Отчёт по лабораторной работе №8  
Целочисленная арифметика многократной точности

Студент: Леонова Алина Дмитриевна, 1032212306

Группа: НФИмд-01-21

Преподаватель: Кулябов Дмитрий Сергеевич,

д-р.ф.-м.н., проф.

Москва 2021

Содержание

# 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 -ичной системе счисления, – натуральное число, ≥ 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)

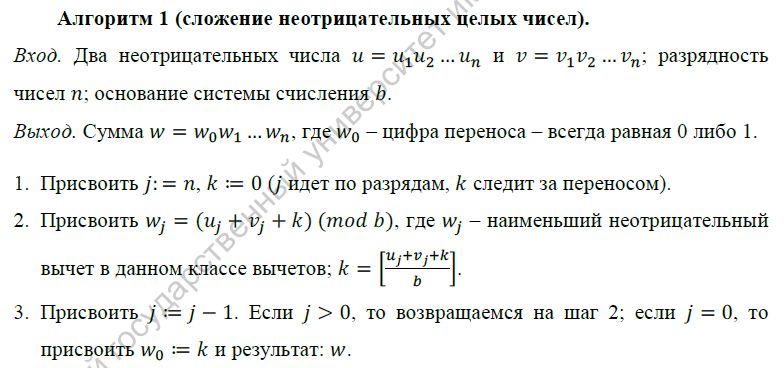


Figure 1: Алгоритм 1

Описание алгоритма 1 (см. рис. 1).

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

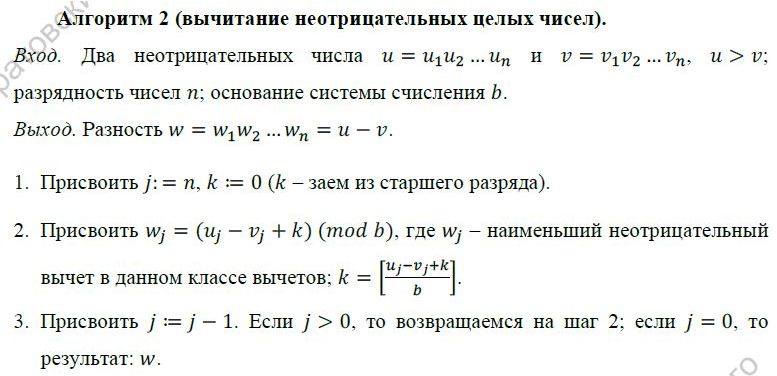


Figure 2: Алгоритм 2

Описание алгоритма 2 (см. рис. 2).

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

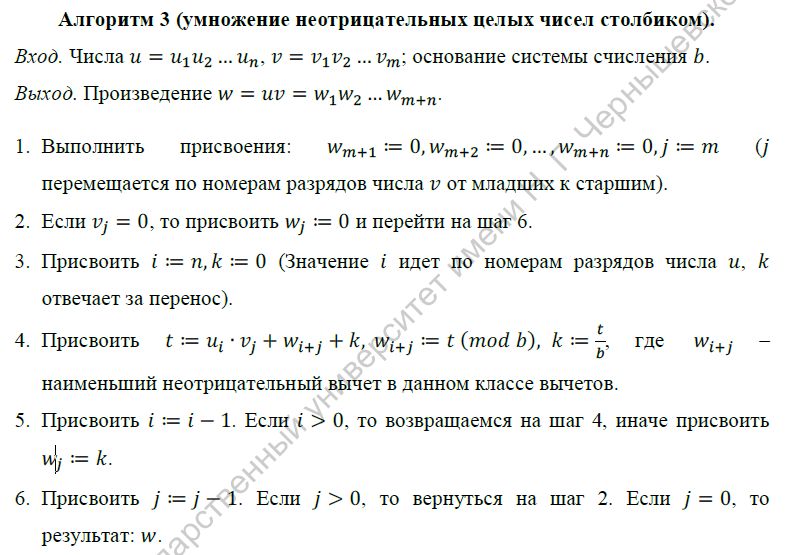


Figure 3: Алгоритм 3

Описание алгоритма 3 (см. рис. 3).

## 3.4 Алгоритм 4 (быстрый столбик).

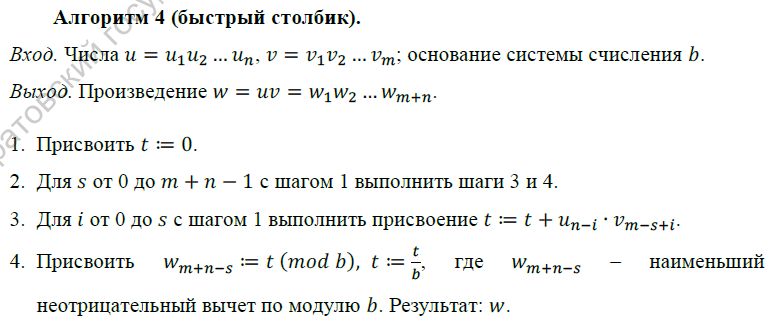


Figure 4: Алгоритм 4

Описание алгоритма 4 (см. рис. 4).

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

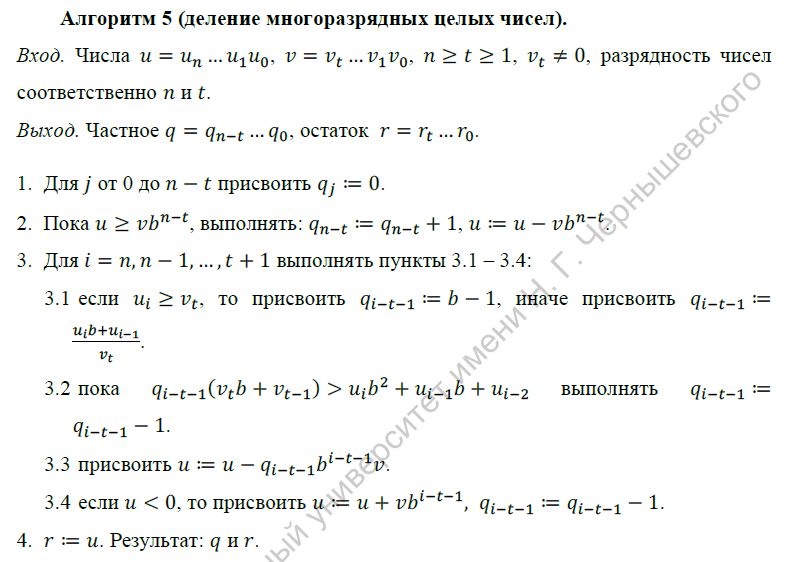


Figure 5: Алгоритм 5

Описание алгоритма 5 (см. рис. 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]  
  
 # шаг 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:

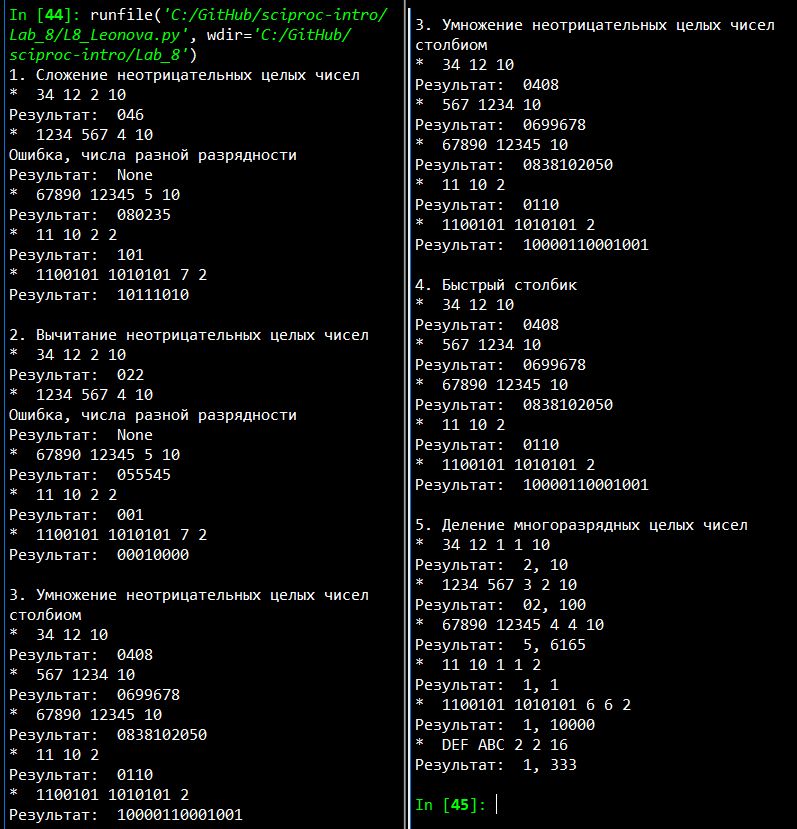


Figure 6: Результат выполнения L8\_Leonova.py

Результат выполнения программы, проверка реализации 5-и алгоритмов для выполнения арифметических операций с большими целыми числами на 5 вариантах входных значений (см. рис. 6).

# 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>.