод коэффициентов  $a, b, c$  (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента  $x$  (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме  $p(x)$ 
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 4
 $p(x) = 1x^2 - 4x + 3$ 
```

Рисунок 4. Вывод текстового представления полинома в указанной форме  $p(x)$

```
1. Ввод коэффициентов a, b, c (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента x (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме p(x)
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 5
Первый корень равен 3
Второй корень равен 1
p(x)=(x-3)(x-1)
```

Рисунок 5. Вывод текстового представления полинома в канонической форме

```
1. Ввод коэффициентов a, b, c (по умолчанию все коэффициенты = 1)
2. Расчет корней полинома и вывод результатов расчета
3. Ввод значения аргумента x (по умолчанию равен 0), расчет значения и его вывод
4. Вывод текстового представления полинома в указанной форме p(x)
5. Вывод текстового представления полинома в канонической форме
0. Выход из приложения
Введите команду:
> 0
```

Рисунок 6. Выход из приложения

### **Выводы по выполнению работы**

В рамках данной практической работы была реализована и отлажена программа, удовлетворяющая сформулированным требованиям и заявленным целям. Разработаны контрольные примеры, и программа оттестирована на них.



## ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ПРОГРАММЫ

### FILE number.h

```
#ifndef NUMBER_H
#define NUMBER_H

typedef int number;

#endif
```

### FILE application.h

```
#include <iostream>
#include "number.h"
#include "polinom.h"

#ifndef APPLICATION_H
#define APPLICATION_H

class Application
{
private:
    int Menu();
public:
    int exec();
};

#endif
```

### FILE application.cpp

```
#include "application.h"

int Application::Menu()
{
    int ch;
    std::cout << std::endl << std::endl <<
        "1. Ввод коэффициентов a, b, c (по умолчанию все коэффициенты = 1)" << std::endl <<
        "2. Расчет корней полинома и вывод результатов расчета" <<
    std::endl <<
        "3. Ввод значения аргумента x (по умолчанию равен 0), расчет значения и его вывод" << std::endl <<
        "4. Вывод текстового представления полинома в указанной форме p(x)" << std::endl <<
```

```

        "5. Вывод текстового представления полинома в канонической форме"
<< std::endl <<
        "0. Выход из приложения" << std::endl << "Введите команду:" <<
std::endl << "> ";
        std::cin >> ch;
        return ch;
};

int Application::exec()
{
    setlocale(LC_ALL, "Russian");
    std::cout << "Практическая работа N1" << std::endl <<
        "Приложение для вычисления корней полинома 2-ой степени " <<
        "вида  $p(x) = a \cdot x^2 + b \cdot x + c$  ( $a \neq 0$ )" << std::endl <<
        "и его значения для заданного аргумента  $x$  " <<
        "на множестве целых чисел.";

    int ch;
    number a = 1, b = -4, c = 3, x1, x2;
    while (true)
    {
        ch = Menu();
        switch (ch)
        {
            case 0:
                return 0;
            case 1:
                std::cout << "Введите коэффициенты, например: 5 7 9 " <<
                    "разделите коэффициенты пробелами." << std::endl;
                std::cin >> a >> b >> c;
                break;
            case 2: {
                Polinom p(a, b, c);
                p.Calculate();
                break;
            }
            case 3: {
                number x = 0;
                std::cout << "x = ";
                std::cin >> x;
                Polinom p(a, b, c);
                number v = p.value(x);
                std::cout << "Значение полинома  $p(" << x << ") = " << v <<
std::endl;
                break;
            }
            case 4: {
                Polinom p(a, b, c);
                p.setPrintMode(EPrintModeClassic);
                std::cout << p << std::endl;
                break;
            }
        }
    }
}$ 
```

```

    }
    case 5: {
        Polinom p(a, b, c);
        p.setPrintMode(EprintModeCanonical);
        std::cout << p << std::endl;
        break;
    }
    default:
        std::cout << "Ошибка, неверный ввод" << std::endl;
        break;
    }
}
return 0;
};

```

## FILE polinom.h

```

#include <iostream>
#include "number.h"

#ifndef POLINOM_H
#define POLINOM_H

enum EPrintMode {
    EPrintModeClassic,
    EprintModeCanonical,
};

class Polinom
{
private:
    number a, b, c;
    EPrintMode printMode;
public:
    Polinom(number, number, number);
    friend std::ostream& operator<< (std::ostream&, Polinom&);
    number value(number);
    void setPrintMode(EPrintMode);
    number* Calculate();
};

#endif

```

## FILE polinom.cpp

```

#include "Polinom.h"
#include "number.h"
#include "math.h"

```

```

Polinom::Polinom(number inputA, number inputB, number inputC)
{
    printMode = EPrintModeClassic;
    a = inputA;
    b = inputB;
    c = inputC;
};

std::ostream& operator<< (std::ostream& os, Polinom& p) {
    if (p.printMode == EPrintModeClassic)
    {
        os << "p(x)=" << p.a << "x^2" << (p.b >= 0 ? "+" : "-") <<
abs(p.b) <<
        "x" << (p.c >= 0 ? "+" : "-") << abs(p.c) << std::endl;
    }
    else
    {
        number* roots = p.Calculate();
        if (roots == nullptr) return os;
        //два корня
        if (roots[0] != NULL && roots[1] != NULL) {
            std::cout << "p(x)=" << "(x" << (roots[0] >= 0 ? "-" :
"+" ) << abs(roots[0]) <<
            ")(x" << (roots[1] >= 0 ? "-" : "+") <<
abs(roots[1]) << ")" << std::endl;
        }

        //один корень
        if (roots[0] != NULL && roots[1] == NULL) {
            std::cout << "p(x)=" << "(x" << (roots[0] >= 0 ? "-" :
"+" ) << abs(roots[0]) <<
            ")^2" << std::endl;
        }
    }
    return os;
};

number Polinom::value(number x)
{
    return a * pow(x, 2) + b*x + c;
};

void Polinom::setPrintMode(EPrintMode mode)
{
    printMode = mode;
};

number* Polinom::Calculate()
{
    number* roots = new number[2];
    //при D>0

```

```

number d = ((b * b) - (4 * a * c));
if (d > 0) //Если дискриминант больше 0
{
    roots[0] = ((-1*b + sqrt(d)) / (2*a));
    roots[1] = ((-1*b - sqrt(d)) / (2*a));

    if (a*roots[0]*roots[0] + b*roots[0] + c == 0 &&
        a * roots[1] * roots[1] + b * roots[1] + c == 0) {
        std::cout << "Первый корень равен " << roots[0] <<
std::endl;
        std::cout << "Второй корень равен " << roots[1] <<
std::endl;
        return roots;
    }
    else
    {
        std::cout << "Полином не разложим над полем целых" <<
std::endl;
        return NULL;
    }
}

//при D=0
if (d == 0)
{
    roots[0] = (-1 * b) / (2 * a);
    roots[1] = NULL;
    if (a * roots[0] * roots[0] + b * roots[0] + c == 0) {
        std::cout << "Корень равен " << roots[0] << std::endl;
        return roots;
    }
    else {
        std::cout << "Полином не разложим над полем целых" <<
std::endl;
        return NULL;
    }
}

//при D<0
else
{
    std::cout << "Полином не разложим над полем целых" <<
std::endl;
    return NULL;
}
};

```

### **FILE main.cpp**

```
//Object-Oriented Programming  
//Practice One
```

```
#include "application.h"
```

```
int main()  
{  
    Application app;  
    return app.exec();  
};
```

### **FILE makefile**

```
all : main
```

```
.PHONY : all clean
```

```
CC = g++
```

```
LD = g++
```

```
main : main.o application.o polinom.o
```

```
main.o: main.cpp application.h polinom.h number.h
```

```
application.o: application.cpp application.h
```

```
polinom.o: polinom.cpp polinom.h
```