Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Программирование

Лабораторная работа №6

Programming at compile-time.

Выполнил студент группы № M3109
Гумбатов Владислав Юрьевич
Подпись:

Санкт-Петербург
2022

Задача

Целью лабораторной работы является написание программы, вычисляющую значение в точке многочлена с целочисленными коэффициентами. Все вычисления должны происходить в момент компиляции. Также необходимо протестировать программу с помощью GoogleTest Framework.

Входные данные:

Входными данными в программе является полином и значение точки многочлена.

Выходные данные:

В качестве выходных данных программа выдает значение в точке многочлена.

Программа была написана с использованием последнего стандарта языка (C++20), поэтому в ней используется вектор, а также consteval функции, гарантирующие выполнение программы в compile time (так как они не могут работать в run-time).

Код:

```
#include "pch.h"

#include <array>
#include <algorithm>
#include <iostream>
#include <numeric>
#include <string>
#include <vector>
#include <array>
#include <math.h>



TEST(TestCaseName, TestName) {
  EXPECT_EQ(1, 1);
  EXPECT_TRUE(true);
}


consteval int vala(const int b) {

    if constexpr (std::is_constant_evaluated()) {



        constexpr int a = 1;

        return 1;
    }


    else {


        return 0;
    }


}

void print(std::vector<std::pair<int, int>> &arr, int constantvar) {



    for (auto [num, dig] : arr)

        if (num < 0) {

            //std::cout << "(" << num << ")" << "x^" << dig << std::showpos;

            if (dig == 1) {

                printf("%s%d%s%s", "(", num, ")x"," + ");
```

```
            continue;
        }
        printf("%s%d%s%d%s", "(", num ,")x^",dig," + ");
    }
    else {
        if (dig == 1) {
            printf("%s%d%s%s", "", num, "x", " + ");
            continue;
        }
        //std::cout << num << "x^ " << dig << std::showpos;
        printf("%s%d%s%d%s", "", num, "x^", dig, " + ");
    }
    std::cout << constantvar << std::endl;

}


template<typename T = int, typename U = int>
consteval T power(T number, U digit) {
    std::vector<int> arr(digit-1, 1);
    int sum = 0;
    int dig = number;
    for (auto poww : arr)
        number =  (number * dig);

    return number;
}
template<typename T = int>
consteval T calca(T N, const bool fl) {

    if (std::is_constant_evaluated()) {

        std::vector<std::pair<int, int>> arr{ {2,2}, {-3,1}, {5,1} };
```

```
        int constantvar = 20;


        int sum = 0;

        for (auto [num, dig] : arr)

            sum = sum + (num * (power(N, dig)));



        if (std::is_constant_evaluated()) {

            return sum + constantvar;
        }
    }

    else {


    }

  return 0;


}

TEST(Test2, Test) {

    constexpr int bbb = calca(4, 0);

    constexpr int f = 60;

    ASSERT_EQ(f, bbb);
    EXPECT_TRUE(true);
}

TEST(Test3, Testing) {

    constexpr int bbb = calca(9, 0);

    constexpr int f = 200;

    ASSERT_EQ(f, bbb);
    EXPECT_TRUE(true);
}

TEST(Test4, Testing) {

    constexpr int bbb = calca(900, 0);

    constexpr int f = 1621820;

    ASSERT_EQ(f, bbb);
    EXPECT_TRUE(true);
```

```
    EXPECT_TRUE(true);
}

TEST(Test5, Testing) {

    constexpr int bbb = calca(1520, 0);

    constexpr int f = 4623860;

    ASSERT_EQ(f, bbb);
    EXPECT_TRUE(true);
}

TEST(Test6, Testing) {

    constexpr unsigned int bbb = calca(312899032, 0);

    constexpr unsigned int f = 195811609078872132;

    ASSERT_EQ(f, bbb);
    EXPECT_TRUE(true);
}


int main(int argc, char* argv[]) {


    std::vector<int> arr(3, 1);

    constexpr int multiplier[]{ 2,3,5 };

    constexpr std::array<int, 3> arra{ 2, 3, 5 };

    constexpr bool fl = 0;

    constexpr int bbb = calca(4, fl);


    std::cout << bbb << std::endl << std::endl;

    constexpr unsigned int aaa = calca(31289, 0);


    std::cout << aaa << std::endl << std::endl;


    std::vector<std::pair<int, int>> poly{ {2,2}, {-3,1}, {5,1} };

    print(poly, 20);

    std::cout << std::endl;
```

```
    std::cout << std::endl;


    ::testing::InitGoogleTest(&argc, argv);

    RUN_ALL_TESTS();

    return 0;
}
```

Вывод:

Программа успешно работает. Программа была написана с использованием C++20 стандарта языка, в частности все функции написаны с использованием consteval, что позволяет гарантировать, что все вычисления были сделаны в compile-time. Также было написано 6 тестов, и все из них были успешно пройдены. Цель лабораторной работы достигнута.