Лабораторная работа №8

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить работу алгоритмов целочисленной арифметики многократной точности: сложение неотрицательных целых чисел; вычитание неотрицательных целых чисел; умножение неотрицательных целых чисел столбиком; быстрый столбик; деление многоразрядных целых чисел; а также реализовать их программно.

2 Теоретическое введение

2.1 Целочисленная арифметика многократной точности

В данной работе рассмотрим алгоритмы для выполнения арифметических операций с большими целыми числами. Будем считать, что число записано в b–ичной системе счисления, b – натуральное число, $b \geq 2$. Натуральное n–разрядное число будем записывать в виде:

$$u = u_1 u_2 \dots u_n.$$

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.

2.2 Алгоритм 1 (сложение неотрицательных целых чисел)

Вход. Два неотрицательных числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$, разрядность чисел n; основание системы счисления b.

Выход. Сумма $w = w_0 w_1 \dots w_n$, где w_0 – цифра переноса – всегда равная 0 либо 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; $k = \left\lfloor \frac{u_j + v_j + k}{b} \right\rfloor$.
3. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2; если $j = 0$, то присвоить $w_0 := k$.

Результат. w.

2.3 Алгоритм 2 (вычитание неотрицательных целых чисел)

Вход. Два неотрицательных числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_n$, $u > v$; разрядность чисел n; основание системы счисления b.

Выход. Разность $w = w_1 w_2 \dots w_n = u - v$.

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; $k = \left\lfloor \frac{u_j - v_j + k}{b} \right\rfloor$.
3. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2, иначе присвоить $w_0 := k$.

Результат. w.

2.4 Алгоритм 3 (умножение неотрицательных целых чисел столбиком)

Вход. Числа $u = u_1 u_2 \dots u_n$ и $v = v_1 v_2 \dots v_m$; основание системы счисления b.

Выход. Произведение $w = u \cdot v = w_1 w_2 \dots w_{m+n}$.

1. Присвоить $w_{m+1} := 0, w_{m+2} := 0, \dots, w_{m+n} := 0, j := m$ (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 $i := n$, $k := 0$ (Значение i идет по номерам разрядов числа u, k отвечает за перенос).
4. Присвоить $t := u_i v_j + w_{i+j} + k$, $w_{i+j} := t \mod b$, $k := \frac{t}{b}$, где w_{i+j} наименьший неотрицательный вычет в данном классе вычетов.
5. Присвоить $w_j = (u_j \cdot v_j + k) \mod b$, где w_j – наименьший неотрицательный вычет.
6. Присвоить $j := j - 1$. Если $j > 0$, то возвращаемся на шаг 2, иначе алгоритм завершён.

Результат. w.

2.5 Алгоритм 4 (быстрый столбик)

Вход. Числа $u = u_1 u_2 \dots u_n$, $v = v_1 v_2 \dots v_m$; основание системы счисления b.
Выход. Произведение $w = u \cdot v = w_1 w_2 \dots w_{m+n}$.
Запишем алгоритм для нумерации массивов, начинающейся с единицы:

1. Присвоить $t := 0$.
2. Для s от 1 до $m + n$ с шагом 1 выполнить шаги 3 и 4.
3. Для i от $\max(1, s - m + 1)$ до $\min(n, s)$ с шагом 1 выполнить присвоение $t := t + u_{n-i+1} * v_{m-s+i}$.
4. Присвоить $w_{m+n-s+1} := t \mod b, t := \lfloor \frac{t}{b} \rfloor$, где $w_{m+n-s+1}$ - наименьший неотрицательный вычет по модулю b.

Результат. w.

2.6 Алгоритм 5 (деление многоразрядных целых чисел)

Вход. Числа $u = u_0 u_1 \dots u_n$ и $v = v_0 v_1 \dots v_t$, $n \geq t \geq 1$, $v_t \neq 0$; основание системы счисления b.

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

1. Преобразовать u и v в массивы цифр в системе счисления с основанием b:
   $u_{\text{digits}} = [u_0, u_1, \dots, u_n]$,
   $v_{\text{digits}} = [v_0, v_1, \dots, v_t]$.
2. Инициализировать:
   - $q_j := 0$ для $j = 0, 1, \dots, n - t$;
   - $r := 0$.
3. Для $i = 0, 1, \dots, n - 1$ выполнять:
   1. Обновить остаток:
      $r := r \cdot b + u_{\text{digits}}[i]$.
   2. Если $r \geq v$:
      - Если $i - t + 1 > 0$, то $q[i - t + 1] := \lfloor r/v \rfloor$;
      - Иначе $q[i - t + 1] := 0$.
      - Обновить остаток: $r := r \mod v$.
   3. Иначе, если $i - t + 1 > 0$, присвоить $q[i - t + 1] := 0$.
4. Удалить ведущие нули из массива q.
5. Вернуть q и r как результат.

Результат. Частное q и остаток r.

3 Выполнение лабораторной работы

Действуя согласно [1], реализуем все описанные алгоритмы на языке Julia.

Сначала реализуем алгоритмы сложения и вычитания многоразрядных неотрицательных целых чисел (Рис.[3.1]), после чего реализуем алгоритмы умножения многоразрядных целых чисел столбиком (Рис.[3.2]), а также алгоритм быстрого столбика (Рис.[3.3]). Наконец, реализуем алгоритм деления многоразрядных целых чисел (Рис.[3.4]). Тогда для двух пар чисел: $u_1 = 6957, v_1 = 1423$ (Рис.[3.5]), $u_2 = 6957142, v_2 = 1423$ (Рис.[3.6]), получим следующий вывод, представленный на Рис.[3.7].

```julia
"""Алгоритм 1. Сложение неотрицательных целых чисел"""
function Add_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    # u, v - складываемые числа, записаны в виде массивов цифр, b - основание системы счисления
    n = length(u)
    m = length(v)
    w = zeros(Int, n)
    k = 0
    j = n
    while j > 0
        temp = u[j] + v[j] + k
        w[j] = mod(temp, b)
        k = div(temp, b)
        j -= 1
    end
    w[1] += k
    return w
end

"""Алгоритм 2. Вычитание неотрицательных целых чисел"""
function Subtract_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    # u, v - вычитаемые числа, записаны в виде массивов цифр, b - основание системы счисления
    n = length(u)
    w = zeros(Int, n)
    k = 0
    j = n
    while j > 0
        temp = u[j] - v[j] + k
        w[j] = mod(temp, b)
        k = div(temp, b)
        j -= 1
    end
    w[1] += k
    return w
end
```

Рис. 3.1: Код алгоритмов сложения и вычитания неотрицательных целых чисел на Julia

```julia
"""Алгоритм 3. Умножение неотрицательных целых чисел столбиком"""
function Multiply_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)
    j = m
    while j > 0
        if v[j] == 0
            w[j] = 0
            j -= 1
        else
            i = n
            k = 0
            while i > 0
                temp = w[i+j] + u[i] * v[j] + k
                w[i+j] = mod(temp, b)
                k = div(temp, b)
                i -= 1
            end
            w[j] = k
            j -= 1
        end
    end
    return w
end
```

Рис. 3.2: Код алгоритма умножения неотрицательных целых чисел столбиком на Julia

```julia
function Fast_Multiply_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)  # Результат будет длиной n + m
    t = 0  # Переменная для переноса

    for s in 1:(n + m)  # Итерации по разрядам результата
        for i in max(1, s - m + 1):min(n, s)
            t += u[n - i + 1] * v[m - (s - i)]
        end
        w[n + m - s + 1] = mod(t, b)  # Записываем младший разряд
        t = div(t, b)  # Переносим в старший разряд
    end

    w[1] += t  # Добавляем оставшийся перенос, если он есть
    return w
end
```

Рис. 3.3: Код умножения неотрицательных целых чисел алгоритмом быстрого столбика на Julia

```julia
function Divide_Large_Positive_Numbers(u::Int, v::Int, b::Int)
    # Преобразуем числа в массивы цифр
    u_digits = reverse(digits(u, base=b))
    v_digits = reverse(digits(v, base=b))

    # Инициализация переменных
    n = length(u_digits)
    t = length(v_digits)
    q = zeros(Int, max(0, n - t + 1)) # Вычисление размера массива q
    r = 0

    # Основной цикл деления
    for i in 1:n
        # Обновляем остаток, добавляя очередную цифру u_digits
        r = r * b + u_digits[i]

        # Определяем текущую цифру частного
        if r >= v
            if i - t + 1 > 0
                q[i - t + 1] = div(r, v)
            else
                q[i - t + 1] = 0 # Избегаем отрицательных индексов
            end
            r %= v
        else
            if i - t + 1 > 0
                q[i - t + 1] = 0
            end
        end
    end
```

Рис. 3.4: Код алгоритма деления многоразрядных неотрицательных целых чисел на Julia

```julia
# Пример использования 1
b = 10  # Основание системы счисления

# Пример чисел для тестирования
u = [6, 9, 5, 7]  # Число u = 6957
v = [1, 4, 2, 3]  # Число v = 1423
u1, v1 = 6957, 1423

println("Число u: ", u, "\t Число v:", v)

# Алгоритм 1: Сложение
sum_result = Add_Large_Positive_Numbers(u, v, b)
println("Сумма: ", sum_result)

# Алгоритм 2: Вычитание
subtract_result = Subtract_Large_Positive_Numbers(u, v, b)
println("Разность: ", subtract_result)

# Алгоритм 3: Умножение
multiply_result = Multiply_Large_Positive_Numbers(u, v, b)
println("Произведение: ", multiply_result)

# Алгоритм 4: Быстрое умножение
fast_multiply_result = Fast_Multiply_Large_Positive_Numbers(u, v, b)
println("Быстрый столбик: ", fast_multiply_result)

# Алгоритм 5: Деление
quotient, remainder = Divide_Large_Positive_Numbers(u1, v1, b)
println("Частное: ", quotient)
println("Остаток: ", remainder)
```

Рис. 3.5: Код начальных данных для проверки работы алгоритмов на Julia (1/2)

```julia
# Пример использования 2
b = 10  # Основание системы счисления

# Пример чисел для тестирования
u = [6, 9, 5, 7, 1, 4, 2]  # Число u = 6957142
v = [1, 4, 2, 3]  # Число v = 1423

println("\nЧисло u: ", u, "\t Число v:", v)

# Алгоритм 3: Умножение
multiply_result = Multiply_Large_Positive_Numbers(u, v, b)
println("Произведение: ", multiply_result)

# Алгоритм 4: Быстрое умножение
fast_multiply_result = Fast_Multiply_Large_Positive_Numbers(u, v, b)
println("Быстрый столбик: ", fast_multiply_result)

# Алгоритм 5: Деление
u2, v2 = 6957142, 1423
#quotient, remainder = Divide_Large_Positive_Numbers(u, v, b)
quotient, remainder = Divide_Large_Positive_Numbers(u2, v2, b)
println("Частное: ", quotient)
println("Остаток: ", remainder)
```

Рис. 3.6: Код начальных данных для проверки работы алгоритмов на Julia (2/2)

```
thbase-infosec\labs\lab08\report\report> julia .\lab8.jl
Число u: [6, 9, 5, 7]     Число v:[1, 4, 2, 3]
Сумма: [8, 3, 8, 0]
Разность: [5, 5, 3, 4]
Произведение: [0, 9, 8, 9, 9, 8, 1, 1]
Быстрый столбик: [0, 9, 8, 9, 9, 8, 1, 1]
Частное: [4]
Остаток: 1265

Число u: [6, 9, 5, 7, 1, 4, 2]    Число v:[1, 4, 2, 3]
Произведение: [0, 9, 9, 0, 0, 0, 1, 3, 0, 6, 6]
Быстрый столбик: [0, 9, 9, 0, 0, 0, 1, 3, 0, 6, 6]
Частное: [4, 8, 8, 9]
Остаток: 95
```

Рис. 3.7: Результат выполнения кода алгоритмов арифметических операций с многоразрядными целыми числами на Julia

4 Выводы

В ходе выполнения лабораторной работы я изучил работу алгоритмов целочисленной арифметики многократной точности: сложение неотрицательных целых чисел; вычитание неотрицательных целых чисел; умножение неотрицательных целых чисел столбиком; быстрый столбик; деление многоразрядных целых чисел; а также реализовал их программно.

Список литературы

1. Лабораторная работа № 8. Целочисленная арифметика многократной точности [Электронный ресурс]. Саратовский государственный университет имени Н.Г. Чернышевского, 2024.