

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе №8.
Целочисленная арифметика многократной
ТОЧНОСТИ

*Дисциплина: Математические основы защиты
информации и информационной безопасности*

Студент: Асеинова Елизавета, 1132236897
Группа: НФИмд-01-23
Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2023

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Арифметика многократной точности	7
4	Выполнение лабораторной работы	8
4.1	Вспомогательные действия	8
4.2	Алгоритм 1. Сложение неотрицательных целых чисел. Реализация	9
4.3	Алгоритм 2. Вычитание неотрицательных целых чисел. Реализация	10
4.4	Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Реализация	11
4.5	Алгоритм 4. Быстрый столбик. Реализация	12
4.6	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	13
4.7	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	14
5	Выводы	15
	Список литературы	16

Список таблиц

Список иллюстраций

4.1	Вспомогательные действия для удобства дальнейших вычислений	8
4.2	Алгоритм 1. Сложение неотрицательных целых чисел	9
4.3	Алгоритм 2. Вычитание неотрицательных целых чисел	10
4.4	Алгоритм 3. Умножение неотрицательных целых чисел столбиком	11
4.5	Алгоритм 4. Быстрый столбик	12
4.6	Алгоритм 5. Деление многоразрядных целых чисел	13
4.7	Алгоритм 5. Деление многоразрядных целых чисел	14

1 Цель работы

Целью данной лабораторной работы является ознакомление с алгоритмами по воплощению целочисленной арифметики многократной точности, а также программная реализация данных алгоритмов.

2 Задание

Реализовать рассмотренные в инструкции к лабораторной работе алгоритмы программно.

Алгоритмы:

1. Сложение неотрицательных целых чисел
2. Вычитание неотрицательных целых чисел
3. Умножение неотрицательных целых чисел столбиком
4. Быстрый столбик
5. Деление многоразрядных целых чисел

3 Теоретическое введение

В данной лабораторной работе предметом нашего изучения стали алгоритмы по воплощению целочисленной арифметики многократной точности.

3.1 Арифметика многократной точности

Арифметика многократной точности — это операции (базовые арифметические действия, элементарные математические функции и пр.) над числами большой разрядности, т.е. числами, разрядность которых превышает длину машинного слова универсальных процессоров общего назначения (более 128 бит)

В современных асимметричных криптосистемах в качестве ключей, как правило, используются целые числа длиной 1000 и более битов. Для задания чисел такого размера не подходит ни один стандартный целочисленный тип данных современных языков программирования.

При работе с большими целыми числами знак такого числа удобно хранить в отдельной переменной. Например, при умножении двух чисел знак произведения вычисляется отдельно.

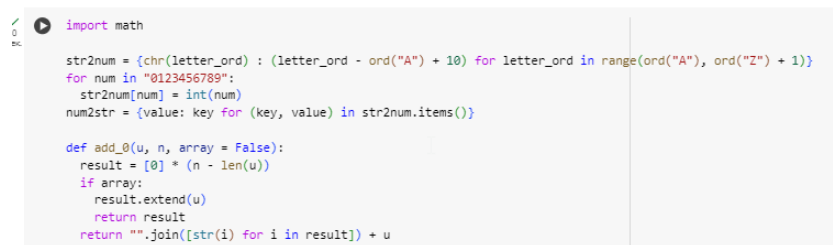
Далее нами были рассмотрены алгоритмы по воплощению целочисленной арифметики многократной точности.

4 Выполнение лабораторной работы

В соответствии с заданием, была написана программа по воплощению алгоритма Ро-метода Полларда для задач дискретного логарифмирования.

Программный код и результаты выполнения программ представлен ниже.

4.1 Вспомогательные действия



```
import math

str2num = {chr(letter_ord) : (letter_ord - ord("A") + 10) for letter_ord in range(ord("A"), ord("Z") + 1)}
for num in "0123456789":
    str2num[num] = int(num)
num2str = {value: key for (key, value) in str2num.items()}

def add_0(u, n, array = False):
    result = [0] * (n - len(u))
    if array:
        result.extend(u)
    return result
return "".join([str(i) for i in result]) + u
```

Рис. 4.1: Вспомогательные действия для удобства дальнейших вычислений

4.2 Алгоритм 1. Сложение неотрицательных целых чисел.

Реализация

✓ Алгоритм 1

```
def alg_1(u_s, v_s, b):  
    u = [str2num[letter] for letter in u_s]  
    v = [str2num[letter] for letter in v_s]  
    if len(u) != len(v):  
        if len(u) < len(v):  
            u = add_0(u, len(v), True)  
        else:  
            v = add_0(v, len(u), True)  
    n = len(u)  
    k = 0  
    w = []  
    for j in range(n-1, -1, -1):  
        w.append((u[j] + v[j] + k) % b)  
        k = math.floor((u[j] + v[j] + k) / b)  
    w.append(k)  
    w.reverse()  
    return "".join([num2str[num] for num in w])
```

```
[10] print(alg_1("221", "1567", 10))  
      print(alg_1("807", "M16", 10))
```

```
01788  
3323
```

Рис. 4.2: Алгоритм 1. Сложение неотрицательных целых чисел

4.3 Алгоритм 2. Вычитание неотрицательных целых чисел.

Реализация

✓ Алгоритм 2

```
✓ [11] def alg_2(u_s, v_s, b):  
0     u = [str2num[letter] for letter in u_s]  
JEC   v = [str2num[letter] for letter in v_s]  
      if len(u) != len(v):  
        if len(u) < len(v):  
          u = add_0(u, len(v), True)  
        else:  
          v = add_0(v, len(u), True)  
      elif u < v:  
        return "u должно быть больше v"  
      n = len(u)  
      k = 0  
      w = []  
      for j in range(n-1, -1, -1):  
        w.append((u[j] - v[j] + k) % b)  
        k = math.floor((u[j] - v[j] + k) / b)  
      w.append(k)  
      w.reverse()  
      return "".join([num2str[num] for num in w])  
  
✓ [13] alg_2("715", "215", 10)  
0  
JEC  
      '0500'
```

Рис. 4.3: Алгоритм 2. Вычитание неотрицательных целых чисел

4.4 Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Реализация

✓ Алгоритм 2

```
✓ [11] def alg_2(u_s, v_s, b):  
0   u = [str2num[letter] for letter in u_s]  
JEC v = [str2num[letter] for letter in v_s]  
    if len(u) != len(v):  
        if len(u) < len(v):  
            u = add_0(u, len(v), True)  
        else:  
            v = add_0(v, len(u), True)  
    elif u < v:  
        return "u должно быть больше v"  
    n = len(u)  
    k = 0  
    w = []  
    for j in range(n-1, -1, -1):  
        w.append((u[j] - v[j] + k) % b)  
        k = math.floor((u[j] - v[j] + k) / b)  
    w.append(k)  
    w.reverse()  
    return "".join([num2str[num] for num in w])  
  
✓ [13] alg_2("715", "215", 10)  
0  
JEC '0500'
```

Рис. 4.4: Алгоритм 3. Умножение неотрицательных целых чисел столбиком

4.5 Алгоритм 4. Быстрый столбик. Реализация

✓ Алгоритм 3

```
✓ [14] def alg_3(u_s, v_s, b):  
0   u = [str2num[letter] for letter in u_s]  
сек. v = [str2num[letter] for letter in v_s]  
n = len(u)  
m = len(v)  
w = [0] * (m+n)  
for j in range(m-1, -1, -1):  
    if v[j] != 0:  
        k=0  
        for i in range(n-1, -1, -1):  
            t = u[i] * v[j] + w[i+j+1] + k  
            w[i+j+1] = t%b  
            k = math.floor(t/b)  
        w[j] = k  
    return "".join([num2str[num] for num in w])
```

```
✓ [16] alg_3("650", "1000", 10)  
0  
сек. '0650000'
```

Рис. 4.5: Алгоритм 4. Быстрый столбик

4.6 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

Алгоритм 5

```
def to10(u_str, b, array = False):
    u_array = u_str if array else [str2num[letter] for letter in u_str]
    u = 0
    for i in range(len(u_array)):
        u += (b ** i) * u_array[len(u_array) - i - 1]
    return u

def to_b(number, b, n = 1):
    (q, r) = (math.floor(number / b), number % b)
    w = num2str[r]

    while q >= b:
        (q, r) = (math.floor(q / b), q % b)
        w = w + num2str[r]

    if q != 0: w = w + num2str[q]

    while len(w) < n:
        w = w + "0"

    return w[::-1]

def trim_zero(a):
    while a[0] == '0' and len(a) > 1:
        a = a[1:]
    return a

def algorithm_5(u_str, v_str, b):
    u = u_str
    v = v_str
    u_10 = to10(u, b)
    v_10 = to10(v, b)
    n = len(u) - 1
    t = len(v) - 1

    if v[0] == 0 or not (n >= t >= 1):
        return "Некорректные входные данные"

    q = [0] * (n - t + 1) # шаг 1

    while u_10 >= v_10 * (b ** (n - t)): #
        q[n - t] = q[n - t] + 1
        u_10 -= v_10 * b ** (n - t)
```

Рис. 4.6: Алгоритм 5. Деление многоразрядных целых чисел

4.7 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

[illegible]

Рис. 4.7: Алгоритм 5. Деление многоразрядных целых чисел

5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: в результате выполнения данной лабораторной работы нам удалось осуществить программно алгоритмы, рассмотренные в описании к лабораторной работе, а также мы осуществили программно данные алгоритмы.

Список литературы