

# Отчёт по лабораторной работе №8.

## Целочисленная арифметика многократной точности

---

*Дисциплина: Математические основы защиты информации  
и информационной безопасности*

**Студент:** Лапшенкова Любовь Олеговна 1032217633

**Группа:** НФИмд-02-21

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

30 декабря, 2021, Москва

# Цели и задачи работы

---

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

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

## **Ход выполнения и результаты**

---

# 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 1:** Вспомогательные действия для удобства дальнейших вычислений

# Алгоритм 1. Сложение неотрицательных целых чисел.

## Реализация

```
def algorythm_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
```

# Алгоритм 1. Сложение неотрицательных целых чисел.

## Результат

```
print(algorythm_1("321","1567",10))  
print(algorythm_1("B007","MI6",10))
```

```
01888  
13393
```

**Figure 3:** Алгоритм 1. Сложение неотрицательных целых чисел



# Алгоритм 2. Вычитание неотрицательных целых чисел.

## Реализация

```
def algrorhythm_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: Алгоритм 2. Вычитание неотрицательных целых чисел

## Алгоритм 2. Вычитание неотрицательных целых чисел.

### Результат

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

678

**Figure 5:** Алгоритм 2. Вычитание неотрицательных целых чисел

## 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#шар3  
            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#шар 5  
    return "".join([num2str[ciphra] for ciphra in w]) #массив в строку
```

**Figure 6:** Алгоритм 3. Умножение неотрицательных целых чисел столбиком

## Алгоритм 3. Умножение неотрицательных целых чисел столбиком. Результат

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

0958818

**Figure 7:** Алгоритм 3. Умножение неотрицательных целых чисел столбиком

## 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 8: Алгоритм 4. Быстрый столбик

## Алгоритм 4. Быстрый столбик. Результат

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

0958818

**Figure 9:** Алгоритм 4. Быстрый столбик

## 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 10: Алгоритм 5. Деление многоразрядных целых чисел

# Алгоритм 5. Деление многоразрядных целых чисел. Реализация

```
def to_b(number, b, n = 1):
    """
    Преобразит десятичное число в систему счисления с основанием b
    n - минимальная размерность 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:
        # продолжим деление
        (q, r) = (math.floor(q / b), q % b)
        w = w + num2str[r]

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

    # если размерность меньше желаемой..
    while len(w) < n:
        # добавим нули
        w = w + "0"

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

Figure 11: Алгоритм 5. Деление многоразрядных целых чисел



## 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 12:** Алгоритм 5. Деление многоразрядных целых чисел

# Алгоритм 5. Деление многоразрядных целых чисел. Реализация

```
def algorithm_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) # шаг 1  
  
    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 13: Алгоритм 5. Деление многоразрядных целых чисел

## Алгоритм 5. Деление многоразрядных целых чисел. Реализация

[illegible]

**Figure 14:** Алгоритм 5. Деление многоразрядных целых чисел

## Алгоритм 5. Деление многоразрядных целых чисел. Результат

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

**Figure 15:** Алгоритм 5. Деление многоразрядных целых чисел

В результате выполнения данной лабораторной работы нам удалось осуществить программно алгоритмы, рассмотренные в описании к лабораторной работе.

**Спасибо за внимание**