

Отчёт по лабораторной работе №8

Целочисленная арифметика многократной точности

*Дисциплина: Математические основы защиты информации
и информационной безопасности*

Студент: Агеева Анастасия Сергеевна, 1032212304

Группа: НФИмд-02-21

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

30 декабря, 2021, Москва

Прагматика

- В рамках дисциплины “Математические основы защиты информации и информационной безопасности” нам необходимо изучить ее разделы.
- Данная работа необходима для более глубоко и детального понимания работы алгоритмов шифрования.

Цель

Цель выполнения данной лабораторной работы

- Цель данной лабораторной работы изучение алгоритмов целочисленной арифметики многократной точности.

Задачи

Задачи выполнения данной лабораторной работы

1. Реализовать программно алгоритм сложения неотрицательных чисел;
2. Реализовать программно алгоритм вычитания неотрицательных чисел;
3. Реализовать программно алгоритм умножения неотрицательных целых чисел столбиком;
4. Реализовать программно алгоритм умножения “быстрый столбик”;
5. Реализовать программно алгоритм деления многоразрядных целых чисел.

Результаты выполнения данной лабораторной работы

Вспомогательные функции (1)

```
In [1]: import math
```

Figure 1: Импорт библиотеки math

```
In [2]: # словарь
# строковый символ : числовой аналог
str2num = {chr(letter_sym) : (letter_sym - ord("A") + 10) for letter_sym in range(ord("A"), ord("Z") + 1)}
for i in "0123456789":
    str2num[i] = int(i)

# число : строковый аналог
num2str = {value : key for (key, value) in str2num.items()}
```

Figure 2: Описание словарей

```
In [3]: # перевод в 10-ную систему счисления
# array = True, если число передается в виде массива чисел

def to_10 (u_str, b, array = False):
    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 3: Перевод в десятичную систему счисления

Вспомогательные функции (2)

```
In [4]: # перевод в систему счисления с основанием b
# n - минимальная разрядность возвращаемого числа
def to_b(number, b, n = 1):
    (q, r) = (math.floor(number / b), number % b)
    w = num2str[r]
    while q >= b:
        (q, r) = (math.floor(q / b), q % b)
        w = w + num2str[r]
    if q != 0:
        w = w + num2str[q]
    while len(w) < n:
        w = w + "0"
    return w[::-1]
```

Figure 4: Перевод в систему счисления с основанием b

```
In [5]: # удаление 0 в начале числа
def trim_zero(a):
    while a[0] == '0' and len(a) > 1:
        a = a[1:]
    return a
```

```
In [6]: # добавления 0 в начало числа
def fill_zero(u, n, array = False):
    result = [0]*(n - len(u))
    if array:
        result.extend(u)
        return result
    return "".join([str(i) for i in result]) + u
```

Figure 5: Удаление/добавление нулей

Сложение неотрицательных чисел

```
In [7]: def addition(u_str, v_str, b):  
        u = [str2num[letter] for letter in u_str]  
        v = [str2num[letter] for letter in v_str]  
        if len(u) != len(v):  
            if len(u) < len(v):  
                u = fill_zero(u, len(v), True)  
            else:  
                v = fill_zero(v, len(u), True)  
        n = len(u)  
        k = 0  
        w = []  
        for j in range(n-1, -1, -1):  
            w.append(((u[j] + v[j] + k)%b))  
            k = math.floor((u[j] + v[j] + k)/b)  
        w.append(k)  
        w.reverse()  
        return "".join([num2str[i] for i in w])
```

```
In [8]: addition("12345", "6789", 10)
```

```
Out[8]: '019134'
```

```
In [9]: addition("010101", "001001", 2)
```

```
Out[9]: '0011110'
```

Figure 6: Сложение

Вычитание неотрицательных чисел

```
In [10]: def subtraction(u_str, v_str, b):  
    u = [str2num[letter] for letter in u_str]  
    v = [str2num[letter] for letter in v_str]  
    if len(u) != len(v):  
        if len(u) < len(v):  
            u = fill_zero(u, len(v), True)  
        else:  
            v = fill_zero(v, len(u), True)  
    if u < v:  
        return "u must be more than v"  
    n = len(u)  
    k = 0  
    w = []  
    for j in range(n-1, -1, -1):  
        w.append(((u[j] - v[j] + k)%b))  
        k = math.floor((u[j] - v[j] + k)/b)  
    w.reverse()  
    return "".join([num2str[i] for i in w])
```

```
In [11]: subtraction("12345", "6789", 10)
```

```
Out[11]: '05556'
```

```
In [12]: subtraction("010101", "001001", 2)
```

```
Out[12]: '001100'
```

```
In [13]: subtraction("345", "6789", 10)
```

```
Out[13]: 'u must be more than v'
```

Figure 7: Вычитание

Умножение неотрицательных целых чисел столбиком

```
In [14]: def column_multiply(u_str, v_str, b):  
    u = [str2num[letter] for letter in u_str]  
    v = [str2num[letter] for letter in v_str]  
    n = len(u)  
    m = len(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 = u[i]*v[j] + w[i+j+1] + k  
                w[j+i+1] = t%b  
                k = math.floor(t/b)  
            w[j] = k  
    return "".join([num2str[i] for i in w])
```

```
In [15]: column_multiply("12345", "6789", 10)
```

```
Out[15]: '083810205'
```

```
In [16]: column_multiply("010101", "001001", 2)
```

```
Out[16]: '000010111101'
```

Figure 8: Умножение столбиком

Умножение неотрицательных чисел быстрым столбиком

```
In [17]: def quick_multiply(u_str, v_str, b):
          u = [str2num[letter] for letter in u_str]
          v = [str2num[letter] for letter in v_str]
          n = len(u)
          m = len(v)
          w = [0]*(m + n)
          t = 0
          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 = t + u[n - i - 1] * v[m - s + i - 1]
              w[m + n - s - 1] = t % b
              t = math.floor(t / b)
          return "".join([num2str[i] for i in w])
```

```
In [18]: quick_multiply("12345", "6789", 10)
```

```
Out[18]: '083810205'
```

```
In [19]: quick_multiply("010101", "001001", 2)
```

```
Out[19]: '000010111101'
```

Figure 9: Быстрый столбик

Деление многоразрядных целых

```
In [20]: def division(u_str, v_str, b):
        u = u_str
        v = v_str
        u_10 = to_10(u, b)
        v_10 = to_10(v, b)
        n = len(u) - 1
        t = len(v) - 1
        if v[0] == 0 or not (n >= t >= 1):
            return "Incorrect data"
        q = [0]*(n - t + 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)
        u = [str2num[letter] for letter in u]
        v = [str2num[letter] for letter in v_str]
        for i in range(n, t, -1):
            if u[n - i] >= v[0]:
                q[i-t+1] = b - 1
            else:
                q[i-t+1] = math.floor((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] = 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, n+1)
                u = [str2num[letter] for letter in u]
        q = "".join([num2str[i] for i in q])
        r = "".join([num2str[i] for i in u])
        return trim_zero(q[:-1]), trim_zero(r)
```

```
In [21]: division("225", "15", 10)
```

```
Out[21]: ('15', '0')
```

```
In [22]: division("225", "0", 10)
```

```
Out[22]: 'Incorrect data'
```

```
In [23]: division("010101", "010", 2)
```

```
Out[23]: ('1010', '0')
```

Figure 10: Деление

- Исходя из теоретических сведений, программа выполнена без ошибок, чему свидетельствуют полученные результаты.
- В ходе данной лабораторной работы я реализовала программно 5 алгоритмов целочисленной арифметики многократной точности..