|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №3 по курсу «Теория информации»  «Арифметика чисел большой разрядности» | |
| Выполнил | Студент гр. 21-ИТ-1  Макеёнок Д.И. |
| Проверила | Васильева Д.М. |
| Полоцк, 2023г. | |

**Ход работы**

Размерность обрабатываемых в вычислительных машинах чисел обычно ограничивается размерностью машинного слова. Типичная переменная целочисленного типа занимает в памяти машины 8, 16, 32 или 64 бит. Для многих криптографических алгоритмов требуются числа намного большего размера. Например, рекомендуемый размер открытого ключа для алгоритма RSA составляет 4 Кбит. Рассмотрим реализацию базовых арифметических операций над целыми числами большого размера. Для представления цифр больших чисел удобно использовать систему счисления с основанием b, равным 2 m , где m – размер машинного слова. Это наиболее компактный способ представления больших чисел, позволяющий хранить все цифры в массиве слов-переменных.

**Задание:**

1. Реализуйте алгоритмы «в столбик» для вычисления суммы, произведения и частного двух целых чисел большой разрядности.

2. Реализуйте алгоритмы Карацубы для умножения целых чисел большой разрядности.

3. Сравните скорость работы и затраты памяти для реализованных в заданиях 1 и 2 алгоритмов умножения целых чисел большой разрядности.

Листинг 1 - реализация программы:

using System.Numerics;

int ToInt(char c)

{

return c - '0';

}

string RowSum(string firstNumber, string secondNumber)

{

var result = string.Empty;

while (firstNumber.Length > secondNumber.Length)

secondNumber = "0" + secondNumber;

while (secondNumber.Length > firstNumber.Length)

firstNumber = "0" + firstNumber;

var mem = 0;

for (var i = firstNumber.Length - 1; i >= 0; i--)

{

var res = (ToInt(firstNumber[i]) + ToInt(secondNumber[i])).ToString();

result = (ToInt(res[^1]) + mem) + result;

if (res.Length > 1)

mem = ToInt(res[0]);

}

if (mem != 0)

result = mem + result;

return result;

}

string RowMultiply(string firstNumber, string secondNumber)

{

var result = string.Empty;

var mem = 0;

if (firstNumber.Length > secondNumber.Length)

(firstNumber, secondNumber) = (secondNumber, firstNumber);

foreach (var na in firstNumber.Reverse())

{

foreach (var nb in secondNumber.Reverse())

{

var res = (ToInt(nb) \* ToInt(na)).ToString();

result = (ToInt(res[^1]) + mem) + result;

if (res.Length > 1)

mem = ToInt(res[0]);

}

}

if (mem != 0)

result = mem + result;

return result;

}

string RowDivide(string firstNumber, string secondNumber)

{

var result = string.Empty;

var mem = 0;

if (firstNumber.Length < secondNumber.Length)

(firstNumber, secondNumber) = (secondNumber, firstNumber);

var i = 0;

while (firstNumber.Length > 0)

{

i = 0;

var s = firstNumber[i].ToString();

while (Compare(s, secondNumber) != 1)

{

i += 1;

s += firstNumber[i];

}

var k = 1;

var temp = RowMultiply(secondNumber, k.ToString());

while (Compare(s, temp) != -1)

{

k++;

temp = RowMultiply(secondNumber, k.ToString());

}

var minus = int.Parse(s) - int.Parse(RowMultiply(secondNumber, (k - 1).ToString()));

firstNumber = firstNumber.Replace(s, minus + s[(i + 1)..]);

if (firstNumber.First() == '0')

firstNumber = firstNumber[1..];

result += (k - 1);

if (firstNumber.All(x => x == '0'))

{

for (var l = 0; l < firstNumber.Length; l++)

result += '0';

break;

}

}

if (mem != 0)

result = mem + result;

return result;

}

int Compare(string firstNumber, string secondNumber)

{

if (firstNumber.Length > secondNumber.Length)

return 1;

if (secondNumber.Length > firstNumber.Length)

return -1;

for (var i = 0; i < firstNumber.Length; i++)

{

if (ToInt(firstNumber[i]) > ToInt(secondNumber[i]))

return 1;

if (ToInt(secondNumber[i]) > ToInt(firstNumber[i]))

return -1;

}

return 0;

}

BigInteger Karatsuba(BigInteger x, BigInteger y)

{

var n = Math.Max(x.GetBitLength(), y.GetBitLength());

if (n <= 2000)

return x \* y;

n = (n / 2) + (n % 2);

var b = x >> (int)n;

var a = x - b << (int)(n);

var d = y >> (int)(n);

var c = y - d << (int)(n);

var ac = Karatsuba(a, c);

var bd = Karatsuba(b, d);

var abcd = Karatsuba(BigInteger.Add(a, b), BigInteger.Add(c, d));

return BigInteger.Add(

BigInteger.Add(ac, abcd - ac - ((bd) << (int)(n))),

bd << (int)(2 \* n));

}

Console.WriteLine("Сумма столбиком: " + RowSum("1234", "1234"));

Console.WriteLine("Деление столбиком : " + RowDivide("503222", "2"));

Console.WriteLine("Умножение столбиком : " + RowMultiply("1234", "1234"));

Console.WriteLine(Karatsuba(19412343463546, 3456576587435647647));

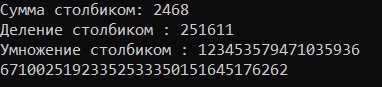


Рисунок 1. Результат работы программы