**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: Леонова Алина Дмитриевна, 1032212306
Группа: НФИмд-01-21
Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2021

# Содержание

# List of Figures

# 1 Цель работы

Целью данной работы является ознакомление с 5 алгоримами для выполнения арифметических операций с большими целыми числами и их реализация на выбранном языке программирования.

# 2 Задание

Реализовать программно алгоритмы:

- сложение неотрицательных целых чисел
- вычитание неотрицательных целых чисел
- умножение неотрицательных целых чисел столбиком
- быстрый столбик
- деление многоразрядных целых чисел

# 3 Теоретическое введение

Рассмотрим алгоритмы для выполнения арифметических операций с большими целыми числами. Будем считать, что число записано в $b$-ичной системе счисления, $b$ – натуральное число, $b \geqslant 2$.

При работе с большими целыми числами знак такого числа удобно хранить в отдельной переменной. Например, при умножении двух чисел, знак произведения вычисляется отдельно. Квадратные скобки обозначают, что берется целая часть числа [1].

## 3.1 Алгоритм 1 (сложение неотрицательных целых чисел)

**Алгоритм 1 (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 $w_j = (u_j + v_j + k) \ (mod\ b)$, где $w_j$ – наименьший неотрицательный вычет в данном классе вычетов; $k = \left[\frac{u_j + v_j + k}{b}\right]$.
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

Описание алгоритма 1 (см. рис. 3.1).

## 3.2 Алгоритм 2 (вычитание неотрицательных целых чисел).

**Алгоритм 2 (вычитание неотрицательных целых чисел).**

*Вход.* Два неотрицательных числа $u = u_1u_2 \dots u_n$ и $v = v_1v_2 \dots v_n$, $u > v$; разрядность чисел $n$; основание системы счисления $b$.

*Выход.* Разность $w = w_1w_2 \dots w_n = u - v$.

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

Figure 3.2: Алгоритм 2

Описание алгоритма 2 (см. рис. 3.2).

## 3.3 Алгоритм 3 (умножение неотрицательных целых чисел столбиком).

**Алгоритм 3 (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 $t := u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} := t \ (mod\ b)$, $k := \frac{t}{b}$, 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 $w_{ij} := k$.
6. Присвоить $j := j - 1$. Если $j > 0$, то вернуться на шаг 2. Если $j = 0$, то результат: $w$.

Figure 3.3: Алгоритм 3

Описание алгоритма 3 (см. рис. 3.3).

## 3.4 Алгоритм 4 (быстрый столбик).

**Алгоритм 4 (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 $w_{m+n-s} := t \ (mod\ b)$, $t := \frac{t}{b}$, где $w_{m+n-s}$ – наименьший неотрицательный вычет по модулю $b$. Результат: $w$.

Figure 3.4: Алгоритм 4

Описание алгоритма 4 (см. рис. 3.4).

## 3.5 Алгоритм 5 (деление многоразрядных целых чисел).

**Алгоритм 5 (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 $u := u + vb^{i-t-1}$, $q_{i-t-1} := q_{i-t-1} - 1$.
4. $r := u$. Результат: $q$ и $r$.



Figure 3.5: Алгоритм 5

Описание алгоритма 5 (см. рис. 3.5).

# 4 Выполнение лабораторной работы

В каждом алгоритме первым делом проверяется корректность входных параметров.

## 4.1 Реализация алгоритма 1 (сложение неотрицательных целых чисел)

```
# 1. Сложение неотрицательных целых чисел
# b - система счисления
def algoritm1(u, v, b):
    n = len(u)
    print('* ', u, v, n, b)
    if len(u) != len(v):
        return print('Ошибка, числа разной разрядности')

    u, v, w = list(u), list(v), []
    k = 0

    for j in range(n-1,-1,-1):
        a = int(u[j]) + int(v[j]) + k
        w.append(a % b)
        k = a // b
        #print(j, a, w, k)
```

```
    w.append(k)
    res = ''
    for i in w[::-1]:
        res += str(i)
    return res
```

## 4.2 Реализация алгоритма 2 (вычитание неотрицательных целых чисел)

```
# 2. Вычитание неотрицательных целых чисел
def algoritm2(u, v, b):
    n = len(u)
    print('* ', u, v, n, b)
    if int(u) < int(v):
        return print('Ошибка, u < v')
    if len(u) != len(v):
        return print('Ошибка, числа разной разрядности')

    u, v, w = list(u), list(v), []
    k = 0

    for j in range(n-1,-1,-1):
        a = int(u[j]) - int(v[j]) + k
        w.append(a % b)
        k = a // b
        #print(j, a, w, k)

    w.append(k)
```

```
    res = ''
    for i in w[::-1]:
        res += str(i)
    return res
```

## 4.3 Реализация алгоритма 3 (умножение неотрицательных целых чисел столбиком)

```
# 3. Умножение неотрицательных целых чисел столбиом
def algoritm3(u, v, b):
    if u < v:
        u, v = v, u
    print('* ', u, v, b)

    n = len(u)
    m = len(v)
    u, v = list(u), list(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 = int(u[i]) * int(v[j]) + w[i+j+1] + k
                w[i+j+1] = t % b
                k = t // b
                #print(j,i, t, w, k)

            w[j] = k
```

```
    res = ''
    for i in w:
        res += str(i)
    return res
```

## 4.4 Реализация алгоритма 4 (быстрый столбик)

```
# 4. Быстрый столбик
def algoritm4(u, v, b):
    if u < v:
        u, v = v, u
    print('* ', u, v, b)

    t = 0
    n = len(u)
    m = len(v)
    u, v = list(u), list(v)
    w = [0] * (m + n)

    for s in range(0, m + n):
        for i in range(0, s + 1):
            if 0 <= n-i-1 < n and 0 <= m-s+i-1 < m:
                t += int(u[n-i-1]) * int(v[m-s+i-1])

        #print(s, i, t)
        w[m+n-s-1] = t % b
        t = t // b
```

```
    res = ''
    for i in w:
        res += str(i)
    return res
```

## 4.5 Промежуточные функции

Словари для перевода в системы счисления более 10

```
# словарь {символ: число}
str2num = {chr(i) : (i - ord('A') + 10) for i in range(ord('A'),ord('Z'))}
for i in '0123456789':
    str2num[i] = int(i)
# словарь {число: символ}
num2str = {value : key for (key, value) in str2num.items()}
```

Функци для перевода числа в десятичное и в b-ичное

```
# перевод b-ичного числа в десятичное
def to_10(u_str, b, array = False):
    # array = True, если число u передано в виде массива 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

# перевод десяичного числоа в b-ичное
def to_b(u, b):
    q, r = u // b, u % b    # частное q и остаток r
    w = num2str[r]
```

```
    while q >= b:
        q, r = q // b, q % b
        w += num2str[r]

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

    return w[::-1]
```

## 4.6 Реализация алгоритма 5 (деление многоразрядных целых чисел)

```
# 5. Деление многоразрядных целых чисел
# Возвращает: частное q, остаток r
def algoritm5(u_str, v_str, b):
    u = u_str; v = v_str;
    n = len(u) - 1; t = len(v) - 1 # разрядности чисел
    print('* ', u, v, n, t, b)
    u_10 = to_10(u, b); v_10 = to_10(v, b)
    if v[0] == 0 or not (n >= t >= 1):
        return "Некорректные входные данные"

    q = [0] * (n - t + 1)
    # шаг 2
    while u_10 >= v_10 * (b ** (n - t)):
        q[n - t] += 1
        u_10 -= v_10 * b ** (n - t)

    u = to_b(u_10, b)
```

```
u = [str2num[letter] for letter in u]
v = [str2num[letter] for letter in v_str]

# war 3
for i in range(n, t, -1):
    if u[n - i] >= v[0]:
        q[i-t-1] = b - 1
    else:
        q[i-t- 1] = (u[n-i] * b + u[n-i+1]) // v[0]

    while q[i-t-1] * (v[0]*b + v[1]) > u[n-i] * (b**2) + u[n-i+1]*b + u[n-i+2]:
        q[i-t-1] -= 1

    u_10 = to_10(u, b, True)
    u_10 -= v_10 * q[i-t-1] * (b ** (i-t-1))

    if u_10 < 0:
        u_10 += v_10 * (b ** (i-t-1))
        q[i-t-1] -= 1

    u = to_b(u_10, b); u = [str2num[letter] for letter in u]

res = ''
for i in q[::-1]:
    res += str(i)
res += ', '
for i in u:     # r
    res += str(i)
```

```
    return res
```

## 4.7 Проверки

Функция для проверки реализованных алгоритмов для выполнения арифметических операций с большими целыми числами на 5 вариантах входных значений:

```
def check(f):
    print('Результат: ',f('34','12', 10))
    print('Результат: ',f('1234','567', 10))
    print('Результат: ',f('67890','12345', 10))
    print('Результат: ',f('11','10', 2))
    print('Результат: ',f('1100101','1010101', 2))



print('1. Сложение неотрицательных целых чисел')
check(algoritm1)

print('\n2. Вычитание неотрицательных целых чисел')
check(algoritm2)

print('\n3. Умножение неотрицательных целых чисел столбиом')
check(algoritm3)

print('\n4. Быстрый столбик')
check(algoritm4)

print('\n5. Деление многоразрядных целых чисел')
check(algoritm5)
```

```
print('Результат: ',algoritm5('DEF','ABC', 16))
```

Вызов проверок работы всех реализованных функций на пяти разных вариантах входных параметров, задаваемых в функции check:

```
In [44]: runfile('C:/GitHub/sciproc-intro/
Lab_8/L8_Leonova.py', wdir='C:/GitHub/
sciproc-intro/Lab_8')
1. Сложение неотрицательных целых чисел
*  34 12 2 10
Результат:  046
*  1234 567 4 10
Ошибка, числа разной разрядности
Результат:  None
*  67890 12345 5 10
Результат:  080235
*  11 10 2 2
Результат:  101
*  1100101 1010101 7 2
Результат:  10111010

2. Вычитание неотрицательных целых чисел
*  34 12 2 10
Результат:  022
*  1234 567 4 10
Ошибка, числа разной разрядности
Результат:  None
*  67890 12345 5 10
Результат:  055545
*  11 10 2 2
Результат:  001
*  1100101 1010101 7 2
Результат:  00010000

3. Умножение неотрицательных целых чисел
столбиом
*  34 12 10
Результат:  0408
*  567 1234 10
Результат:  0699678
*  67890 12345 10
Результат:  0838102050
*  11 10 2
Результат:  0110
*  1100101 1010101 2
Результат:  10000110001001

3. Умножение неотрицательных целых чисел
столбиом
*  34 12 10
Результат:  0408
*  567 1234 10
Результат:  0699678
*  67890 12345 10
Результат:  0838102050
*  11 10 2
Результат:  0110
*  1100101 1010101 2
Результат:  10000110001001

4. Быстрый столбик
*  34 12 10
Результат:  0408
*  567 1234 10
Результат:  0699678
*  67890 12345 10
Результат:  0838102050
*  11 10 2
Результат:  0110
*  1100101 1010101 2
Результат:  10000110001001

5. Деление многоразрядных целых чисел
*  34 12 1 1 10
Результат:  2, 10
*  1234 567 3 2 10
Результат:  02, 100
*  67890 12345 4 4 10
Результат:  5, 6165
*  11 10 1 1 2
Результат:  1, 1
*  1100101 1010101 6 6 2
Результат:  1, 10000
*  DEF ABC 2 2 16
Результат:  1, 333

In [45]:
```

Figure 4.1: Результат выполнения L8_Leonova.py

Результат выполнения программы, проверка реализации 5-и алгоритмов для выполнения арифметических операций с большими целыми числами на 5 вариантах входных значений (см. рис. 4.1).

# 5 Выводы

Цель лабораторной работы была достигнута, 5 алгоритмов для выполнения арифметических операций с большими целыми числами были реализованы на языке программирования Python.

# Список литературы

1. Бубнов С.А. Лабораторный практикум по основам криптографии [Электронный ресурс]. Саратовский государственный университет имени Н.Г.Чернышевского, 2012. URL: http://elibrary.sgu.ru/uch_lit/656.pdf.