|  |  |
| --- | --- |
| 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  Катушёнок И.В. |
| Проверила | Васильева Д.М. |
| Полоцк, 2023г. | |

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:

class Calculate

{

public string Reverse(string s)

{

char[] charArray = s.ToCharArray();

Array.Reverse(charArray);

return new string(charArray);

}

public int ToInt(char charNumber)

{

return charNumber - '0';

}

public char ToChar(int intNumber)

{

int temp = intNumber + '0';

return (char)temp;

}

public int Sum(string firstNumber, string secondNumber)

{

while(firstNumber.Length < secondNumber.Length)

{

string temp = "";

temp = "0" + firstNumber;

firstNumber = temp;

}

while (secondNumber.Length < firstNumber.Length)

{

string temp = "";

temp = "0" + secondNumber;

secondNumber = temp;

}

int result = 0;

firstNumber = Reverse(firstNumber);

secondNumber = Reverse(secondNumber);

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

{

result += (ToInt(firstNumber[i]) + ToInt(secondNumber[i])) \* (Int32)Math.Pow(10,i);

}

return result;

}

public int Multiple(string firstNumber, string secondNumber)

{

int result = 0;

firstNumber = Reverse(firstNumber);

secondNumber = Reverse(secondNumber);

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

{

for(int j = 0; j < secondNumber.Length; j++)

{

result += ToInt(firstNumber[i]) \* ToInt(secondNumber[j]) \* (Int32)Math.Pow(10, i) \* (Int32)Math.Pow(10, j);

}

}

return result;

}

public string Divide(string firstNumber, string secondNumber)

{

string result = "";

string dividend = "0";

int divider = 0;

string remainder = "";

for(int i = 0; i <= firstNumber.Length - 1; i++)

{

for(int j = 0; int.Parse(dividend) < int.Parse(secondNumber); j++)

{

if (dividend == "0") dividend = "";

if (remainder == "0") remainder = "";

int divideWithReminder = 0;

int.TryParse(dividend, out divideWithReminder);

if(divideWithReminder < int.Parse(secondNumber) && i > firstNumber.Length - 1)

{

for (int a = 0; a < dividend.Length; a++)

{

result += '0';

}

return result;

}

dividend += remainder + firstNumber[i];

if (int.Parse(dividend) > int.Parse(secondNumber) || int.Parse(dividend) == 0) break;

i++;

}

char resultNumber = ' ';

for(int k = 0; divider <= int.Parse(dividend); k++)

{

divider += int.Parse(secondNumber);

resultNumber = ToChar(k);

}

divider -= int.Parse(secondNumber);

result += resultNumber;

remainder = Convert.ToString(int.Parse(dividend) - divider);

divider = 0;

dividend = "0";

}

return result;

}

public BigInteger KaratsubaMultiple(BigInteger x, BigInteger y)

{

int n = (int)Math.Max(BigInteger.Log(x, 2), BigInteger.Log(y, 2));

if (n <= 100) return x \* y;

n = ((n + 1) / 2);

BigInteger b = x >> n;

BigInteger a = x - (b << n);

BigInteger d = y >> n;

BigInteger c = y - (d << n);

BigInteger ac = KaratsubaMultiple(a, c);

BigInteger bd = KaratsubaMultiple(b, d);

BigInteger abcd = KaratsubaMultiple(a + b, c + d);

return ac + ((abcd - ac - bd) << n) + (bd << (2 \* n));

}

}

class Program

{

static void Main()

{

Calculate calc = new Calculate();

string sum1 = "110", sum2 = "321";

Console.WriteLine($"Sum {sum1} + {sum2} : " + calc.Sum(sum1,sum2));

string mult1 = "3245364", mult2 = "12536432";

Console.WriteLine($"Multiple {mult1} \* {mult2} : " + calc.Multiple("12345","12345"));

string div1 = "4567176", div2 = "14124";

Console.WriteLine($"Divide {div1} / {div2} : " + calc.Divide(div1,div2));

BigInteger karatsuba1 = 8475647382938476364, karatsuba2 = 9485748394857674832;

Console.WriteLine($"Karatsuba Multiple {karatsuba1} \* {karatsuba2} : " + calc.KaratsubaMultiple(karatsuba1,karatsuba2));

}

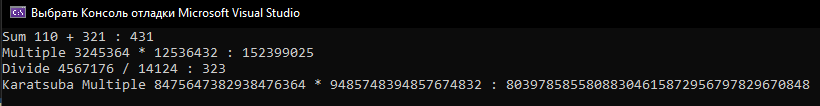


Рисунок 1. – результат работы программы