Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

**Кафедра вычислительных машин систем и сетей**

Лабораторная работа №6

«**Исследование эффективности высокоточных вычислений с плавающей точкой**»

Выполнили:

студенты группы А-07м-23

Рогов Д.Р.  
Балашов С.А.

**Москва, 2023**

**Вариант 2.**

Разработать программу для реализации высокоточных вычислений. Проверить работу программы на примере вычисления скалярного произведения с использованием представленной выше арифметики и сравнить по точности с арифметикой с плавающей двойной точности.

Вычислим скалярное произведение:

,, *\**,,

Результат работы программы:

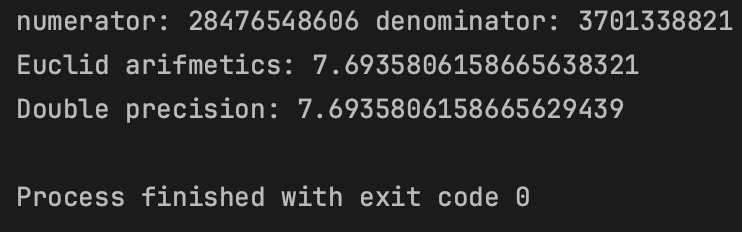


Рис. 1. Результат работы программы

**Листинг**

**main.cpp**

#include "Euclid.h"

using namespace euclid;

int main() {

vector<Euclid> euclids = { Euclid(371, 243), Euclid(26, 17), Euclid(31, 43) };

vector<Euclid> euclids2 = { Euclid(55, 67), Euclid(113, 27), Euclid(17, 311) };

Euclid result(0, 1);

double dResult = 0;

for (int i = 0; i < euclids.size(); i++) {

result = result + euclids[i] \* euclids2[i];

auto da = double(euclids[i].getNumerator()) / double(euclids[i].getDenominator());

auto db = double(euclids2[i].getNumerator()) / double(euclids2[i].getDenominator());

dResult += da \* db;

}

cout << setprecision(20);

cout << result << endl;

cout << double(result.getNumerator()) / double(result.getDenominator()) << endl;

cout << dResult << endl;

return 0;

}

**Euclid.h**

#ifndef LAB6\_EUCLID\_H

#define LAB6\_EUCLID\_H

#include <ostream>

#include <iostream>

#include "iomanip"

namespace euclid {

using namespace std;

class Euclid {

public:

Euclid(long long numerator, long long denominator);

Euclid();

~Euclid();

long long getNumerator() const;

long long getDenominator() const;

friend std::ostream& operator<<(std::ostream& os, const Euclid& euclid1);

Euclid operator+(const Euclid& euclid2) const;

Euclid operator-(const Euclid& euclid2) const;

Euclid operator\*(const Euclid& euclid2) const;

Euclid operator/(const Euclid& euclid2) const;

private:

long long numerator;

long long denominator;

pair<long long, long long> calc(long long b2, long long b1, long long u2, long long u1, long long v2, long long v1) const;

};

} // euclid

**Euclid.cpp**

#include "Euclid.h"

namespace euclid {

Euclid::Euclid(long long numerator, long long denominator) : numerator(numerator), denominator(denominator)

{

}

Euclid::Euclid() : numerator(0), denominator(0)

{

}

Euclid::~Euclid() {

}

std::ostream& operator<<(std::ostream& os, const Euclid& euclid1) {

os << "numerator: " << euclid1.numerator << " denominator: " << euclid1.denominator;

return os;

}

Euclid Euclid::operator+(const Euclid& euclid2) const {

pair<long long, long long> result = calc(numerator, denominator, euclid2.numerator, euclid2.denominator, euclid2.denominator, 0);

return Euclid(result.first, result.second);

}

Euclid Euclid::operator-(const Euclid& euclid2) const {

pair<long long, long long> result = calc(numerator, denominator, -euclid2.numerator, euclid2.denominator, euclid2.denominator, 0);

return Euclid(result.first, result.second);

}

Euclid Euclid::operator\*(const Euclid& euclid2) const {

pair<long long, long long> result = calc(numerator, denominator, 0, euclid2.numerator, euclid2.denominator, 0);

return Euclid(result.first, result.second);

}

Euclid Euclid::operator/(const Euclid& euclid2) const {

pair<long long, long long> result = calc(numerator, denominator, 0, euclid2.denominator, euclid2.numerator, 0);

return Euclid(result.first, result.second);

}

pair<long long, long long> Euclid::calc(long long b2, long long b1, long long u2, long long u1, long long v2, long long v1) const

{

long long a = b2 / b1;

long long b = b2 % b1;

long long u = u1 \* a + u2;

long long v = v1 \* a + v2;

if (b == 0)

{

return { u, v };

}

return calc(b1, b, u1, u, v1, v);

}

long long Euclid::getNumerator() const {

return numerator;

}

long long Euclid::getDenominator() const {

return denominator;

}

}