Лабораторная работа №8

Математические основы защиты информации и информационной безопасности

Леонтьева Ксения Андреевна | НПМмд-02-23

Содержание

Список иллюстраций

1 Цель работы

Реализовать на языке программирования алгоритмы для выполнения арифметических операций с большими целыми числами.

2 Теоретическое введение

Считаем, что число записано в b-ичной системе счисления, b - натуральное число, $b \geq 2$. Натуральное n-разрядное число будем записывать в виде

$$u = u_1 u_2 ... u_n$$

. 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.

Более подробно см. в [1].

3 Выполнение лабораторной работы

Алгоритм 1 (сложение неотрицательных целых чисел).

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

Выход. Сумма $w = w_0w_1...w_n$, где w_0 - цифра переноса - всегда равная 0 или 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; $k = [\frac{u_j+v_j+k}{b}]$.
3. Присвоить $j = j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то присвоить $w_0 = k$ и результат w.

Код программы (рис. 3.1).

```
def remove_zeros(w):
        z = 0
        while w[z] == 0:
            z = z + 1
        return(w[z:])
```

```
#Алгоритм 1
def a1(uu,vv,b):

    u = [int(i) for i in str(uu)]
    v = [int(i) for i in str(vv)]
    l = len(u)
    w = []
    n = len(u) - 1

    j = n
    k = 0
    while j != -1:
        w.append((u[j] + v[j] + k) % b)
        k = (u[j] + v[j] + k) // b
        j = j - 1
    if k != 0:
        w.append(k)
    w.reverse()
    return print(''.join(str(i) for i in remove_zeros(w)))
a1(23, 11, 10)
```

```
34
```

Рис. 3.1: Алгоритм 1 (сложение неотрицательных целых чисел)

Алгоритм 2 (вычитание неотрицательных целых чисел).

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

Выход. Сумма $w = w_1w_2...w_n = u - v$.

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; $k = [\frac{u_j - v_j + k}{b}]$.
3. Присвоить $j = j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то результат w.

Код программы (рис. 3.2).

```
#Алгоритм 2
def a2(uu,vv,b):
    u = [int(i) for i in str(uu)]
    v = [int(i) for i in str(vv)]
    l = len(u)
    w = []
    n = len(u) - 1

    j = n
    k = 0
    while j != -1:
        w.append((u[j] - v[j] + k) % b)
        k = (u[j] - v[j] + k) // b
        j = j - 1
    w.reverse()
    return print(''.join(str(i) for i in remove_zeros(w)))

a2(2035, 2000, 10)
```

```
35
```

Рис. 3.2: Алгоритм 2 (вычитание неотрицательных целых чисел)

Алгоритм 3 (умножение неотрицательных целых чисел столбиком).
Вход. Числа $u = u_1u_2...u_n$ и $v = v_1v_2...v_m$; основание системы счисления b.
Выход. Произведение $w = uv = w_1w_2...w_{m+n}$.

1. Выполнить присвоения: $w_{m+1} = 0, w_{m+2} = 0, ..., w_{m+n} = 0, j = m$ (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 $i = u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} = t(mod\ b)$, $k = \frac{t}{b}$, где w_{i+j} - наименьший неотрицательный вычет в данном классе вычетов.

4. Присвоить $i = i-1$. Если $i > 0$, то возвращаемся на шаг 4, иначе присвоить $w_j = k$.

5. Присвоить $j = j - 1$. Если $j > 0$, то вернуться на шаг 2. Если $j = 0$, то результат w.

Код программы (рис. 3.3).

```
#Алгоритм 3
def a3(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
    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
        z = 0
        while w[z] == 0:
            z = z + 1
    return print(''.join(str(i) for i in remove_zeros(w)))
a3(5497, 296, 10)
```

```
1627112
```

Рис. 3.3: Алгоритм 3 (умножение неотрицательных целых чисел столбиком)

Алгоритм 4 (быстрый столбик).

Вход. Числа $u = u_1u_2...u_n$ и $v = v_1v_2...v_m$; основание системы счисления b.

Выход. Произведение $w = uv = w_1w_2...w_{m+n}$.

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.

Код программы (рис. 3.4).

```
#Алгоритм 4
def a4(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 remove_zeros(w)))

a4(5497, 296, 10)
```

```
1627112
```

Рис. 3.4: Алгоритм 4 (быстрый столбик)

Алгоритм 5 (деление многоразрядных целых чисел).

Вход. Числа $u = u_n...u_1u_0$ и $v = v_t...v_1v_0, n \geq t \geq 1, v_t \neq 0$, разрядность чисел соответственно n и t.

Выход. Частное $q = q_{n-t}...q_0$, остаток $r = r_t...r_0$.

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 $i = n, n - 1, ..., t + 1$ выполнять пункты 3.1 - 3.4:

   3.1. если $u_i \geq v_t$, то присвоить $q_{i-t-1} = b - 1$, иначе присвоить $q_{i-t-1} = \frac{u_ib+u_{i-1}}{v_t}$.

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 $u = u + v b^{i-t-1}$, $q_{i-t-1} = q_{i-t-1} - 1$.

4. $r = u$. Результат q и r.

Код программы (рис. 3.5 - 3.6).

```
#Алгоритм 5
def a5(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)

    n = len([int(i) for i in str(u)]) - 1
    t = len([int(i) for i in str(v)]) - 1

    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]
```

Рис. 3.5: Алгоритм 5 (деление многоразрядных целых чисел)

```
        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 remove_zeros(q)), 'Остаток =', u)

a5(389725851, 79116, 10)
```

```
Частное = 4926 Остаток = 435
```

Рис. 3.6: Алгоритм 5 (деление многоразрядных целых чисел)

4 Выводы

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для выполнения арифметических операций с большими целыми числами.

Список литературы

1. Целочисленная арифметика многократной точности [Электронный ресурс]. URL: https://studfile.net/preview/2439346/page:35/.