

# **Лабораторная работа №8**

**Дисциплина: Математические основы защиты информации и  
информационной безопасности**

Аветисян Давид Артурович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>13</b>

# List of Tables

# List of Figures

3.1	Начальные данные . . . . .	7
3.2	Сложение неотрицательных целых чисел . . . . .	8
3.3	Вычитание неотрицательных целых чисел . . . . .	9
3.4	Умножение неотрицательных целых чисел столбиком . . . . .	10
3.5	Алгоритм быстрого столбика . . . . .	11
3.6	Деление многоразрядных целых чисел . . . . .	12

# 1 Цель работы

Познакомиться с целочисленной арифметикой многократной точности.

## 2 Задание

1. Реализовать алгоритм сложения неотрицательных целых чисел.
2. Реализовать алгоритм вычитания неотрицательных целых чисел.
3. Реализовать алгоритм умножения неотрицательных целых чисел столбиком.
4. Реализовать алгоритм быстрого столбика.
5. реализовать алгоритм деления многоразрядных целых чисел.

### 3 Выполнение лабораторной работы

Для всех последующих алгоритмов были использованы числа  $u$  и  $v$ .

```
from math import floor

n = 3
m = 3
u = 175
v = 125
b = 10
```

Figure 3.1: Начальные данные

- 1) Для реализации алгоритма сложения неотрицательных целых чисел была написана следующая программа.

```

def sum_(n, u, v, b):
    k = 0
    w = []
    for j in range(n, 0, -1):
        u_j = u % b
        v_j = v % b
        w.append((u_j + v_j + k) % b)
        k = floor((u_j + v_j + k) / b)
        u = u // b
        v = v // b
    w0 = k
    if w0 == 1:
        w.append(w0)
    return w

w = sum_(n, u, v, b)
w.reverse()
print(*w, sep = '')

300

```

Figure 3.2: Сложение неотрицательных целых чисел

В данной программе:

- 1-3 строки. Задаём функцию и начальные данные
- 4-10 строки. Реализация алгоритма: отделяем от числа цифры, производим с ними вычисления при помощи формул из лабораторной и отсекаем цифру.
- 13 строка. Запись цифры ответа в список.

В данном случае я вычислил сумму 175 и 125. Вывод представлен на скриншоте



выше.

- 2) Для реализации алгоритма вычитания неотрицательных целых чисел была написана следующая программа.

```
def sub_(n, u, v, b):  
    k = 0  
    w = []  
    for j in range(n, 0, -1):  
        u_j = u % b  
        v_j = v % b  
        w.append((u_j - v_j + k) % b)  
        k = floor((u_j - v_j + k) / b)  
        u = u // b  
        v = v // b  
    return w  
  
w = sub_(n, u, v, b)  
w.reverse()  
print(*w, sep = ' ')
```

050

Figure 3.3: Вычитание неотрицательных целых чисел

Программа реализована аналогично предыдущей, но со знаком минуса. Вывод представлен на скриншоте выше.

- 3) Для реализации алгоритма умножения неотрицательных целых чисел столбиком была написана следующая программа.

```

def mult1(uu,vv,b):
    u = []
    v = []
    for i in str(uu):
        u.append(int(i))
    for i in str(vv):
        v.append(int(i))
    n = len(u)-1
    m = len(v)-1
    j = m
    w = [0] * (len(u) + len(v))

    while j >= 0:
        if v[j] == 0:
            w[j] = 0
            j = j-1
        else:
            i = n
            k = 0
            while i >= 0:
                t = u[i]*v[j]+w[i+j+1]+k
                w[i+j+1] = t%b
                k = t//b
                i = i-1
            w[j] = k
            j = j-1
    return print(''.join(str(i) for i in w))

```

```
mult1(u,v,b)
```

```
021875
```

Figure 3.4: Умножение неотрицательных целых чисел столбиком

В данной программе:

- 1-3 строки. Задаём функцию и подготавливаем переменные
- 4-29 строки. Реализация алгоритма: присваиваем нулевые значения, отделяем

цифры от числа и вычисляем новое значение по нескольким формулам, затем отсекаем цифру от числа и начинаем алгоритм заново.

В данном случае я вычислил разность 175 и 125. Вывод представлен на скриншоте выше.

- 4) Для реализации алгоритма быстрого столбика была написана следующая программа.

```
def mult2(uu,vv,b):
    u = [int(i) for i in str (uu)]
    v = [int(i) for i in str (vv)]
    n = len(u)-1
    m = len(v)-1
    w = [0] * (len(u) + len(v))

    t = 0
    for s in range(m+n+2):
        for i in range(s+1):
            if (n-i<0) or (m-s+i<0):
                t = t
            else:
                t = t+u[n-i]*v[m-s+i]
        w[m+n-s+1] = t%b
        t = t//b
    return print(''.join(str(i) for i in w))

mult2(u,v,b)

021875
```

Figure 3.5: Алгоритм быстрого столбика

Данная программа считает произведение более коротким образом. Вывод представлен на скриншоте выше. Он совпадает с предыдущим, но программа считает быстрее.

- 5) Для реализации алгоритма деления многоразрядных целых чисел была написана следующая программа.

```
def div_(uu,vv,b):
    u = uu
    v = vv
    n = len([int(i) for i in str (uu)])-1
    t = len([int(i) for i in str (vv)])-1
    q = [0] * (n-t+1)
    r = [0] * (t+1)

    while u >= v*b**(n-t):
        q[n-t] = q[n-t]+1
        u = u-v*b**(n-t)
    for i in range(n, t, -1):
        u_ = [int(i) for i in str(u)]
        u_.reverse()
        v_ = [int(i) for i in str(v)]
        v_.reverse()
        if u_[i] >= v_[t]:
            q[i-t-1] = b-1
        else:
            q[i-t-1] = (u_[i]*b+u_[i-1])//v_[t]
            while q[i-t-1]*(v_[t]*b+v_[t-1]) > u_[i]*b**2+u_[i-1]*b+u_[i-2]:
                q[i-t-1] = q[i-t-1]-1
            u = u-q[i-t-1]*b**(i-t-1)*v
            if u < 0:
                u = u+v*b**(i-t-1)
                q[i-t-1] = q[i-t-1]-1

    q.reverse()
    return print('Частное =', ''.join(str(i) for i in q), 'Остаток =', u)

div_(123456, 9, b)

Частное = 013605 Остаток = 1011
```

Figure 3.6: Деление многоразрядных целых чисел

Данный алгоритм аналогично путём отделения цифр от чисел считает их частное и записывает остаток. С каждой цифрой работаем отдельно и записываем, что мы взяли от других разрядов. В данном случае я вычислил частное и остаток при делении 123456 на 9. Вывод представлен на скриншоте выше.

## 4 Выводы

Я познакомился с целочисленной арифметикой многократной точности.