МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем і мереж

Лабораторна робота №5

з дисципліни: «Екстремальне програмування»

Ключовий обмін Діффі-Хеллмана

|  |
| --- |
| Виконав:  Студент групи ЕКП-ІСМ-301  Гасій В. Ю.  Прийняв:  Щербак С. С. |

Львів – 2020

**Побудова початкового кореня по модулю n**

Згідно з теоремою Ейлера для будь-яких взаємно простих *а* та *n* виконується відношення:

*(1)*

де *μ(n)* визначає функцію Ейлера, значення якої рівне кількості додатнихцілих значень, менших *n* та взаємно простих з *n*. Для простого числа *p* виконується:

*(2)*

Якщо припустити, що два числа *p* і *q –* прості, тоді для *n=pαqβ* функціяЕйлера матиме вид:

*(3)*

Розглянемо більш загальне співвідношення, ніж (1). Визначають, щочисло *а*, є взаємно простим з модулем *n*, належить показнику m, якщо *m* – такенайменше натуральне число, що виконується порівняння:

*≡1(mod n) (4)*

Якщо *a* і *n* є взаємно простими, то існує, принаймні, одне число *m=μ(n)*,що задовольняє (4). Найменше з додатних чисел *m*, для яких виконується (4),називається довжиною періоду послідовності, що генерується ступенями *а*. Зогляду на те, справедливі наступні властивості.

**Властивість 1.** Числа *, , ...*, - попарно непорівнянні по модулю *n*.**Властивість 2.** **

**Властивість 3.** Число *а*, що належить показнику *μ(n)* називаєтьсяпочатковим коренем за модулем *n*

**Властивість 4.** За будь-яким простим модулем *p* існує початковий корінь. Початкові корені існують за модулями *2, 4, , 2*, де *р* – непарне просте число, а *α* є *N*.

**Властивість 5.** Нехай *c = μ(n)* і , , ..., - різні прості дільники числа *c*. Число *a*, взаємно просте з модулем *n*, буде початковим коренем тоді і лише тоді, коли не виконується жодне з наступних порівнянь:



**Завдання**

Виконати алгоритм побудови початкового кореня по модулю n.

**Розв’язання**

Main.cpp

#include "Include.h"

using boost::multiprecision::cpp\_int;

int main()

{

cpp\_int num = create\_prime\_number(25);

std::vector<cpp\_int> temp = initial\_roots(num, 10);

for (int i = 0; i < 10; i++)

{

std::cout << temp[i] << " ";

}

std::cout << std::endl;

system("pause");

return 0;

}

Initial\_Roots.cpp

#include "Initial\_Roots.h"

#include "Prime\_Number.h"

cpp\_int euler\_function(cpp\_int n)

{

cpp\_int result = n;

for (cpp\_int i = 2; i \* i <= n; i++)

{

if (n % i == 0)

{

while (n % i == 0)

n /= i;

result -= result / i;

}

}

if (n > 1)

result -= result / n;

return result;

}

std::vector<cpp\_int> initial\_roots(cpp\_int number, int amount)

{

std::vector<cpp\_int> div;

std::vector<cpp\_int> roots;

cpp\_int n = euler\_function(number);

cpp\_int temp = n;

for (int i = 2; i <= n; i++)

{

if (n%i == 0)

{

n /= 2;

div.push\_back(i);

i--;

}

}

std::sort(div.begin(), div.end());

div.erase(unique(div.begin(), div.end()), div.end());

n = temp;

cpp\_int counter = 2;

while (roots.size() < amount)

{

bool is\_root = true;

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

{

if (modular\_pow(counter, n / div[i], number) == 1)

{

is\_root = false;

}

}

if (counter >= number)

{

break;

}

if (is\_root)

{

roots.push\_back(counter);

}

counter++;

}

return roots;

}

Initial\_Roots.h

#pragma once

#include <iostream>

#include <cmath>

#include <boost/multiprecision/cpp\_int.hpp>

#include <ctime>

#include <vector>

using boost::multiprecision::cpp\_int;

cpp\_int euler\_function(cpp\_int n);

std::vector<cpp\_int> initial\_roots(cpp\_int number, int amount);

Include.h

#pragma once

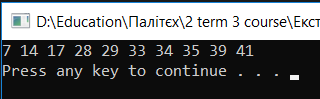
#include <iostream>

#include <boost/multiprecision/cpp\_int.hpp>

#include <vector>

#include "Prime\_Number.h"

#include "Initial\_Roots.h"



**Висновок**: на цій лабораторній роботі було виконано алгоритм побудови початкового кореня по модулю n.