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 головною особливістю .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.

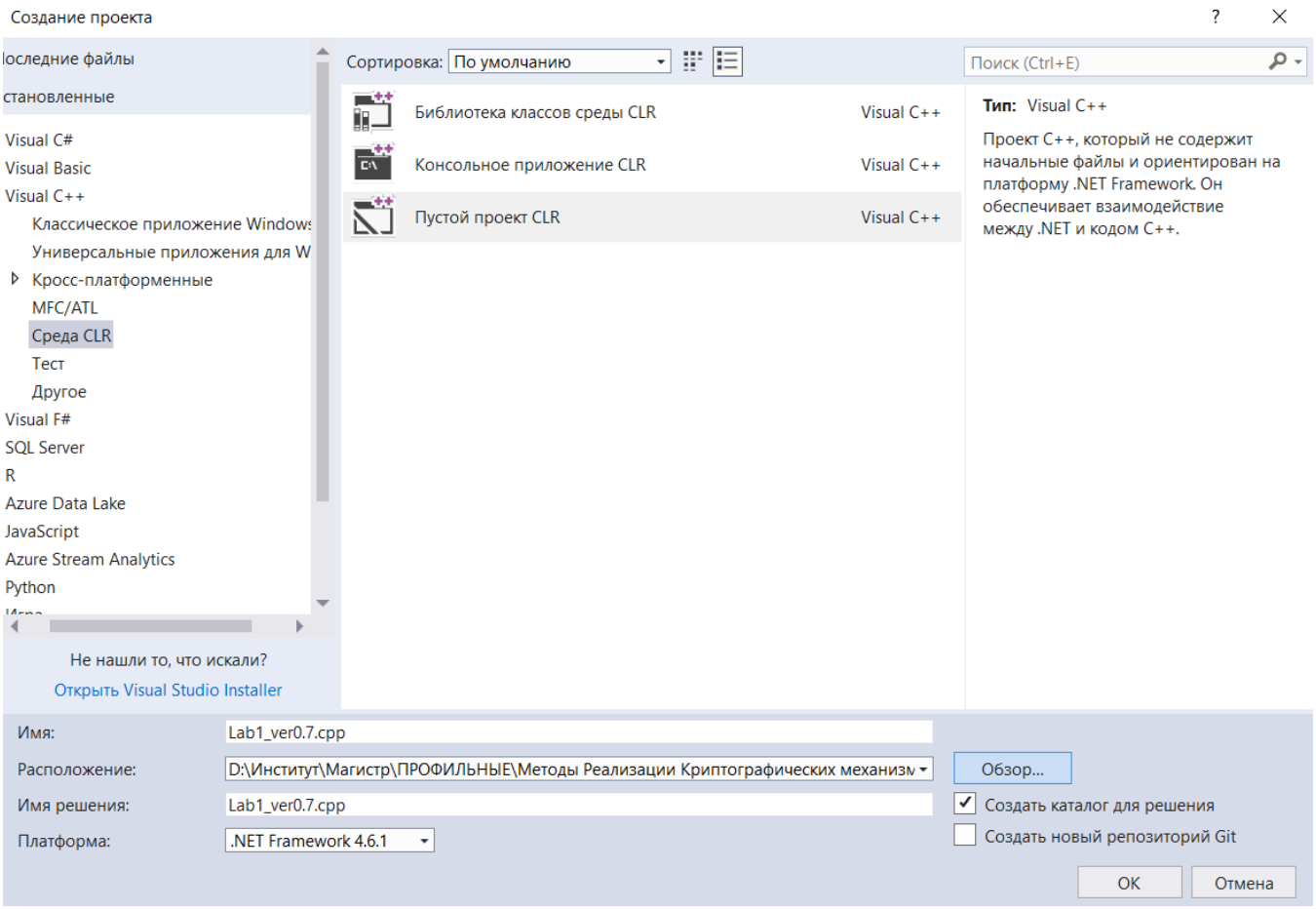


Рис.1. Створення CLR проекту.

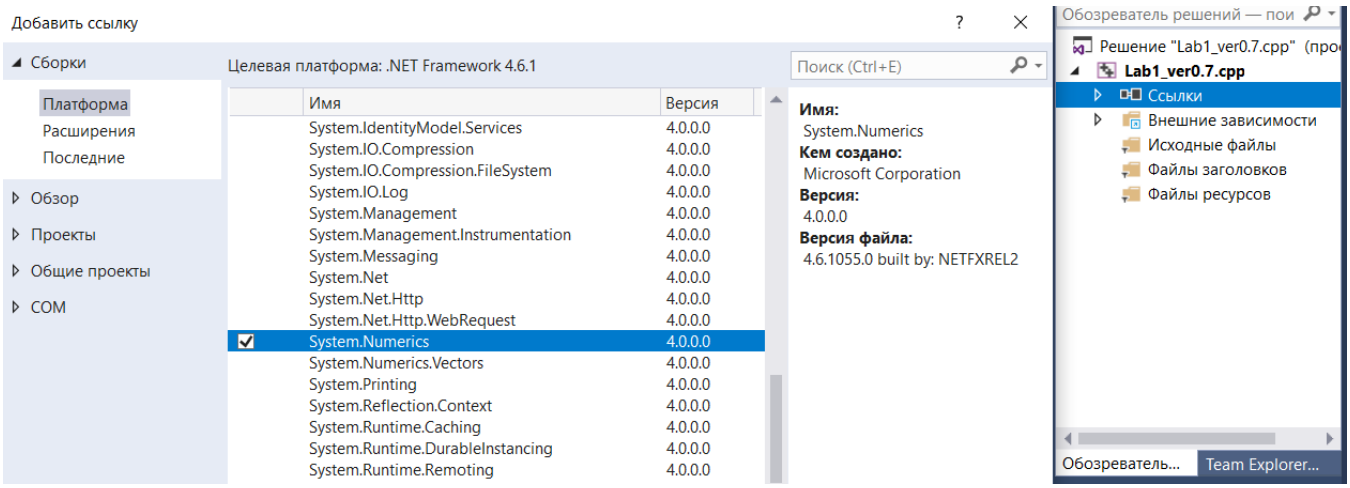


Рис.2. Підключення посилання на System::Numerics.

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 $A^B mod C$;
- 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;
- 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) для 256, 512, 1024 та 2048-бітових чисел.

Окрім цього, реалізуємо 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 також 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. Для підрахунку середніх часових витрат використовувався 1 млрд. запусків кожної базової операції (методів Add, Subtract, Multiply, Divide) для 256, 512, 1024 та 2048-бітових чисел.

**Код програми:**

```
#include <iostream>
#include <string>
#include <chrono>
#include <windows.h>

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(hStd
Out, 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(hStd
Out, 7);  // Меняем цвет текста на серый
                cout << message;  // Выводим
сообщение
                SetConsoleTextAttribute(hStd
Out, 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 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);
                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, BigInteger(1000000000)) != 0) { // Пока не пройдет 1000 итераций, делаем

                start = system_clock::now();
                result = BigInteger::Add(a, b);
                end = system_clock::now();
                average_time_count_add = BigInteger::Add(average_time_count_add, BigInteger(elapsedTime2(start, end)));

                start = system_clock::now();
                result = BigInteger::Subtract(a, b);
                end = system_clock::now();
                average_time_count_sub = BigInteger::Add(average_time_count_sub, BigInteger(elapsedTime2(start, end)));

                start = system_clock::now();
                result = BigInteger::Multiply(a, b);
                end = system_clock::now();
                average_time_count_mul = BigInteger::Add(average_time_count_mul, BigInteger(elapsedTime2(start, end)));

                start = system_clock::now();
                result = BigInteger::Divide(a, b);
                end = system_clock::now();
                average_time_count_div = BigInteger::Add(average_time_count_div, 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(1000000000));
                average_time_count_sub = BigInteger::Divide(average_time_count_sub, BigInteger(1000000000));
                average_time_count_mul = BigInteger::Divide(average_time_count_mul, BigInteger(1000000000));
                average_time_count_div = BigInteger::Divide(average_time_count_div, BigInteger(1000000000));
                print("Время работы операции Add: ", average_time_count_add);
                print("Время работы операции Subtract: ", average_time_count_sub);
                print("Время работы операции Multiply: ", average_time_count_mul);
                print("Время работы операции Divide: ", average_time_count_div);
}


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);

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

cout << "Замеры среднего времени базовых операций по модулю 2^256 для 1000000000 итераций (в мкс): " << endl;

average_time_count(a_256, b_256, mod_256);

cout << endl << "Замеры среднего времени базовых операций по модулю 2^512 для 1000000000 итераций (в мкс): " << endl;

average_time_count(a_512, b_512, mod_512);

cout << endl << "Замеры среднего времени базовых операций по модулю 2^1024 для 1000000000 итераций (в мкс): " << endl;

average_time_count(a_1024, b_1024, mod_1024);

cout << endl << "Замеры среднего времени базовых операций по модулю 2^2048 для 1000000000 итераций (в мкс): " << endl;

average_time_count(a_2048, b_2048, mod_2048);

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


BigInteger a_bigInt = BigInteger::Parse("17BC9E6E5CA3078D067DDB15CB2DF7B7D5E6437F99FE5FBB91750C3D2E4AFC7FCABC5F8795D0570EBF7A90AA0603D88FB10169B55D592959BBF88E2F141B16C3", NumberStyles::AllowHexSpecifier);

BigInteger b_bigInt = BigInteger::Parse("1F2A11C94AB245C494806E38BDADD88C3EBFD5E688A51FDDFDFFE23069EFF0BAE0BC82B1B2D07E6B56764F01B331B47D7DDE8641778784A6AEF78BC72F337B3E", NumberStyles::AllowHexSpecifier);
```

```
                BigInteger mod_bigInt =
BigInteger::Parse("10C6E1E34644B70666262C3DC6BD3
21A8C8A65753C6D90F4628EB5866FA0B4D494CC9294
BF4A7F252B74521DFBC9D290AFB73B7844D896C658F
5A76423D074D8", 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();

}
```

**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 :
1157920892373161954235709850086879078532699846656405640394575839779731296
39936

Результат b*c+c*a :
1157920892373161954235709850086879078532699846656405640394575839779731296
39936

Первый элемент (a):
1340780792994259709957402499820584612747936582059239337772356144372176403
0073546976801874298166903427690031858186486050853753882811946569946433599006084096

Второй элемент (b):
1340780792994259709957402499820584612747936582059239337772356144372176403
0073546976801874298166903427690031858186486050853753882811946569946433639006084096

Третий элемент (c):
1340780792994259709957402499820584612747936582059239337772356144372176403
0073546976801874298166903427690031858186486050853753882811946569946433649006084596

Модуль (mod):
1340780792994259709957402499820584612747936582059239337772356144372176403
0073546976801874298166903427690031858186486050853753882811946569946433649006084096

Третий элемент по модулю:500

Время работы операции взятия по модулю: 2 мкс.

Сумма первого и третьего по модулю:
1340780792994259709957402499820584612747936582059239337772356144372176403
0073546976801874298166903427690031858186486050853753882811946569946433599006084596

Время работы операции сложения: 2 мкс.

Разница первого элемента и второго по модулю:
134078079299425970995740249982058461274793658205923933777235614437217640
300735469768018742981669034276900318581864860508537538828119465699464336
09006084096

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):
250000000000000000000

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

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

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):
134078079299425970995740249982058461274793658205923933777235614437217640
300735469768018742981669034276900318581864860508537538828119465699464335
99006084096

Наш второй элемент (b):
134078079299425970995740249982058461274793658205923933777235614437217640
300735469768018742981669034276900318581864860508537538828119465699464336
39006084096

Наш третий элемент (c): 499

Результат (a+b)*c :
134078079299425970995740249982058461274793658205923933777235614437217640

300735469768018742981669034276900318581864860508537538828119465699464037
09006084096

Результат b*c+c*a :
134078079299425970995740249982058461274793658205923933777235614437217640
300735469768018742981669034276900318581864860508537538828119465699464037
09006084096

Первый элемент (a):
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329574224137216

Второй элемент (b):
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329614224137216

Третий элемент (c):
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329624224137216

Модуль (mod):
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329624224137216

Третий элемент по модулю:500

Время работы операции взятия по модулю: 4 мкс.

Сумма первого и третьего по модулю:
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329574224137716

Время работы операции сложения: 5 мкс.

Разница первого элемента и второго по модулю:
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329584224137216

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):
250000000000000000000

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):
179769313486231590772930519078902473361797697894230657273430081157732675

805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329574224137216

Наш второй элемент (b):
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329614224137216

Наш третий элемент (c): 499

Результат (a+b)*c :
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356299684224137216

Результат b*c+c*a :
179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356299684224137216

Первый элемент (a):
323170060713110073007148766886699519604441026697154840321303454275246551
388678908931972014115229134636887179609218980194941195591504909210950881
523864482831206308773673009960917501977503896521067960576383840675682767
922186426197561618380943384761704705816458520363050428875758915410658086
075523991239303855219143333896683424206849747865645694948561760353263220
580778056593310261927084603141502585928641771167259436037184618573575983
511523016459044036976132332872312271256847108202097251571017269313234696
785425806566979350459972683529986382155251663894373355436021354332296046
453184786049521481935558536110095962306 56

Второй элемент (b):
323170060713110073007148766886699519604441026697154840321303454275246551
388678908931972014115229134636887179609218980194941195591504909210950881
523864482831206308773673009960917501977503896521067960576383840675682767
922186426197561618380943384761704705816458520363050428875758915410658086

075523991239303855219143333896683424206849747865645694948561760353263220
580778056593310261927084603141502585928641771167259436037184618573575983
511523016459044036976132332872312271256847108202097251571017269313234696
785425806566979350459972683529986382155251663894373355436021354332296046
453184786049521481935558536110495962306 56

Третий элемент (c):
323170060713110073007148766886699519604441026697154840321303454275246551
388678908931972014115229134636887179609218980194941195591504909210950881
523864482831206308773673009960917501977503896521067960576383840675682767
922186426197561618380943384761704705816458520363050428875758915410658086
075523991239303855219143333896683424206849747865645694948561760353263220
580778056593310261927084603141502585928641771167259436037184618573575983
511523016459044036976132332872312271256847108202097251571017269313234696
785425806566979350459972683529986382155251663894373355436021354332296046
453184786049521481935558536110595962311 56

Модуль (mod):
323170060713110073007148766886699519604441026697154840321303454275246551
388678908931972014115229134636887179609218980194941195591504909210950881
523864482831206308773673009960917501977503896521067960576383840675682767
922186426197561618380943384761704705816458520363050428875758915410658086
075523991239303855219143333896683424206849747865645694948561760353263220
580778056593310261927084603141502585928641771167259436037184618573575983
511523016459044036976132332872312271256847108202097251571017269313234696
785425806566979350459972683529986382155251663894373355436021354332296046
453184786049521481935558536110595962306 56

Третий элемент по модулю:500

Время работы операции взятия по модулю: 4 мкс.

Сумма первого и третьего по модулю:
323170060713110073007148766886699519604441026697154840321303454275246551
388678908931972014115229134636887179609218980194941195591504909210950881
523864482831206308773673009960917501977503896521067960576383840675682767
922186426197561618380943384761704705816458520363050428875758915410658086
075523991239303855219143333896683424206849747865645694948561760353263220
580778056593310261927084603141502585928641771167259436037184618573575983
511523016459044036976132332872312271256847108202097251571017269313234696
785425806566979350459972683529986382155251663894373355436021354332296046
453184786049521481935558536110095962311 56

Время работы операции сложения: 5 мкс.

Разница первого элемента и второго по модулю:
32317006071311007300714876688669951960444102669715484032130345427524655138867890893197201411522913463688717960921898019494119559150490921095088152386448283120630877367300996091750197750389652106796057638384067568276792218642619756161838094338476170470581645852036305042887575891541065808607552399123930385521914333389668342420684974786564569494856176035326322058077805659331026192708460314150258592864177116725943603718461857357598351152301645904403697613233287231227125684710820209725157101726931323469678542580656697935045997268352998638215525166389437335543602135433229604645318478604952148193555853611019596230656

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):
250000000000000000000

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

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

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):
32317006071311007300714876688669951960444102669715484032130345427524655138867890893197201411522913463688717960921898019494119559150490921095088152386448283120630877367300996091750197750389652106796057638384067568276792218642619756161838094338476170470581645852036305042887575891541065808619

0755239912393038552191433338966834242068497478656456949485617603532632205807780565933102619270846031415025859286417711672594360371846185735759835115230164590440369761323328723122712568471082020972515710172693132346967854258065669793504599726835299863821552516638943733554360213543322960464531847860495214819355585361100959623065 6

Наш второй элемент (b):
3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393038552191433338966834242068497478656456949485617603532632205807780565933102619270846031415025859286417711672594360371846185735759835115230164590440369761323328723122712568471082020972515710172693132346967854258065669793504599726835299863821552516638943733554360213543322960464531847860495214819355585361104959623065 6

Наш третий элемент (c): 499

Результат (a+b)*c :
3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393038552191433338966834242068497478656456949485617603532632205807780565933102619270846031415025859286417711672594360371846185735759835115230164590440369761323328723122712568471082020972515710172693132346967854258065669793504599726835299863821552516638943733554360213543322960464531847860495214819355585361104959623065 6

Результат b*c+c*a :
3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393038552191433338966834242068497478656456949485617603532632205807780565933102619270846031415025859286417711672594360371846185735759835115230164590440369761323328723122712568471082020972515710172693132346967854258065669793504599726835299863821552516638943733554360213543322960464531847860495214819355585358111959623065 6

---

Замеры среднего времени базовых операций по модулю 2^256 для 1000000000 итераций (в мкс):

Время работы операции Add: 0

Время работы операции Subtract: 0

Время работы операции Multiply: 0

Время работы операции Divide: 0

Замеры среднего времени базовых операций по модулю 2^512 для 1000000000 итераций (в мкс):

Время работы операции Add: 0

Время работы операции Subtract: 0

Время работы операции Multiply: 0

Время работы операции Divide: 0

Замеры среднего времени базовых операций по модулю 2^1024 для 1000000000 итераций (в мкс):

Время работы операции Add: 0

Время работы операции Subtract: 0

Время работы операции Multiply: 2

Время работы операции Divide: 0

Замеры среднего времени базовых операций по модулю 2^2048 для 1000000000 итераций (в мкс):

Время работы операции Add: 0

Время работы операции Subtract: 0

Время работы операции Multiply: 11

Время работы операции Divide: 0

A:
124319669635590774014750025620478947808700165819011435478996397708647173949498634754583078575183243707870303821892726287005996019793386789079518018083808 3

B:
163220860609423562100402440339663746322765174304323474914781656310832121560480213270624439117097610373549993618285673520260759915351572100532048552391353 4

N:
878676726604422491045404036048823798868212919670382876756929076385186148
871029440376747104253270514609775175198537825961735875892555537715429395
592672472

A+B:
287540530245014336115152465960142694131465340123334910393778054019479295
509978848025207517692280854081420297440178399807266755935144958889611566
5704751617

(A+B)modN:
239375122636875888015312551454955544710014642222200473666993311039234508
486700159121833864162996996984877448806170520187459931673782975749827478
926734201

A-B: -
389011909738327880856524147191847985140650084853120394357852586021849476
109815785160413605419143666656796897963929472332547638955581853114525305
343075451

(A-B)modN:
489664816866094610188879888856975813727562834817262482399076490363336672
761213655216333498834126847952978277234608353629188236936973684600904090
249597021

A*B:
202915634686003486512201215548367179090574726933472204080290261804027103
155682278204782970343833802785435601421040162145183301026707160337752246
741571629768620891203751066696306314966528846089507001168314479622213730
942098206498356359989442536909370447461962823446435896686748281531518291
8226739894146315322

(A*B)modN:
548325362011551653253266777893364811017857538037360381387307581549488759
873650187359287549718430089665773675396802847698965879849515629743676207
790858498

B/A: 1

(B/A)modN: 1

Rem_(B/A)modN:
389011909738327880856524147191847985140650084853120394357852586021849476
109815785160413605419143666656796897963929472332547638955581853114525305
343075451

**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 великих чисел, тобто чисел, 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 $10^{-6}$ с).