

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: Лапшенкова Любовь Олеговна, 10322127633

Группа: НФИмд-02-21

Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2021

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Арифметика многократной точности	7
3.2	Сложение неотрицательных целых чисел	8
3.3	Вычитание неотрицательных целых чисел	8
3.4	Умножение неотрицательных целых чисел столбиком	9
3.5	Быстрый столбик	9
3.6	Деление многоразрядных целых чисел	10
4	Выполнение лабораторной работы	11
4.1	Вспомогательные действия	11
4.2	Алгоритм 1. Сложение неотрицательных целых чисел. Реализация	12
4.3	Алгоритм 1. Сложение неотрицательных целых чисел. Результат .	12
4.4	Алгоритм 2. Вычитание неотрицательных целых чисел. Реализация	13
4.5	Алгоритм 2. Вычитание неотрицательных целых чисел. Результат	13
4.6	Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Реализация	14
4.7	Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Результат	14
4.8	Алгоритм 4. Быстрый столбик. Реализация	15
4.9	Алгоритм 4. Быстрый столбик. Результат	15
4.10	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	16
4.11	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	16
4.12	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	17
4.13	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	17
4.14	Алгоритм 5. Деление многоразрядных целых чисел. Реализация .	18
4.15	Алгоритм 5. Деление многоразрядных целых чисел. Результат . .	18
5	Выводы	19
	Список литературы	20

List of Figures

3.1	Алгоритм 1. Сложение неотрицательных целых чисел	8
3.2	Алгоритм 2. Вычитание неотрицательных целых чисел	8
3.3	Алгоритм 3. Умножение неотрицательных целых чисел столбиком	9
3.4	Алгоритм 4. Быстрый столбик	9
3.5	Алгоритм 5. Деление многоразрядных целых чисел	10
4.1	Вспомогательные действия для удобства дальнейших вычислений	11
4.2	Алгоритм 1. Сложение неотрицательных целых чисел	12
4.3	Алгоритм 1. Сложение неотрицательных целых чисел	12
4.4	Алгоритм 2. Вычитание неотрицательных целых чисел	13
4.5	Алгоритм 2. Вычитание неотрицательных целых чисел	13
4.6	Алгоритм 3. Умножение неотрицательных целых чисел столбиком	14
4.7	Алгоритм 3. Умножение неотрицательных целых чисел столбиком	14
4.8	Алгоритм 4. Быстрый столбик	15
4.9	Алгоритм 4. Быстрый столбик	15
4.10	Алгоритм 5. Деление многоразрядных целых чисел	16
4.11	Алгоритм 5. Деление многоразрядных целых чисел	16
4.12	Алгоритм 5. Деление многоразрядных целых чисел	17
4.13	Алгоритм 5. Деление многоразрядных целых чисел	17
4.14	Алгоритм 5. Деление многоразрядных целых чисел	18
4.15	Алгоритм 5. Деление многоразрядных целых чисел	18

List of Tables

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) [1].

В современных асимметричных криптосистемах в качестве ключей, как правило, используются целые числа длиной 1000 и более битов [2]. 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 [3]. Например, при умножении двух чисел знак произведения вычисляется отдельно.

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

3.2 Сложение неотрицательных целых чисел

Вход. Два неотрицательных числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$; разрядность чисел n ; основание системы счисления b .

Выход. Сумма $w = w_0 w_1 \dots w_n$, где w_0 – цифра переноса – всегда равная 0 либо 1.

1. Присвоить $j := n, k := 0$ (j идет по разрядам, k следит за переносом).
2. Присвоить $w_j = (u_j + v_j + k) \pmod{b}$, где w_j – наименьший неотрицательный вычет в данном классе вычетов; $k = \left\lfloor \frac{u_j + v_j + k}{b} \right\rfloor$.
3. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то присвоить $w_0 := k$ и результат: w .

Figure 3.1: Алгоритм 1. Сложение неотрицательных целых чисел

3.3 Вычитание неотрицательных целых чисел

Вход. Два неотрицательных числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$, $u > v$; разрядность чисел n ; основание системы счисления b .

Выход. Разность $w = w_1 w_2 \dots w_n = u - v$.

1. Присвоить $j := n, k := 0$ (k – заем из старшего разряда).

31

-
2. Присвоить $w_j = (u_j - v_j + k) \pmod{b}$, где w_j – наименьший неотрицательный вычет в данном классе вычетов; $k = \left\lfloor \frac{u_j - v_j + k}{b} \right\rfloor$.
 3. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то результат: w .

Figure 3.2: Алгоритм 2. Вычитание неотрицательных целых чисел

3.4 Умножение неотрицательных целых чисел столбиком

Вход. Числа $u = u_1u_2 \dots u_n$, $v = v_1v_2 \dots v_m$; основание системы счисления b .

Выход. Произведение $w = uv = w_1w_2 \dots w_{m+n}$.

1. Выполнить присвоения: $w_{m+1} := 0, w_{m+2} := 0, \dots, w_{m+n} := 0, j := m$ (j перемещается по номерам разрядов числа v от младших к старшим).
2. Если $v_j = 0$, то присвоить $w_j := 0$ и перейти на шаг 6.
3. Присвоить $i := n, k := 0$ (Значение i идет по номерам разрядов числа u , k отвечает за перенос).
4. Присвоить $t := u_i \cdot v_j + w_{i+j} + k, w_{i+j} := t \pmod{b}, k := \frac{t}{b}$, где w_{i+j} — наименьший неотрицательный вычет в данном классе вычетов.
5. Присвоить $i := i - 1$. Если $i > 0$, то возвращаемся на шаг 4, иначе присвоить $w_j := k$.
6. Присвоить $j := j - 1$. Если $j > 0$, то вернуться на шаг 2. Если $j = 0$, то результат: w .

Figure 3.3: Алгоритм 3. Умножение неотрицательных целых чисел столбиком

3.5 Быстрый столбик

Вход. Числа $u = u_1u_2 \dots u_n$, $v = v_1v_2 \dots v_m$; основание системы счисления b .

Выход. Произведение $w = uv = w_1w_2 \dots w_{m+n}$.

1. Присвоить $t := 0$.
2. Для s от 0 до $m + n - 1$ с шагом 1 выполнить шаги 3 и 4.
3. Для i от 0 до s с шагом 1 выполнить присвоение $t := t + u_{n-i} \cdot v_{m-s+i}$.
4. Присвоить $w_{m+n-s} := t \pmod{b}, t := \frac{t}{b}$, где w_{m+n-s} — наименьший неотрицательный вычет по модулю b . Результат: w .

Figure 3.4: Алгоритм 4. Быстрый столбик

3.6 Деление многоразрядных целых чисел

Вход. Числа $u = u_n \dots u_1 u_0$, $v = v_t \dots v_1 v_0$, $n \geq t \geq 1$, $v_t \neq 0$, разрядность чисел соответственно n и t .

Выход. Частное $q = q_{n-t} \dots q_0$, остаток $r = r_t \dots r_0$.

1. Для j от 0 до $n - t$ присвоить $q_j := 0$.
2. Пока $u \geq vb^{n-t}$, выполнять: $q_{n-t} := q_{n-t} + 1$, $u := u - vb^{n-t}$.
3. Для $i = n, n - 1, \dots, t + 1$ выполнять пункты 3.1 – 3.4:
 - 3.1 если $u_i \geq v_t$, то присвоить $q_{i-t-1} := b - 1$, иначе присвоить $q_{i-t-1} := \frac{u_i b + u_{i-1}}{v_t}$.
 - 3.2 пока $q_{i-t-1}(v_t b + v_{t-1}) > u_i b^2 + u_{i-1} b + u_{i-2}$ выполнять $q_{i-t-1} := q_{i-t-1} - 1$.
 - 3.3 присвоить $u := u - q_{i-t-1} b^{i-t-1} v$.
 - 3.4 если $u < 0$, то присвоить $u := u + v b^{i-t-1}$, $q_{i-t-1} := q_{i-t-1} - 1$.
4. $r := u$. Результат: q и r .

Figure 3.5: Алгоритм 5. Деление многоразрядных целых чисел

4 Выполнение лабораторной работы

Примечание: комментарии по коду представлены на скриншотах к каждому из проделанных заданий.

В соответствии с заданием, были написаны программы по воплощению алгоритмов, представленных в описании к лабораторной работе.

Программный код и результаты выполнения программ представлены ниже.

4.1 Вспомогательные действия

```
import math

#создаем словарь 1 (символ строки = аналог в числ. форме)
str2num = {chr(letter_ord) : (letter_ord - ord("A") + 10) for letter_ord in range(ord("A"), ord("Z") + 1)}
for ciphra in "0123456789":
    str2num[ciphra]=int(ciphra)
#создаем словарь 2 (число = строковый аналог)
num2str={value:key for (key,value)in str2num.items()}

def add_0(u, n, array = False):
    """
    Функция, отвечающая за добавление нулей к числу u до размерности n
    array=True если u-массив чисел
    """
    result=[0]*(n-len(u))
    if array:
        result.extend(u)
        return result
    return "".join([str(i) for i in result])+u
```

Figure 4.1: Вспомогательные действия для удобства дальнейших вычислений

4.2 Алгоритм 1. Сложение неотрицательных целых чисел.

Реализация

```
def algorith_1(u_s,v_s,b):  
    '''  
    Сложение неотрицательных целых чисел  
    '''  
  
    #представить u и v в виде массивов чисел  
    u=[str2num[letter]for letter in u_s]  
    v=[str2num[letter]for letter in v_s]  
    #проверка на совпадение разрядностей чисел  
    if len(u)!=len(v):  
        #добавить к меньшему нули  
        if len(u)<len(v):  
            u=add_0(u, len(v), True)  
        else:  
            v=add_0(v, len(u), True)  
    #шаг 1  
    n=len(u)#разрядность числа  
    k=0#счетчик, который отвечает за перенос  
    w=[]#будущая сумма  
  
    for j in range(n-1, -1, -1):  
        '''  
        шаги 2-3  
        '''  
        w.append((u[j]+v[j]+k)%b)  
        k=math.floor((u[j]+v[j]+k)/b)  
    #шаг 3  
    w.append(k)  
    w.reverse()#записываем массив в обратном порядке  
    return "".join([num2str[ciphra] for ciphra in w]) #массив в строку
```

Figure 4.2: Алгоритм 1. Сложение неотрицательных целых чисел

4.3 Алгоритм 1. Сложение неотрицательных целых чисел.

Результат

```
print(algorith_1("321","1567",10))  
print(algorith_1("B007","MI6",10))
```

```
01888  
13393
```

Figure 4.3: Алгоритм 1. Сложение неотрицательных целых чисел

4.4 Алгоритм 2. Вычитание неотрицательных целых чисел. Реализация

```
def algorythm_2(u_s,v_s,b):  
    '''  
    Вычитание неотрицательных целых чисел  
    '''  
  
    #представить u и v в виде массивов чисел  
    u=[str2num[letter]for letter in u_s]  
    v=[str2num[letter]for letter in v_s]  
    #проверка на совпадение разрядностей чисел  
    if len(u)!=len(v):  
        #добавить к меньшему нули  
        if len(u)<len(v):  
            u=add_0(u, len(v), True)  
        else:  
            v=add_0(v, len(u), True)  
    elif u<v:#проверка на удовлетворение условию задачи  
        return "u должно быть больше v"  
    #шаг1  
    n=len(u)#разрядность числа  
    k=0#счетчик, который отвечает за перенос  
    w=[]#будущая сумма  
    for j in range(n-1, -1, -1):  
        '''  
        шаги 2-3  
        '''  
        w.append((u[j]-v[j]+k)%b)  
        k=math.floor((u[j]-v[j]+k)/b)  
    w.reverse()#записываем массив в обратном порядке  
    return "".join([num2str[ciphra] for ciphra in w]) #массив в строку
```

Figure 4.4: Алгоритм 2. Вычитание неотрицательных целых чисел

4.5 Алгоритм 2. Вычитание неотрицательных целых чисел. Результат

```
print(algorythm_2("789","111",10))
```

678

Figure 4.5: Алгоритм 2. Вычитание неотрицательных целых чисел

4.6 Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Реализация

```
def algorythm_3(u_s,v_s,b):  
    '''  
    Умножение неотрицательных чисел столбиком  
    '''  
  
    #представить u и v в виде массивов чисел  
    u=[str2num[letter]for letter in u_s]  
    v=[str2num[letter]for letter in v_s]  
    #выписали разрядности для u и для v  
    n=len(u)  
    m=len(v)  
    #произведение  
    w=[0]*(m+n)  
    for j in range(m-1, -1, -1):  
        if v[j]!=0:  
            k=0#шаг3  
            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#шаг 5  
    return "".join([num2str[ciphra] for ciphra in w]) #массив в строку
```

Figure 4.6: Алгоритм 3. Умножение неотрицательных целых чисел столбиком

4.7 Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Результат

```
print(algorythm_3("777","1234",10))  
  
0958818
```

Figure 4.7: Алгоритм 3. Умножение неотрицательных целых чисел столбиком

4.8 Алгоритм 4. Быстрый столбик. Реализация

```
def algorythm_4(u_s,v_s,b):  
    ...  
    Быстрый столбик  
    ...  
    #представить u и v в виде массивов чисел  
    u=[str2num[letter]for letter in u_s]  
    v=[str2num[letter]for letter in v_s]  
    #выписали разрядности для u и для v  
    n=len(u)  
    m=len(v)  
    #произведение  
    w=[0]*(m+n)  
  
    t=0#шаг1  
    for s in range(0,m+n):  
        ...  
        шаг2  
        ...  
        for i in range (0,s+1):  
            if (0<=n-i-1<n) and (0<=m-s+i-1<m): #шаг3  
                t=t+u[n-i-1]*v[m-s+i-1]  
            w[m+n-s-1]=t%b  
            t=math.floor(t/b)#шаг 4  
    return "".join([num2str[ciphra] for ciphra in w]) #массив в строку
```

Figure 4.8: Алгоритм 4. Быстрый столбик

4.9 Алгоритм 4. Быстрый столбик. Результат

```
print(algorythm_3("777","1234",10))
```

0958818

Figure 4.9: Алгоритм 4. Быстрый столбик

4.10 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

```
def to10(u_str, b, array = False):
    """
    Переводит число в десятичную систему исчисления
    array = True, если число u- массив чисел
    """
    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
```

Figure 4.10: Алгоритм 5. Деление многоразрядных целых чисел

4.11 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

```
def to_b(number, b, n = 1):
    """
    Переводит десятичное число в систему счисления с основанием b
    n - минимальная разрядность результирующей записи числа
    """
    # будем последовательно делить number на b и сохранять остатки
    # q - очередное частное, r - очередной остаток
    # w - результат, в который последовательно записываем остатки
    (q, r) = (math.floor(number / b), number % b)
    w = num2str[r]

    # пока частное больше основания системы счисления
    while q >= b:
        # продолжаем деление
        (q, r) = (math.floor(q / b), q % b)
        w = w + num2str[r]

    # если частное ненулевое, тоже добавляем его в результат
    if q != 0: w = w + num2str[q]

    # если разрядность меньше желаемой..
    while len(w) < n:
        # добавляем ведущие нули
        w = w + "0"

    # записываем число в обратном порядке
    return w[::-1]
```

Figure 4.11: Алгоритм 5. Деление многоразрядных целых чисел

4.12 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

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

Figure 4.12: Алгоритм 5. Деление многоразрядных целых чисел

4.13 Алгоритм 5. Деление многоразрядных целых чисел.

Реализация

```
def algorith_5(u_str, v_str, b):  
    """  
    деление многоразрядных целых чисел  
    Производит деление целых неотрицательных чисел,  
    Результат: (q, r), где q - частное, r - остаток  
    """  
    u = u_str  
    v = v_str  
    u_10 = to10(u, b)  
    v_10 = to10(v, b)  
    n = len(u) - 1 # разрядности чисел  
    t = len(v) - 1 # разрядности чисел  
  
    # проверка условий  
    if v[0] == 0 or not (n >= t >= 1):  
        return "Некорректные входные данные"  
  
    q = [0] * (n - t + 1) # инициализация  
  
    while u_10 >= v_10 * (b ** (n - t)): #  
        q[n - t] = q[n - t] + 1  
        u_10 -= v_10 * b ** (n - t)  
  
    u = to_b(u_10, b, n + 1)
```

Figure 4.13: Алгоритм 5. Деление многоразрядных целых чисел

4.14 Алгоритм 5. Деление многоразрядных целых чисел.

```

4  @param {number} n - width of matrix
5  @param {number} m - height of matrix
6  @return {number[][]}
7  """
8  # Create an empty matrix
9  matrix = [[0] * m for _ in range(n)]
10
11  # Fill the matrix with values
12  for i in range(n):
13      for j in range(m):
14          # Calculate the value for matrix[i][j]
15          # The value is the sum of the values at (i-1, j) and (i, j-1)
16          # If i=0 or j=0, the value is 1
17          if i == 0 or j == 0:
18              matrix[i][j] = 1
19          else:
20              matrix[i][j] = matrix[i-1][j] + matrix[i][j-1]
21
22  # Return the matrix
23  return matrix
24
25 # Example usage
26 n = 3
27 m = 3
28 result = num_paths(n, m)
29 print(result)
30
31 # Output:
32 [[1, 1, 1],
33 [1, 2, 3],
34 [1, 3, 6]]
35
36 # Explanation:
37 # There are 1 path from (0,0) to (0,0)
38 # There are 2 paths from (0,0) to (0,1)
39 # There are 3 paths from (0,0) to (0,2)
40 # There are 1 paths from (0,0) to (1,0)
41 # There are 2 paths from (0,0) to (1,1)
42 # There are 3 paths from (0,0) to (1,2)
43 # There are 1 paths from (0,0) to (2,0)
44 # There are 3 paths from (0,0) to (2,1)
45 # There are 6 paths from (0,0) to (2,2)
46
47 # Time complexity: O(n * m)
48 # Space complexity: O(n * m)

```

Figure 4.14: Алгоритм 5. Деление многоразрядных целых чисел

4.15 Алгоритм 5. Деление многоразрядных целых чисел.

```
print(algorythm_5("1000", "15", 10))  
  
( '66', '10' )
```

Figure 4.15: 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

1. Исупов К.С. Методы и алгоритмы организации высокоточных вычислений в арифметике остаточных классов для универсальных процессорных платформ: phdthesis. Вятский государственный университет, 2014.
2. Панкратова И.А. Теоретико-числовые методы в криптографии: учебное пособие. Томск: Томский государственный университет, 2009. С. 120.
3. Бубнов С.А. Лабораторный практикум по основам криптографии: учебно-методическое пособие. Саратов; http://elibrary.sgu.ru/uch_lit/656.pdf: Саратовский государственный университет им. Н.Г. Чернышевского, 2012.