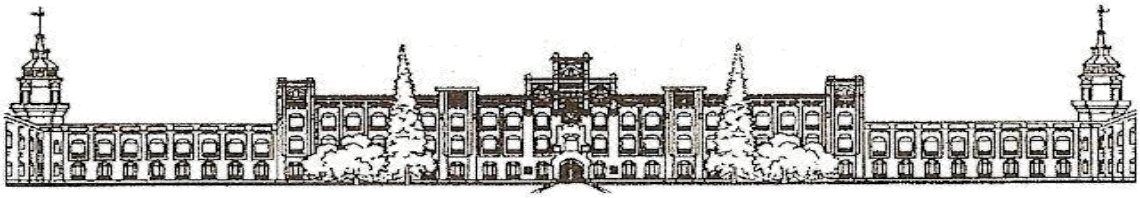

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

КОМП’ЮТЕРНИЙ ПРАКТИКУМ З ДИСЦИПЛІНИ «МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ» №1

Вибір та реалізація базових фреймворків та бібліотек

Виконала:
студентка групи ФІ-12мн
Звичайна Анастасія Олександрівна

Перевірила:
студентка групи ФІ-12мн
Селюх Поліна Валентинівна

Київ 2021

Мета роботи: Вибір базових бібліотек/сервісів для подальшої реалізації криптосистеми.

Умова задачі: дослідити алгоритми реалізації арифметичних операцій над великими (багато розрядними) числами над скінченими полями та групами з точки зору їх ефективності за часом та пам’яттю для різних програмно-апаратних середовищ, а саме дослідити бібліотеки багаторозрядної арифметики, вбудовані в програмні платформи C++/C# (BigInteger), Java (BigInt) та Python (обрати одну з них) для процесорів із 32-розрядною архітектурою та обсягом оперативної пам’яті до 8 ГБ (робочі станції).

Хід роботи та необхідна теорія:

Не секрет, що C та C++ частіше за інші мови використовувались у розробці апаратних та системних програм саме через їх швидкодію. Більш того, популярність цих мов буде тільки рости, адже сьогодні відбувається стрімкий розвиток та впровадження у життя малоресурсних пристроїв, а також технологій, що об’єднують їх у єдину мережу (WSN, IoT). Таким чином, вибір бібліотеки BigInteger для розробки C++ додатків як предмета дослідження даного комп’ютерного практикуму є цілком обґрунтованим.

Для роботи з обраною бібліотекою треба розробити проект у фрейморку .NET – інтегрованій компоненті Windows, яка містить середовище CLR (Common Language Runtime), що є реалізацією компанією Microsoft інфраструктури CLI (Common Language Infrastructure). Тобто головними особливостями .NET є можливість взаємодії програмних реалізацій, що написані на різних мовах програмування; повна інтеграція з Windows платформою, а з 12 квітня 2010 року ще й можливість роботи з багаторозрядними знаковими числами. Для створення проекту у Microsoft Visual Studio створюємо CLR проект (попередньо встановивши бібліотеки C++ / CLI через Microsoft Visual Studio Installer) та робимо посилання на простір імен System.Numerics (див. рис. 1-2). Після цього можемо починати роботу з BigInteger.

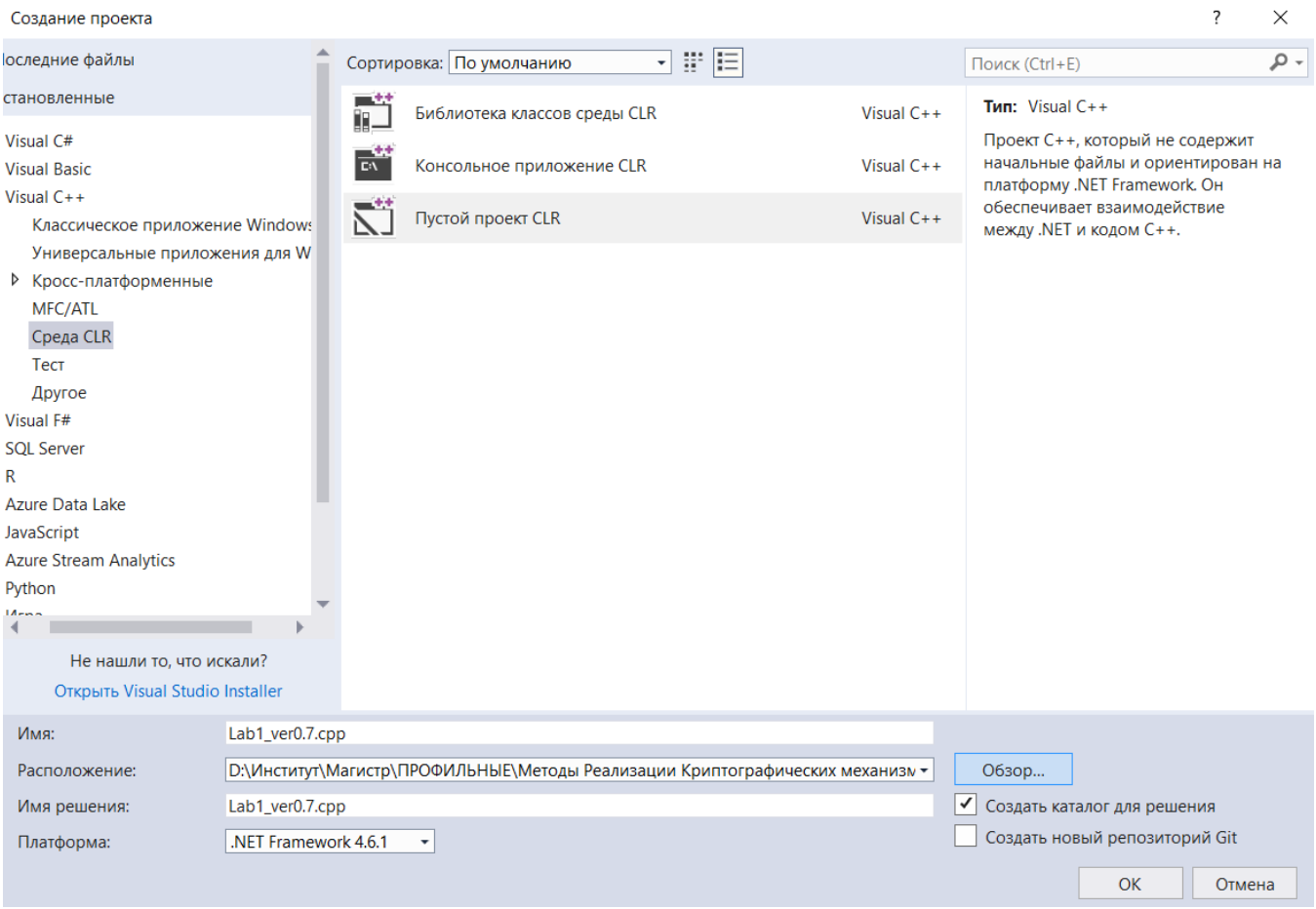

Рис.1. Створення CLR проекту.

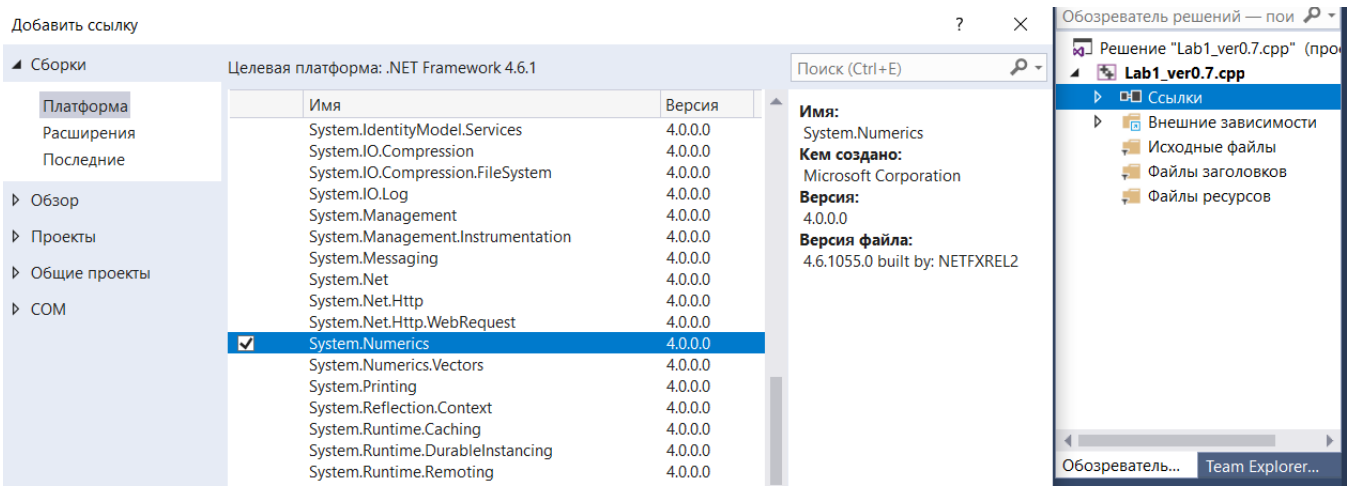

Рис.2. Підключення посилання на System::Numerics.

Бібліотека BigInteger містить такі конструктори:

- BigInteger(Byte[]) – ініціалізація екземпляру BigInteger за допомогою значень у масиві типу Byte;
- BigInteger(Decimal) – ініціалізація екземпляру BigInteger за допомогою значення Decimal (зазвичай використовується у фінансових розрахунках);

- BigInteger(Double) – ініціалізація екземпляру BigInteger за допомогою значення з подвійною точністю;
- BigInteger(Int32) – ініціалізація екземпляру BigInteger за 32-бітовим цілим значенням;
- BigInteger(Int64) – ініціалізація екземпляру BigInteger за 64-бітовим цілим значення;
- BigInteger(Single) – ініціалізація екземпляру BigInteger за допомогою значення з одинарною точністю (на мою думку, краще не використовувати, адже такі перетворення зводять взагалі точність Single нанівець);
- BigInteger(UInt32) – ініціалізація екземпляру BigInteger за 32-бітовим беззнаковим цілим значенням;
- BigInteger(UInt64) – ініціалізація екземпляру BigInteger за 64-бітовим беззнаковим цілим значенням.

Характеристиками BigInteger є:

- IsEven – показує чи є значення поточного BigInteger об’єкту парним (true OR false);
- IsOne – визначає чи є значення поточного BigInteger рівним 1 (true або false);
- IsPowerOfTwo – показує чи є значення поточного BigInteger; степенем двійки (true або false);
- IsZero – визначає чи є значення поточного BigInteger рівним 0 (true або false);
- MinusOne – повертає значення -1 як тип BigInteger;
- One – повертає значення 1 як тип BigInteger;
- Sign – повертає число, що вказує знак (-1, або 0, або 1);
- Zero – повертає значення 0 як тип BigInteger.

Основними методами BigInteger є:

- Abs(BigInteger) – повертає абсолютне значення об’єкту BigInteger;
- Add(BigInteger, BigInteger) – повертає суму двох значень типу BigInteger;

- Compare(BigInteger, BigInteger) – порівняння двох об’єктів типу BigInteger (-1, або 0, або 1);
- Divide(BigInteger, BigInteger) – повертає цілу частину від ділення першого числа на друге;
- Log(BigInteger) – повертає значення натурального логарифма від BigInteger;
- Log(BigInteger, Double) – повертає значення логарифма від BigInteger за основою з подвійною точністю;
- Log10(BigInteger) – повертає значення десяткового логарифма від BigInteger;
- Max(BigInteger, BigInteger) – повертає максимальне з двох значень;
- Min(BigInteger, BigInteger) – повертає мінімальне з двох значень;
- ModPow(BigInteger A, BigInteger B, BigInteger C) – повертає значення ;
- Multiply(BigInteger, BigInteger) – повертає значення добутку двох чисел;
- Parse(String) – конвертує значення String до його BigInteger еквіваленту;
- Parse(String, NumberStyles) – конвертує значення String, яке задано у форматі NumberStyles (для цього потрібно підключити посилання System.Globalization) у BigInteger;
- Pow(BigInteger, Int32) – підносить значення BigInteger до Int32;
- Remainder(BigInteger, BigInteger) – залишок від ділення першого BigInteger числа на друге;
- Subtract(BigInteger, BigInteger) – повертає різницю двох значень.

У програмній реалізації користуємось цими методами та підраховуємо середні часові витрати базових операцій (методів Add, Subtract, Multiply, Divide) та методу піднесення до степеня (Pow) для 256, 512, 1024 та 2048-бітових чисел.

Окрім цього, реалізовуємо функцію mod(BigInteger, BigInteger), яка разом з методом Pow(BigInteger, Int32) обчислює модуль значення, що піднесли до 32-бітового степеня, та порівнюємо її швидкість зі швидкістю при використанні конструктора BigInteger(Int32) та метода ModPow(BigInteger, BigInteger, BigInteger) обраної бібліотеки.

У бібліотеці BigInteger наявні оператори, наприклад такі:

- Addition(BigInteger A, BigInteger B) – те саме, що A+B;
- BitwiseAnd(BigInteger A, BigInteger B) – побітове додавання (A&B);
- BitwiseOr(BigInteger A, BigInteger B) – побітове АБО (A|B);
- Decrement(BigInteger A) – зменшення значення A на 1;
- Division(BigInteger A, BigInteger B) – те саме, що A/B;
- ExclusiveOr(BigInteger A, BigInteger B) – виключне АБО (A^B);
- Equality(BigInteger A, BigInteger B) – повертає true, якщо A=B, інакше – false (A==B);
- GreaterThan(BigInteger A, BigInteger B) – перевіряє чи A>B;
- LessThan(BigInteger A, BigInteger B) – перевіряє чи A<B.

Детальніше з бібліотекою BigInteger можна ознайомитись за посиланням: https://docs.microsoft.com/en-us/dotnet/api/system.numerics.biginteger?view=net-5.0.

Приклади використання даної бібліотеки наведені та закоментовані у коді програми (див. додаток 1, додаток 2, Lab1_Zvychainaia.cpp). Наприклад, для 519-бітових чисел:

A=17BC9E6E5CA3078D067DDB15CB2DF7B7D5E6437F99FE5FBB91750C3D2E4AFC7FCABC5F8795D0570EBF7A90AA0603D88FB10169B55D592959BBF88E2F141B16C3,

B=1F2A11C94AB245C494806E38BDADD88C3EBFD5E688A51FDDFDFFE23069EFF0BAE0BC82B1B2D07E6B56764F01B331B47D7DDE8641778784A6AEF78BC72F337B3E,

N=10C6E1E34644B70666262C3DC6BD321A8C8A65753C6D90F4628EB5866FA0B4D494CC9294BF4A7F252B74521DFBC9D290AFB73B7844D896C658F5A76423D074D8,

маємо такі результати у десятковому представлені:

A:
1243196696355907740147500256204789478087001658190114354789963977086471739494986347545830785751832437078703038218927262870059960197933867890795180180838083

B:
1632208606094235621004024403396637463227651743043234749147816563108321215604802132706244391170976103735499936182856735202607599153515721005320485523913534

N:
878676726604422491045404036048823798868212919670382876756929076385186148871029440376747104253270514609775175198537825961735875892555537715429395592672472

A+B:
287540530245014336115152465960142694131465340123334910393778054019479295509978848025207517692280854081420297440178399807266755935144958889611566570475161 7

(A+B)modN:
239375122636875888015312551454955544710014642222200473666993311039234508486700159121833864162996996984877448806170520187459931673782975749827478926734201

A-B: -
389011909738327880856524147191847985140650084853120394357852586021849476109815785160413605419143666656796897963929472332547638955581853114525305343075451

(A-B)modN:
489664816866094610188879888856975813727562834817262482399076490363336672761213655216333498834126847952978277234608353629188236936973684600904090249597021

A*B:
202915634686003486512201215548367179090574726933472204080290261804027103155682278204782970343833802785435601421040162145183301026707160337752246741571629768620891203751066696306314966528846089507001168314479622213730942098206498356359989442536909370447461962823446435896686748281531518291822673989414631532 2

(A*B)modN:
548325362011551653253266777893364811017857538037360381387307581549488759873650187359287549718430089665773675396802847698965879849515629743676207790858498

B/A: 1

(B/A)modN: 1

Rem_(B/A)modN:
389011909738327880856524147191847985140650084853120394357852586021849476109815785160413605419143666656796897963929472332547638955581853114525305343075451

Для підрахунку середніх часових витрат використовувались 1 тис., 10 тис., 100 тис., 1 млн. та 10 млн. запусків кожної базової операції (методів Add, Subtract, Multiply, Divide) та операції піднесення до степеня Pow з використанням вбудованих методів Remainder та ModPow, а також з використанням власної реалізації функції знаходження залишку (функції mod) для 256, 512, 1024 та 2048-бітових чисел. Порівняльна таблиця результатів наведена нижче.

	256-бітові числа	**512-бітові числа**	**1024-бітові числа**	**2048-бітові числа**
1 000 ітерацій		***ДОДАВАННЯ***		

Remainder(Add(A,B),N)	90 мкс.	361 мкс.	260 мкс.	136 мкс.
mod(Add(A,B),N)	102 мкс.	289 мкс.	421 мкс.	199 мкс.
10 000 ітерацій				
Remainder(Add(A,B),N)	55 мкс.	231 мкс.	653 мкс.	1 мс.
mod(Add(A,B),N)	133 мкс.	203 мкс.	900 мкс.	1 мс.
100 000 ітерацій				
Remainder(Add(A,B),N)	1 мс.	1 мс.	7 мс.	18 мс.
mod(Add(A,B),N)	1 мс.	2 мс.	9 мс.	26 мс.
1 000 000 ітерацій				
Remainder(Add(A,B),N)	15 мс.	29 мс.	83 мс.	222 мс.
mod(Add(A,B),N)	24 мс.	58 мс.	125 мс.	302 мс.
10 000 000 ітерацій				
Remainder(Add(A,B),N)	187 мс.	274 мс.	930 мс.	2936 мс.
mod(Add(A,B),N)	297 мс.	516 мс.	1376 мс.	3756 мс.
1 000 ітерацій		***ВІДНІМАННЯ***		
Remainder(Sub(A,B),N)	32 мкс.	37 мкс.	41 мкс.	7 мкс.
mod(Sub(A,B),N)	81 мкс.	257 мкс.	345 мкс.	99 мкс.
10 000 ітерацій				
Remainder(Sub(A,B),N)	16 мкс.	35 мкс.	65 мкс.	37 мкс.
mod(Sub(A,B),N)	178 мкс.	236 мкс.	517 мкс.	872 мкс.
100 000 ітерацій				
Remainder(Sub(A,B),N)	306 мкс.	279 мкс.	905 мкс.	1 мс.
mod(Sub(A,B),N)	1 мс.	3 мс.	5 мс.	17 мс.
1 000 000 ітерацій				
Remainder(Sub(A,B),N)	4 мс.	7 мс.	9 мс.	9 мс.
mod(Sub(A,B),N)	29 мс.	52 мс.	86 мс.	172 мс.
10 000 000 ітерацій				
Remainder(Sub(A,B),N)	64 мс.	67 мс.	81 мс.	115 мс.
mod(Sub(A,B),N)	350 мс.	520 мс.	988 мс.	2034 мс.
1 000 ітерацій		***МНОЖЕННЯ***		

Remainder(Mul(A,B),N)	1 мс.	3 мс.	10 мс.	30 мс.
mod(Mul(A,B),N)	1 мс.	3 мс.	10 мс.	30 мс.
10 000 ітерацій				
Remainder(Mul(A,B),N)	10 мс.	20 мс.	83 мс.	305 мс.
mod(Mul(A,B),N)	10 мс.	21 мс.	83 мс.	306 мс.
100 000 ітерацій				
Remainder(Mul(A,B),N)	102 мс.	212 мс.	836 мс.	3 с.
mod(Mul(A,B),N)	102 мс.	212 мс.	838 мс.	3 с.
1 000 000 ітерацій				
Remainder(Mul(A,B),N)	1 с.	2 с.	9 с.	34 с.
mod(Mul(A,B),N)	1 с.	2 с.	9 с.	34 с.
10 000 000 ітерацій				
Remainder(Mul(A,B),N)	10 с.	24 с.	92 с.	351 с.
mod(Mul(A,B),N)	10 с.	24 с.	92 с.	351 с.
1 000 ітерацій		***ДІЛЕННЯ***		
Remainder(Div(A,B),N)	4 мкс.	12 мкс.	14 мкс.	1 мкс.
Mod(Div(A,B),N)	2 мкс.	25 мкс.	39 мкс.	13 мкс.
10 000 ітерацій				
Remainder(Div(A,B),N)	8 мкс.	10 мкс.	22 мкс.	44 мкс.
mod(Div(A,B),N)	57 мкс.	26 мкс.	52 мкс.	62 мкс.
100 000 ітерацій				
Remainder(Div(A,B),N)	131 мкс.	130 мкс.	340 мкс.	966 мкс.
mod(Div(A,B),N)	180 мкс.	205 мкс.	603 мкс.	1 мс.
1 000 000 ітерацій				
Remainder(Div(A,B),N)	2 мс.	4 мс.	7 мс.	9 мс.
mod(Div(A,B),N)	3 мс.	5 мс.	8 мс.	12 мс.
10 000 000 ітерацій				
Remainder(Div(A,B),N)	27 мс.	32 мс.	53 мс.	99 мс.
mod(Div(A,B),N)	45 мс.	57 мс.	81 мс.	178 мс.
1 000 ітерацій		***ПІДНЕСЕННЯ ДО КВАДРАТУ***		

Remainder(Pow(A,2),N)	1 мс.	3 мс.	11 мс.	31 мс.
Mod(Pow(A,2),N)	1 мс.	3 мс.	11 мс.	31 мс.
ModPow(A,2,N)	1 мс.	3 мс.	10 мс.	31 мс.
10 000 ітерацій				
Remainder(Pow(A,2),N)	10 мс.	23 мс.	91 мс.	317 мс.
mod(Pow(A,2),N)	10 мс.	26 мс.	92 мс.	318 мс.
ModPow(A,2,N)	10 мс.	20 мс.	83 мс.	315 мс.
100 000 ітерацій				
Remainder(Pow(A,2),N)	103 мс.	236 мс.	917 мс.	3 с.
mod(Pow(A,2),N)	103 мс.	261 мс.	926 мс.	3 с.
ModPow(A,2,N)	102 мс.	210 мс.	841 мс.	3 с.
1 000 000 ітерацій				
Remainder(Pow(A,2),N)	1 с.	3 с.	9 с.	34 с.
mod(Pow(A,2),N)	1 с.	3 с.	9 с.	34 с.
ModPow(A,2,N)	1 с.	2 с.	9 с.	34 с.
10 000 000 ітерацій				
Remainder(Pow(A,2),N)	11 с.	30 с.	97 с.	355 с.
mod(Pow(A,2),N)	11 с.	30 с.	97 с.	355 с.
ModPow(A,2,N)	10 с.	23 с.	93 с.	349 с.
1 000 ітерацій	***ПІДНЕСЕННЯ ДО ВЕЛИКОГО СТЕПЕНЯ***			
ModPow(A,B,N)	13 мс.	33 мс.	77 мс.	170 мс.
10 000 ітерацій				
ModPow(A,B,N)	123 мс.	273 мс.	658 мс.	1 с.
100 000 ітерацій				
ModPow(A,B,N)	1 с.	2 с.	6 с.	17 с.
1 000 000 ітерацій				
ModPow(A,B,N)	13 с.	29 с.	70 с.	184 с.
10 000 000 ітерацій				
ModPow(A,B,N)	135 с.	298 с.	714 с.	1883 с.

Висновки:

Сьогодні для забезпечення захисту інформації використовують криптографічні алгоритми, де параметрами є 2048-бітові числа. Очевидно, що використання стандартних 32-бітових (як unsigned int) чи 64-бітових (як unsigned long long) типів даних для реалізації таких алгоритмів є певним ускладненням, що призводить до необхідності реалізації й самої бібліотеки для роботи з великими числами, тобто числами, розрядність яких більша за довжину машинного слова комп'ютера. Нині існують готові бібліотеки багаторозрядної арифметики, реалізація яких безпосередньо спирається на роботу з числами менших порядків. Бібліотеки представлені у вигляді структур та вбудовані у програмні платформи (зазвичай на основі .NET Framework): BigInteger, BigDecimal – для C#/C++, BigInt – для Java, Decimal – для Python.

У даному комп'ютерному практикумі свою увагу я зосередила на досліджені можливостей використання бібліотеки BigInteger у контексті розробки C++ додатків для захисту інформації. Для цього я познайомилась з середовищем CLR (Common Language Runtime), яке є розробкою компанії Microsoft з метою прискорення розробки проектів; розглянула конструктори, характеристики, методи, операції бібліотеки BigInteger; практично визначила час роботи основних операцій для 256, 512, 1024 та 2048-бітових чисел (час росте зі збільшенням довжини числа та всі операції у середньому виконуються менше ніж за с). Цікавим виявилось те, що при 10 000 000 ітерацій операція ділення виконується значно швидше за множення та обмежується мілісекундами, коли множення – секундами (351 с. проти 99 мс.). У подальшому пропонується дослідити час роботи операцій при більшій кількості ітерацій.

ДОДАТОК 1. Код програми

```
#include <iostream>
#include <string>
#include <chrono>
#include <windows.h>
#include <map>
using namespace std;
using namespace std::chrono;
using namespace System::Globalization; // Для NumberStyles::AllowHexSpecifier
using namespace System::Numerics;
using namespace System;


void input(string message, string el)  // Ввод элементов
{
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);  // Получаем дескриптор консоли
SetConsoleTextAttribute(hStdOut, 7);  // Меняем цвет текста на серый
cout << message;  // Выводим сообщение
SetConsoleTextAttribute(hStdOut, 15);  // Меняем цвет текста на белый
cin >> el; // Вводим элемент
}
void print(string message, BigInteger el)  // Вывод элементов
{
```

```
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);  // Получаем дескриптор консоли
SetConsoleTextAttribute(hStdOut, 7);  // Меняем цвет текста на серый
cout << message;  // Выводим сообщение
SetConsoleTextAttribute(hStdOut, 15);  // Меняем цвет текста на белый
Console::WriteLine(el);  // Выводим элемент
}
void print(string message, string el)  // Вывод элементов
{
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);  // Получаем дескриптор консоли
SetConsoleTextAttribute(hStdOut, 7);  // Меняем цвет текста на серый
cout << message;  // Выводим сообщение
SetConsoleTextAttribute(hStdOut, 15);  // Меняем цвет текста на белый
cout << el << endl;  // Выводим элемент
}
string elapsedTime(time_point<system_clock> start, time_point<system_clock> end) // Конвертирование время в строку
{
int count = duration_cast<microseconds>(end - start).count();  // Считаем прошедшее время
return count > 1000 ? to_string(count / 1000) + " мс." : to_string(count) + " мкс.";  // Возвращаем строку
}
long elapsedTime2(time_point<system_clock> start, time_point<system_clock> end) // Конвертирование время в строку
{
return duration_cast<microseconds>(end - start).count();  // Возвращаем прошедшее время в мкс
}
static BigInteger mod(BigInteger a, BigInteger b)
```

```
{
if (a > BigInteger(0))  // Если делимое больше 0
return a % b;  // Возвращаем модуль a на b
if ((a % b) != BigInteger(0))  // Если модуль не равен 0
// Возвращаем остаток от деления
return BigInteger::Subtract(BigInteger::Multiply(BigInteger::Add(BigInteger::Divide(BigInteger::Abs(a),
b), BigInteger(1)), b), BigInteger::Abs(a));
return 0;  // Возвращаем 0
}
void print(BigInteger a, BigInteger b, BigInteger c, BigInteger mod_amount) {
BigInteger result;
time_point<system_clock> start, end;  // Переменные для измерения времени

print("Первый элемент (a): ", a);
print("Второй элемент (b): ", b);
print("Третий элемент (c): ", c);
print("Модуль (mod): ", mod_amount);
cout << endl;

start = system_clock::now();
result = mod(BigInteger(0), BigInteger(1));
end = system_clock::now();

start = system_clock::now();
c = mod(c, mod_amount);
end = system_clock::now();
```

```
print("Третий элемент по модулю:", c);
print("Время работы операции взятия по модулю: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = mod(BigInteger::Add(c, a), mod_amount);
end = system_clock::now();

print("Сумма первого и третьего по модулю: ", result);
print("Время работы операции сложения: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = mod(BigInteger::Subtract(a, b), mod_amount);
end = system_clock::now();

print("Разница первого элемента и второго по модулю: ", result);
print("Время работы операции разница: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = mod(BigInteger::Multiply(a, b), mod_amount);
end = system_clock::now();

print("Умножение первого элемента на второй по модулю: ", result);
```

```
print("Время работы операции умножения: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = mod(BigInteger::Pow(a, 2), mod_amount);
end = system_clock::now();

print("Возведение первого элемента в квадрат по модулю (моя функция): ", result);
print("Время работы операции возведения в квадрат первого элемента: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = BigInteger::ModPow(a, BigInteger(2), mod_amount);
end = system_clock::now();

print("Возведение первого элемента в квадрат по модулю (встроенная функция): ", result);
print("Время работы операции возведения в квадрат первого элемента: ", elapsedTime(start, end));
cout << endl;

start = system_clock::now();
result = BigInteger::Divide(a, b);
end = system_clock::now();

print("Целая часть от деления a на b: ", result);
print("Время работы операции деление: ", elapsedTime(start, end));
cout << endl;
```

```
print("Тест на корректность (проверка дистрибутивности): ", "");

print("Наш первый элемент (a): ", a);

print("Наш второй элемент (b): ", b);

print("Наш третий элемент (c): ", BigInteger::Subtract(c, BigInteger(1)));

print("Результат (a+b)*c : ", mod(BigInteger::Multiply(BigInteger::Add(a, b), BigInteger::Subtract(c, BigInteger(1))), mod_amount));

print("Результат b*c+c*a : ", mod(BigInteger::Add(BigInteger::Multiply(b, BigInteger::Subtract(c, BigInteger(1))), BigInteger::Multiply(BigInteger::Subtract(c, BigInteger(1)), a)), mod_amount));

cout << endl;

}


void average_time_count(BigInteger a, BigInteger b, BigInteger mod_amount, BigInteger iter_amount) {

BigInteger num_iter = BigInteger(0);

BigInteger result;

BigInteger average_time_count_add = BigInteger(0);

BigInteger average_time_count_sub = BigInteger(0);

BigInteger average_time_count_mul = BigInteger(0);

BigInteger average_time_count_div = BigInteger(0);

BigInteger average_time_count_powb = BigInteger(0);

BigInteger average_time_count_pow2 = BigInteger(0);

BigInteger average_time_count_pow2_rem = BigInteger(0);

//--------------------------------------------------

BigInteger average_time_count_add_my = BigInteger(0);

BigInteger average_time_count_sub_my = BigInteger(0);

BigInteger average_time_count_mul_my = BigInteger(0);
```

```
BigInteger average_time_count_div_my = BigInteger(0);

BigInteger average_time_count_pow2_my = BigInteger(0);

time_point<system_clock> start, end;  // Переменные для измерения времени

start = system_clock::now();

a = mod(a, mod_amount);

end = system_clock::now();


while (BigInteger::Compare(num_iter, iter_amount) != 0) { // Пока не пройдет iter_amount итераций, делаем


start = system_clock::now();

result = BigInteger::Remainder(BigInteger::Add(a, b), mod_amount);

end = system_clock::now();

average_time_count_add = BigInteger::Add(average_time_count_add, BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = BigInteger::Remainder(BigInteger::Subtract(a, b), mod_amount);

end = system_clock::now();

average_time_count_sub = BigInteger::Add(average_time_count_sub, BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = BigInteger::Remainder(BigInteger::Multiply(a, b), mod_amount);

end = system_clock::now();

average_time_count_mul = BigInteger::Add(average_time_count_mul, BigInteger(elapsedTime2(start, end)));
```

```
start = system_clock::now();

result = BigInteger::Remainder(BigInteger::Divide(a, b), mod_amount);

end = system_clock::now();

average_time_count_div = BigInteger::Add(average_time_count_div, BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = BigInteger::ModPow(a, 2, mod_amount);

end = system_clock::now();

average_time_count_pow2 = BigInteger::Add(average_time_count_pow2, BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = BigInteger::Remainder(BigInteger::Pow(a, 2), mod_amount);

end = system_clock::now();

average_time_count_pow2_rem = BigInteger::Add(average_time_count_pow2_rem, BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = BigInteger::ModPow(a, b, mod_amount);

end = system_clock::now();

average_time_count_powb = BigInteger::Add(average_time_count_powb, BigInteger(elapsedTime2(start, end)));


// ----------With my func "mod":

start = system_clock::now();
```

```
result = mod(BigInteger::Add(a, b), mod_amount);

end = system_clock::now();

average_time_count_add_my = BigInteger::Add(average_time_count_add_my,
BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = mod(BigInteger::Subtract(a, b), mod_amount);

end = system_clock::now();

average_time_count_sub_my = BigInteger::Add(average_time_count_sub_my,
BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = mod(BigInteger::Multiply(a, b), mod_amount);

end = system_clock::now();

average_time_count_mul_my = BigInteger::Add(average_time_count_mul_my,
BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = mod(BigInteger::Divide(a, b), mod_amount);

end = system_clock::now();

average_time_count_div_my = BigInteger::Add(average_time_count_div_my,
BigInteger(elapsedTime2(start, end)));


start = system_clock::now();

result = mod(BigInteger::Pow(a, 2), mod_amount);

end = system_clock::now();
```

```
average_time_count_pow2_my = BigInteger::Add(average_time_count_pow2_my, BigInteger(elapsedTime2(start, end)));


//a = BigInteger::Add(a, BigInteger(1));

//b = BigInteger::Add(b, BigInteger(1));

num_iter++;

}

//average_time_count_add = BigInteger::Divide(average_time_count_add, BigInteger(iter_amount));

//average_time_count_sub = BigInteger::Divide(average_time_count_sub, BigInteger(iter_amount));

//average_time_count_mul = BigInteger::Divide(average_time_count_mul, BigInteger(iter_amount));

//average_time_count_div = BigInteger::Divide(average_time_count_div, BigInteger(iter_amount));

if (average_time_count_add > BigInteger(1000))

print("Время работы операции Add (в мс): ", BigInteger::Divide(average_time_count_add, BigInteger(1000)));

else print("Время работы операции Add (в мкс): ", average_time_count_add);


if (average_time_count_add_my > BigInteger(1000))

print("Время работы операции Add (my mod_func, в мс): ", BigInteger::Divide(average_time_count_add_my, BigInteger(1000)));

else print("Время работы операции Add (my mod_func, в мкс): ", average_time_count_add_my);


if (average_time_count_sub > BigInteger(1000))

print("Время работы операции Subtract (в мс): ", BigInteger::Divide(average_time_count_sub, BigInteger(1000)));

else print("Время работы операции Subtract (в мкс): ", average_time_count_sub);


if (average_time_count_sub_my > BigInteger(1000))
```

```
print("Время работы операции Subtract (my mod_func, в мс): ", BigInteger::Divide(average_time_count_sub_my, BigInteger(1000)));

else print("Время работы операции Subtract (my mod_func, в мкс): ", average_time_count_sub_my);

//--------------------Multiply-------------------------------------

if (average_time_count_mul > BigInteger(1000000))

print("Время работы операции Multiply (в с): ", BigInteger::Divide(average_time_count_mul, BigInteger(1000000)));

else

if (average_time_count_mul > BigInteger(1000))

print("Время работы операции Multiply (в мс): ", BigInteger::Divide(average_time_count_mul, BigInteger(1000)));

else print("Время работы операции Multiply (в мкс): ", average_time_count_mul);

//---------------------Multiply(My mod_func)------------------------

if (average_time_count_mul_my > BigInteger(1000000))

print("Время работы операции Multiply (my mod_func, в с): ", BigInteger::Divide(average_time_count_mul_my, BigInteger(1000000)));

else

if (average_time_count_mul_my > BigInteger(1000))

print("Время работы операции Multiply (my mod_func, в мс): ", BigInteger::Divide(average_time_count_mul_my, BigInteger(1000)));

else print("Время работы операции Multiply (my mod_func, в мкс): ", average_time_count_mul_my);

//----------------------------------------------------------------

if (average_time_count_div > BigInteger(1000))

print("Время работы операции Divide (в мс): ", BigInteger::Divide(average_time_count_div, BigInteger(1000)));

else print("Время работы операции Divide (в мкс): ", average_time_count_div);
```

```
if (average_time_count_div_my > BigInteger(1000))

print("Время работы операции Divide (my mod_func, в мс): ", BigInteger::Divide(average_time_count_div_my, BigInteger(1000)));

else print("Время работы операции Divide (my mod_func, в мкс): ", average_time_count_div_my);

//---------------------------ModPow---------------------------------------

if (average_time_count_pow2 > BigInteger(1000000))

print("Время работы операции ModPow (a^2, в с): ", BigInteger::Divide(average_time_count_pow2, BigInteger(1000000)));

else

if (average_time_count_pow2 > BigInteger(1000))

print("Время работы операции ModPow (a^2, в мс): ", BigInteger::Divide(average_time_count_pow2, BigInteger(1000)));

else print("Время работы операции ModPow (a^2, в мкс): ", average_time_count_pow2);

//----------------------------Pow(Rem_func)-----------------------------------

if (average_time_count_pow2_rem > BigInteger(1000000))

print("Время работы операции Pow (a^2, Rem_func, в с): ", BigInteger::Divide(average_time_count_pow2_rem, BigInteger(1000000)));

else

if (average_time_count_pow2_rem > BigInteger(1000))

print("Время работы операции Pow (a^2, Rem_func, в мс): ", BigInteger::Divide(average_time_count_pow2_rem, BigInteger(1000)));

else print("Время работы операции Pow (a^2, Rem_func, в мкс): ", average_time_count_pow2_rem);

//----------------------------Pow(my mod_func)-----------------------------------

if (average_time_count_pow2_my > BigInteger(1000000))

print("Время работы операции Pow (a^2, my mod_func, в с): ", BigInteger::Divide(average_time_count_pow2_my, BigInteger(1000000)));

else
```

```
if (average_time_count_pow2_my > BigInteger(1000))
print("Время работы операции Pow (a^2, my mod_func, в мс): ", BigInteger::Divide(average_time_count_pow2_my, BigInteger(1000)));
else print("Время работы операции Pow (a^2, my mod_func, в мкс): ", average_time_count_pow2_my);
//----------------------------ModPow (a^b)----------------------------------------
if (average_time_count_powb > BigInteger(1000000))
print("Время работы операции ModPow (a^b, в с): ", BigInteger::Divide(average_time_count_powb, BigInteger(1000000)));
else
if (average_time_count_powb > BigInteger(1000))
print("Время работы операции ModPow (a^b, в мс): ", BigInteger::Divide(average_time_count_powb, BigInteger(1000)));
else print("Время работы операции ModPow (a^b, в мкс): ", average_time_count_powb);
}

void main()
{
setlocale(LC_ALL, "rus");  // Поддержка кириллицы
// Переменные (seed)
BigInteger a_256 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 256), BigInteger(50000000000));
BigInteger b_256 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 256), BigInteger(10000000000));
BigInteger c_256 = BigInteger::Add(BigInteger::Pow(BigInteger(2), 256), BigInteger(500));

BigInteger a_512 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 512), BigInteger(50000000000));
```

```
BigInteger b_512 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 512), BigInteger(10000000000));

BigInteger c_512 = BigInteger::Add(BigInteger::Pow(BigInteger(2), 512), BigInteger(500));


BigInteger a_1024 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 1024), BigInteger(50000000000));

BigInteger b_1024 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 1024), BigInteger(10000000000));

BigInteger c_1024 = BigInteger::Add(BigInteger::Pow(BigInteger(2), 1024), BigInteger(500));


BigInteger a_2048 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 2048), BigInteger(50000000000));

BigInteger b_2048 = BigInteger::Subtract(BigInteger::Pow(BigInteger(2), 2048), BigInteger(10000000000));

BigInteger c_2048 = BigInteger::Add(BigInteger::Pow(BigInteger(2), 2048), BigInteger(500));


// Модули

BigInteger mod_256 = BigInteger::Pow(BigInteger(2), 256);

BigInteger mod_512 = BigInteger::Pow(BigInteger(2), 512);

BigInteger mod_1024 = BigInteger::Pow(BigInteger(2), 1024);

BigInteger mod_2048 = BigInteger::Pow(BigInteger(2), 2048);


/*print(a_256, b_256, c_256, mod_256);

cout << endl;

print(a_512, b_512, c_512, mod_512);

cout << endl;

print(a_1024, b_1024, c_1024, mod_1024);
```

```
cout << endl;

print(a_2048, b_2048, c_2048, mod_2048);*/

int sizeIter_amount = 5; // 1 000,   10 000,   100 000,   1 000 000,   10 000 000

int *Iter_amount = new int[sizeIter_amount];

for (int i = 0, j = 1000; i < sizeIter_amount; i++) {

Iter_amount[i] = j;

j *= 10;

}

cout << "-----------------------------------------------------" << endl;

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

cout << "Замеры среднего времени базовых операций по модулю 2^256 для " << Iter_amount[i] << " итераций: " << endl;

average_time_count(a_256, b_256, mod_256, BigInteger(Iter_amount[i]));

cout << endl << "Замеры среднего времени базовых операций по модулю 2^512 для " << Iter_amount[i] << " итераций: " << endl;

average_time_count(a_512, b_512, mod_512, BigInteger(Iter_amount[i]));

cout << endl << "Замеры среднего времени базовых операций по модулю 2^1024 для " << Iter_amount[i] << " итераций: " << endl;

average_time_count(a_1024, b_1024, mod_1024, BigInteger(Iter_amount[i]));

cout << endl << "Замеры среднего времени базовых операций по модулю 2^2048 для " << Iter_amount[i] << " итераций: " << endl;

average_time_count(a_2048, b_2048, mod_2048, BigInteger(Iter_amount[i]));

cout << "-----------------------------------------------------" << endl;

}

delete[] Iter_amount;

/*BigInteger a_bigInt = BigInteger::Parse("17BC9E6E5CA3078D067DDB15CB2DF7B7D5E6437F99FE5FBB91750C3D2E4AF
```

```
C7FCABC5F8795D0570EBF7A90AA0603D88FB10169B55D592959BBF88E2F141B16C3",
NumberStyles::AllowHexSpecifier);

BigInteger b_bigInt = BigInteger::Parse("1F2A11C94AB245C494806E38BDADD88C3EBFD5E688A51FDDFDFFE23069EF
F0BAE0BC82B1B2D07E6B56764F01B331B47D7DDE8641778784A6AEF78BC72F337B3E",
NumberStyles::AllowHexSpecifier);

BigInteger mod_bigInt = BigInteger::Parse("10C6E1E34644B70666262C3DC6BD321A8C8A65753C6D90F4628EB5866FA0B4
D494CC9294BF4A7F252B74521DFBC9D290AFB73B7844D896C658F5A76423D074D8",
NumberStyles::AllowHexSpecifier);

print("A: ", a_bigInt);

print("B: ", b_bigInt);

print("N: ", mod_bigInt);

print("A+B: ", BigInteger::Add(a_bigInt, b_bigInt));

print("(A+B)modN: ", mod(BigInteger::Add(a_bigInt, b_bigInt), mod_bigInt));

print("A-B: ", BigInteger::Subtract(a_bigInt, b_bigInt));

print("(A-B)modN: ", mod(BigInteger::Subtract(a_bigInt, b_bigInt), mod_bigInt));

print("A*B: ", BigInteger::Multiply(a_bigInt, b_bigInt));

print("(A*B)modN: ", mod(BigInteger::Multiply(a_bigInt, b_bigInt), mod_bigInt));

if (BigInteger::Compare(a_bigInt, b_bigInt) == 1) {

print("A/B: ", BigInteger::Divide(a_bigInt, b_bigInt));

print("(A/B)modN: ", mod(BigInteger::Divide(a_bigInt, b_bigInt), mod_bigInt));

print("Rem_(A/B)modN: ", mod(BigInteger::Remainder(a_bigInt, b_bigInt), mod_bigInt));

}

else {

print("B/A: ", BigInteger::Divide(b_bigInt, a_bigInt));

print("(B/A)modN: ", mod(BigInteger::Divide(b_bigInt, a_bigInt), mod_bigInt));

print("Rem_(B/A)modN: ", mod(BigInteger::Remainder(b_bigInt, a_bigInt), mod_bigInt));
```

```
}

cout << "----------------------------------------------------" << endl;*/

cin.get();

cin.get();

}
```

ДОДАТОК 2. Виконання програми та приклади реалізації

Первый элемент (a):
1157920892373161954235709850086879078532699846656405640394575840078631296399 36

Второй элемент (b):
1157920892373161954235709850086879078532699846656405640394575840079031296399 36

Третий элемент (c):
1157920892373161954235709850086879078532699846656405640394575840079131296404 36

Модуль (mod):
1157920892373161954235709850086879078532699846656405640394575840079131296399 36

Третий элемент по модулю:500

Время работы операции взятия по модулю: 3 мкс.

Сумма первого и третьего по модулю:
1157920892373161954235709850086879078532699846656405640394575840078631296404 36

Время работы операции сложения: 3 мкс.

Разница первого элемента и второго по модулю:
1157920892373161954235709850086879078532699846656405640394575840078731296399 36

Время работы операции разница: 14 мкс.

Умножение первого элемента на второй по модулю: 500000000000000000000

Время работы операции умножения: 15 мкс.

Возведение первого элемента в квадрат по модулю (моя функция): 250000000000000000000

Время работы операции возведения в квадрат первого элемента: 6 мкс.

Возведение первого элемента в квадрат по модулю (встроенная функция):
250000000000000000000

Время работы операции возведения в квадрат первого элемента: 7 мкс.

Целая часть от деления a на b: 0

Время работы операции деление: 1 мкс.

Тест на корректность (проверка дистрибутивности):

Наш первый элемент (a):
1157920892373161954235709850086879078532699846656405640394575840078631296399 36

Наш второй элемент (b):
1157920892373161954235709850086879078532699846656405640394575840079031296399 36

Наш третий элемент (c): 499

Результат (a+b)*c :
1157920892373161954235709850086879078532699846656405640394575839779731296399 36

Результат b*c+c*a :
1157920892373161954235709850086879078532699846656405640394575839779731296399 36

Первый элемент (a):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433599006084096

Второй элемент (b):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433639006084096

Третий элемент (c):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433649006084596

Модуль (mod):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433649006084096

Третий элемент по модулю:500

Время работы операции взятия по модулю: 2 мкс.

Сумма первого и третьего по модулю:
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433599006084596

Время работы операции сложения: 2 мкс.

Разница первого элемента и второго по модулю:
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433609006084096

Время работы операции разница: 7 мкс.

Умножение первого элемента на второй по модулю: 500000000000000000000

Время работы операции умножения: 9 мкс.

Возведение первого элемента в квадрат по модулю (моя функция): 2500000000000000000000

Время работы операции возведения в квадрат первого элемента: 13 мкс.

Возведение первого элемента в квадрат по модулю (встроенная функция):
2500000000000000000000

Время работы операции возведения в квадрат первого элемента: 12 мкс.

Целая часть от деления a на b: 0

Время работы операции деление: 2 мкс.

Тест на корректность (проверка дистрибутивности):

Наш первый элемент (a):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433599006084096

Наш второй элемент (b):
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946433639006084096

Наш третий элемент (c): 499

Результат (a+b)*c :
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946403709006084096

Результат b*c+c*a :
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680
1874298166903427690031858186486050853753882811946569946403709006084096

Первый элемент (a):
1797693134862315907729305190789024733617976978942306572734300811577326758055009631327
0847732240753602112011387987139335765878976881441662249284743063947412437776789342486
5485276302219601246094119453082952085005768838150682342462881473913110540827237163350
5106845862982399472459384797163048353563295742241372l6

Второй элемент (b):
1797693134862315907729305190789024733617976978942306572734300811577326758055009631327
0847732240753602112011387987139335765878976881441662249284743063947412437776789342486
5485276302219601246094119453082952085005768838150682342462881473913110540827237163350
5106845862982399472459384797163048353563296142241372l6

Третий элемент (c):
1797693134862315907729305190789024733617976978942306572734300811577326758055009631327
0847732240753602112011387987139335765878976881441662249284743063947412437776789342486
5485276302219601246094119453082952085005768838150682342462881473913110540827237163350
5106845862982399472459384797163048353563296242241372l6

Модуль (mod):
1797693134862315907729305190789024733617976978942306572734300811577326758055009631327
0847732240753602112011387987139335765878976881441662249284743063947412437776789342486
5485276302219601246094119453082952085005768838150682342462881473913110540827237163350
5106845862982399472459384797163048353563296242241372l6

Третий элемент по модулю:500

Время работы операции взятия по модулю: 4 мкс.

Сумма первого и третьего по модулю:
1797693134862315907729305190789024733617976978942306572734300811577326758055009631327
0847732240753602112011387987139335765878976881441662249284743063947412437776789342486

548527630221960124609411945308295208500576883815068234246288147391311054082723716335051068458629823994724593847971630483535632957422413771 6

Время работы операции сложения: 5 мкс.

Разница первого элемента и второго по модулю:
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377677678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353563295842241377 16

Время работы операции разница: 11 мкс.

Умножение первого элемента на второй по модулю: 500000000000000000000

Время работы операции умножения: 26 мкс.

Возведение первого элемента в квадрат по модулю (моя функция): 2500000000000000000000

Время работы операции возведения в квадрат первого элемента: 15 мкс.

Возведение первого элемента в квадрат по модулю (встроенная функция):
250000000000000000000

Время работы операции возведения в квадрат первого элемента: 17 мкс.

Целая часть от деления a на b: 0

Время работы операции деление: 6 мкс.

Тест на корректность (проверка дистрибутивности):

Наш первый элемент (a):
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377677678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353563295842241377 16

Наш второй элемент (b):
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377677678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353563296142241377 16

Наш третий элемент (c): 499

Результат (a+b)*c :
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377677678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353562996842241377 16

Результат b*c+c*a :
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377677678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353562996842241377 16

Первый элемент (a):
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558536111009596230656

Второй элемент (b):
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558536111049596230656

Третий элемент (c):
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558536111059596231156

Модуль (mod):
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558536111059596230656

Третий элемент по модулю:500

Время работы операции взятия по модулю: 4 мкс.

Сумма первого и третьего по модулю:
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425

80656697935045997268352998638215525166389437335543602135433229604645318478604952148
1935558536110095962311 56

Время работы операции сложения: 5 мкс.

Разница первого элемента и второго по модулю:
32317006071311007300714876688669951960444102669715484032130345427524655138867890893197
20141152291346368871796092189801949411955915049092109508815238644828312063087736730 0
99609175019775038965210679605763838406756827679221864261975616183809433847617047058 16
45852036305042887575891541065808607552399123930385521914333389668342420684974786564 56
94948561760353263220580778056593310261927084603141502585928641771167259436037184618 57
35759835115230164590440369761323328723122712568471082020972515710172693132346967854 25
80656697935045997268352998638215525166389437335543602135433229604645318478604952148 19
35558536110195962306 56

Время работы операции разница: 9 мкс.

Умножение первого элемента на второй по модулю: 500000000000000000000

Время работы операции умножения: 38 мкс.

Возведение первого элемента в квадрат по модулю (моя функция): 2500000000000000000000

Время работы операции возведения в квадрат первого элемента: 71 мкс.

Возведение первого элемента в квадрат по модулю (встроенная функция):
2500000000000000000000

Время работы операции возведения в квадрат первого элемента: 67 мкс.

Целая часть от деления a на b: 0

Время работы операции деление: 2 мкс.

Тест на корректность (проверка дистрибутивности):

Наш первый элемент (a):
32317006071311007300714876688669951960444102669715484032130345427524655138867890893197
20141152291346368871796092189801949411955915049092109508815238644828312063087736730 0
99609175019775038965210679605763838406756827679221864261975616183809433847617047058 16
45852036305042887575891541065808607552399123930385521914333389668342420684974786564 56
94948561760353263220580778056593310261927084603141502585928641771167259436037184618 57
35759835115230164590440369761323328723122712568471082020972515710172693132346967854 25
80656697935045997268352998638215525166389437335543602135433229604645318478604952148 19
35558536110095962306 56

Наш второй элемент (b):
32317006071311007300714876688669951960444102669715484032130345427524655138867890893197
20141152291346368871796092189801949411955915049092109508815238644828312063087736730 0
99609175019775038965210679605763838406756827679221864261975616183809433847617047058 16
45852036305042887575891541065808607552399123930385521914333389668342420684974786564 56
94948561760353263220580778056593310261927084603141502585928641771167259436037184618 57
35759835115230164590440369761323328723122712568471082020972515710172693132346967854 25
80656697935045997268352998638215525166389437335543602135433229604645318478604952148 19
35558536110495962306 56

Наш третий элемент (c): 499

Результат (a+b)*c :
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558535811195962306 56

Результат b*c+c*a :
323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558535811195962306 56
