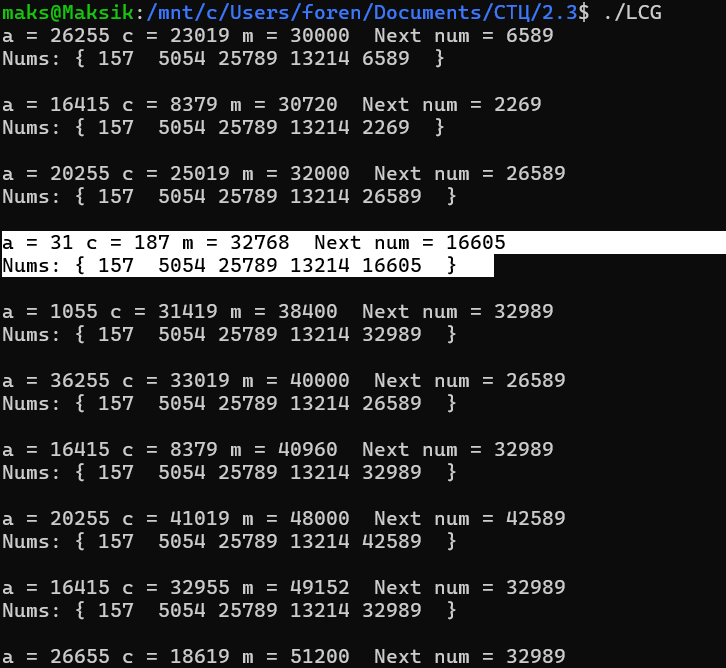
Для линейного конгруэнтного генератора чисел реализовать алгоритм вычисления последующего числа по четырем предыдущим. Считать параметр m не более 65535. Например, есть 4 числа : 157; 5054; 25789; 13214 Следующим за этими числами будет число 16605.



Код:

#include <iostream>

#include <vector>

#include <cmath>

#define AUTO\_FILL

constexpr int msize = 65535;

class LcgGenerator

{

public:

struct Params {

int a, c, m;

};

LcgGenerator(int a\_, int c\_, int m\_, int num\_)

: params{a\_, c\_, m\_}

, num(num\_)

{

}

int Generate()

{

num = (params.a \* num + params.c) % params.m;

return num;

}

int GetA() const { return params.a; }

int GetC() const { return params.c; }

int GetM() const { return params.m; }

int GetNum() const { return num; }

private:

Params params;

int num;

};

std::vector<LcgGenerator> GetLcgCandidates(std::vector<int> nums)

{

std::vector<long long> tn, un, divisors;

std::vector<std::pair<int, int>> coefs;

std::vector<LcgGenerator> gens;

for (int i = 0; i < nums.size() - 1; i++) {

tn.push\_back(nums[i + 1] - nums[i]);

}

for (int i = 0; i < tn.size() - 2; i++) {

long long un\_ = abs(tn[i + 2] \* tn[i] - std::pow(tn[i + 1], 2));

un.push\_back(un\_);

}

for (long long i = std::max(nums[1], std::max(nums[2], nums[3])) + 1; i <= msize; i++) {

if (un[0] % i == 0) {

divisors.push\_back(i);

}

}

for (auto m : divisors) {

for (int a = 0; a < m; a++) {

if (a \* (nums[0] - nums[1]) % m == nums[1] - nums[2]) {

coefs.push\_back(std::make\_pair(a, m));

}

}

}

for (const auto& am : coefs) {

for (int c = 0; c < am.second; c++) {

if ((am.first \* nums[0] + c) % am.second == nums[1]) {

gens.push\_back(LcgGenerator(am.first, c, am.second, nums[3]));

}

}

}

return gens;

}

int main()

{

std::vector<int> nums;

#ifdef AUTO\_FILL

nums = { 157, 5054, 25789, 13214 };

#else

nums.resize(4);

std::cin >> nums[0] >> nums[1] >> nums[2] >> nums[3];

#endif

auto gens = GetLcgCandidates(nums);

for (auto& gen : gens) {

std::cout << "a = " << gen.GetA() << " c = " << gen.GetC() << " m = " << gen.GetM() <<

" Next num = " << gen.Generate() << std::endl;

LcgGenerator ngen(gen.GetA(), gen.GetC(), gen.GetM(), nums[0]);

std::cout << "Nums: { " << nums[0] << " ";

for (int i = 0; i < 4; i++) {

std::cout << ngen.Generate() << " ";

}

std::cout << " } \n\n";

}

return 0;

}