

**Федеральное государственное автономное
образовательное учреждение высшего
образования**

**«Национальный исследовательский
университет ИТМО»**

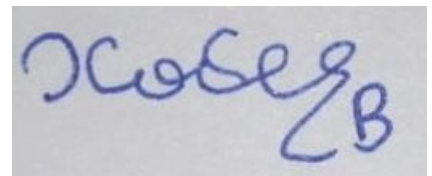
**Факультет информационных технологий
и Программирования**

Программирование

Лабораторная работа № 6

Выполнил студент группы № М3104
Хобер Владислав Алимович

Подпись:



Санкт-Петербург

2022

Лабораторная №6

Задача

1. Вам требуется реализовать возможность вычисления значения в точке многочлена с целочисленными коэффициентами. Точка также целочисленная. Все вычисления должны происходить в момент компиляции.
2. Перегрузить оператор вывода для многочлена из пункта 1.
3. Полученный код протестировать с помощью GoogleTest Framework

polynom_test.cc

```
#include <gtest/gtest.h>
#include "polynom.h"
TEST(PolynomTest, First) {

    constexpr int m[4] = {1, 2, 3, 4};
    constexpr Polynom<4> polynom(m, 4);
    constexpr int result = polynom.value(1);
    static_assert(result == 10, "Wrong");
    EXPECT_EQ(result, 10);
}
TEST(PolynomTest, Second) {

    constexpr int m[4] = {0, 0, 0, 0};
    constexpr Polynom<4> polynom(m, 4);
    constexpr int result = polynom.value(10);
    static_assert(result == 0, "Wrong");
    EXPECT_EQ(result, 0);
}
TEST(PolynomTest, Third) {

    constexpr int m[6] = {5, 3, 4, 6, 1, 1};
    constexpr Polynom<6> polynom(m, 6);
    constexpr int result = polynom.value(0);
    static_assert(result == 5, "Wrong");
    EXPECT_EQ(result, 5);
}
TEST(PolynomTest, Fourth) {

    constexpr int m[6] = {5, 3, 4, 6, 1, 1};
    constexpr Polynom<6> polynom(m, 6);
    constexpr int result = polynom.value(2);
    static_assert(result == 123, "Wrong");
    EXPECT_EQ(result, 123);
}
```

polynom.h

```

#ifdef TEST_20_POLYNOM_H
#define TEST_20_POLYNOM_H
template<size_t power>
class Polynom
{
public:
    constexpr Polynom() {}
    constexpr Polynom(const int* array, const size_t size)
    {
        add(array, size - 1);
    }
    constexpr int operator[](const size_t position) const
    {
        return coeff[position];
    }
    constexpr int value(const int point) const
    {
        constexpr int point_degree = 1;
        constexpr int position = 0;
        return calculating_value(point, point_degree * point, position + 1)
+ coeff[0];
    }
    friend std::ostream &operator<<(std::ostream &out, const Polynom
&polynom)
    {
        int val_out;
        for (int i = 0; i < power; i++)
        {
            val_out = polynom.coeff[i];
            if (i != 0)
            {
                if (val_out < 0)
                {
                    out << '-' << ' ';
                    val_out = -val_out;
                }
                else
                {
                    out << '+' << ' ';
                }
            }
            out << val_out << ' ' << '*' << ' ' << "x^" << i << ' ';
        }
        return out;
    }
private:
    constexpr void add(const int* array, const size_t position)
    {
        if (position == 0)
        {
            coeff[position] = array[0];
        }
    }

```

```

    }
    else
    {
        coeff[position] = array[position];
        add(array, position - 1);
    }
}
constexpr int calculating_value(const int point, const int
point_degree, const int position) const
{
    if (position == power - 1)
        return point_degree * coeff[position];
    return calculating_value(point, point * point_degree, position +
1) + point_degree * coeff[position];
}
int coeff[power];
};
#endif

```

Ввод и вывод

Программа тестируется с помощью GoogleTest Framework.

Вывод

Используя принципы ООП, я выполнил предложенное мне задание, понял, каким образом реализовывать вычисления в compile-time.