g***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №4**

**По дисциплине КМЗИ за IV семестр**

**Тема: «Факторизация чисел»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Хацкевич М. В.

Брест 2019

Цель работы: изучить методы факторизации чисел.

Ход работы

В данной лабораторной работе я реализую метод Ферма, а также решето Эратосфена и нахождение простых делителей методом подстановок.

Листинги:

1. Нахождение простых делителей

std::vector<\_\_int64> TrialDivision(\_\_int64 n)

{

std::vector<\_\_int64> vecRes;

int f = 2;

while (n > 1)

{

if (n % f == 0)

{

vecRes.push\_back(f);

n /= f;

}

else

f++;

}

return vecRes;

}

1. Решето Эратосфена

std::vector<int> SieveOfEratosthenes(int n)

{

std::vector<int> vecSieve(n);

std::generate(vecSieve.begin(), vecSieve.end(), []() {

static int i = 1;

return i++;

});

int f = 2;

std::vector<bool> vecMask(vecSieve.size());

std::fill(vecMask.begin(), vecMask.end(), true);

while (true)

{

for (int i = f; i < vecSieve.size(); i++)

{

if (vecSieve[i] % f == 0)

vecMask[i] = false;

}

int i = f;

for (; i < vecSieve.size(); i++)

{

if (vecMask[i] == true)

{

f = vecSieve[i];

break;

}

}

if (i == vecSieve.size())

break;

}

for (int i = 0; i < vecSieve.size();)

{

if (vecMask[i] == false)

{

vecSieve.erase(vecSieve.begin() + i);

vecMask.erase(vecMask.begin() + i);

continue;

}

i++;

}

return vecSieve;

}

1. Факторизация методом Ферма

bool FermatFactorization(int n, int n0)

{

if (n % 2 == 0)

return false;

int a = ceil(sqrt(n));

for (int i = 0; i < n0; i++)

{

int substr = (a + i)\*(a + 1) - n;

int sr = sqrt(substr);

if (sr \* sr == substr)

return false;

}

return true;

}

1. Фрагмент main

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

{

if (argc != 2)

return -1;

\_\_int64 n = atof(argv[1]);

auto vecDiv = TrialDivision(n);

std::cout << "Prime deviders: ";

for (auto div : vecDiv)

std::cout << div << " ";

std::endl(std::cout);

auto vecPrimes = SieveOfEratosthenes(n);

std::cout << "Prime numbers until " << n << ": ";

for (auto prime : vecPrimes)

std::cout << prime << " ";

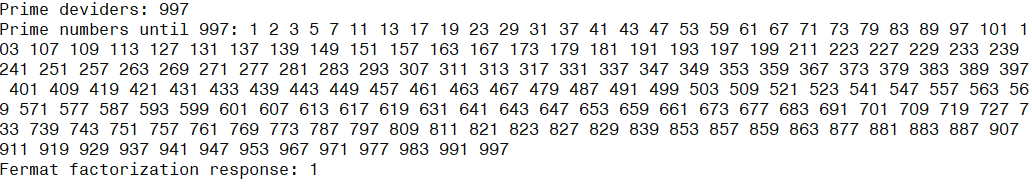
std::cout << std::endl;

std::cout << "Fermat factorization response: " << FermatFactorization(n, n\*n) << std::endl;

getchar();

}

Результаты работы:



Как видно из листинга реализованные алгоритмы работают.

Вывод: по ходу лабораторной работы научился производить проверку числа на простоту, находить простые делители.