

Отчет по лабораторной работе №8

***дисциплина: Математические основы защиты информации и
информационной безопасности***

Морозова Ульяна

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
2.1	Алгоритм 1 (сложение неотрицательных целых чисел)	4
2.2	Алгоритм 2 (вычитание)	5
2.3	Алгоритм 3 (умножение столбиком)	6
2.4	Алгоритм 4 (умножение быстрое)	7
2.5	Алгоритм 5 (деление)	8
2.6	Работа алгоритмов	11
3	Выводы	15

1 Цель работы

Целью работы является изучение алгоритмов для выполнения арифметических операций с большими целыми числами и реализация их на языке Julia.

2 Выполнение лабораторной работы

2.1 Алгоритм 1 (сложение неотрицательных целых чисел)

```
using Printf
```

```
function remove_leading_zeros(arr::Vector{Int})  
    first_nonzero = findfirst(x -> x != 0, arr)  
    if isnothing(first_nonzero)  
        return [0]  
    end  
    return arr[first_nonzero:end]  
end
```

```
function pad_arrays(u::Vector{Int}, v::Vector{Int})  
    n = length(u)  
    m = length(v)  
    max_len = max(n, m)  
    new_u = vcat(zeros{Int}(max_len - n), u)  
    new_v = vcat(zeros{Int}(max_len - m), v)  
    return new_u, new_v, max_len  
end
```

```

function add_long(u_in::Vector{Int}, v_in::Vector{Int}, b::Int)
    u, v, n = pad_arrays(u_in, v_in)

    w = zeros(Int, n + 1)

    k = 0

    for j in n:-1:1
        # Здесь k используется (чтение), поэтому оно должно быть определено выше
        sum_val = u[j] + v[j] + k

        w[j+1] = mod(sum_val, b)

        # Здесь k обновляется
        k = div(sum_val, b)
    end

    w[1] = k
    return remove_leading_zeros(w)
end

```

2.2 Алгоритм 2 (вычитание)

```

function sub_long(u_in::Vector{Int}, v_in::Vector{Int}, b::Int)
    u, v, n = pad_arrays(u_in, v_in)
    w = zeros(Int, n)

    k = 0 # <--- ДОБАВЛЕНО: Инициализация заема

```

```

for j in n:-1:1
    diff = u[j] - v[j] + k

    if diff < 0
        w[j] = diff + b
        k = -1
    else
        w[j] = diff
        k = 0
    end
end

return remove_leading_zeros(w)
end

```

2.3 Алгоритм 3 (умножение столбиком)

```

function mul_long_classic(u::Vector{Int}, v::Vector{Int}, b::Int)
    n = length(u)
    m = length(v)
    w = zeros(Int, m + n)

    for j in m:-1:1
        if v[j] == 0
            continue
        end

        k = 0
    end

```

```

    for i in n:-1:1

        t = u[i] * v[j] + w[i+j] + k

        w[i+j] = mod(t, b)
        k = div(t, b)

    end

    w[j] = k

end

return remove_leading_zeros(w)
end

```

2.4 Алгоритм 4 (умножение быстрое)

```

function mul_long_fast(u::Vector{Int}, v::Vector{Int}, b::Int)
    n = length(u)
    m = length(v)
    w = zeros{Int, m + n}
    t = 0 # [cite: 64]

    for s in 0:(m + n - 1)

        for i in 0:s
            idx_u = n - i
            idx_v = m - s + i

```

```

        if idx_u >= 1 && idx_u <= n && idx_v >= 1 && idx_v <= m
            t = t + u[idx_u] * v[idx_v]
        end
    end

    if (m + n - s) >= 1
        w[m + n - s] = mod(t, b)
        t = div(t, b)
    end
end

return remove_leading_zeros(w)
end

```

2.5 Алгоритм 5 (деление)

```

function div_long(u_in::Vector{Int}, v_in::Vector{Int}, b::Int)
    # Копируем и обязательно убираем лидирующие нули у делителя для корректной длины
    v = remove_leading_zeros(v_in)

    # ПРОВЕРКА НА 0
    if v == [0]
        error("Division by zero")
    end

    # Если делитель больше делимого, возвращаем 0 и остаток
    if length(v) > length(u_in)
        return [0], u_in
    end

```

```

# 1. ДОБАВЛЯЕМ ЛИДИРУЮЩИЙ НОЛЬ К ДЕЛИМОМУ
# Это критически важно для корректной работы алгоритма Кнута
u = [0; u_in]

n = length(u)
m = length(v)

# Длина частного: (L_u_original - L_v + 1)
q_len = length(u_in) - length(v_in) + 1
q = zeros(Int, q_len)

# Основной цикл деления
for k in 1:q_len
    # Эвристическая оценка q_hat
    # Используем u[k], u[k+1] и v[1]
    u_high = u[k]
    u_next = u[k+1]
    v_high = v[1]

    q_hat = 0

    if u_high == v_high
        q_hat = b - 1
    else
        # Оценка сверху: (u[k]*b + u[k+1]) / v[1]
        q_hat = div(u_high * b + u_next, v_high)
    end
end

```

Коррекция q_hat (проверка с третьей цифрой)

`v_next = (m > 1) ? v[2] : 0`

`u_third = (k + 2 <= n) ? u[k+2] : 0`

while true

`left = q_hat * (v_high * b + v_next)`

`right = u_high * b^2 + u_next * b + u_third`

if left > right

`q_hat -= 1`

else

break

end

end

*# Вычитание q_hat * v из u, начиная с позиции k*

Нам нужно вычесть v из u[k ... k+m]

if q_hat > 0

`borrow = 0`

for i **in** m:-1:1

`idx_u = k + i`

*# Вычитаем: u - (q * v + borrow)*

`val = u[idx_u] - (q_hat * v[i]) + borrow`

Обработка заема в текущей системе счисления

`curr_digit = mod(val, b)`

`borrow = div(val - curr_digit, b)` *# borrow будет отрицательным или 0*

`u[idx_u] = curr_digit`

```

    end

    # Финальный перенос к старшему разряду (u[k])
    u[k] += borrow
end

# Если "перестарались" (u[k] < 0), делаем коррекцию (add back)
if u[k] < 0
    q_hat -= 1
    carry = 0
    for i in m:-1:1
        idx_u = k + i
        val = u[idx_u] + v[i] + carry
        u[idx_u] = mod(val, b)
        carry = div(val, b)
    end
    u[k] += carry
end

q[k] = q_hat
end

return remove_leading_zeros(q), remove_leading_zeros(u)
end

```

2.6 Работа алгоритмов

```

function number_to_digits(n::Int, b::Int)
    if n == 0 return [0] end
    digits = Int[]

```

```

while n > 0
    push!(digits, mod(n, b))
    n = div(n, b)
end
return reverse(digits)
end

function digits_to_number(arr::Vector{Int}, b::Int)
    res = 0
    for d in arr
        res = res * b + d
    end
    return res
end

function run_tests()
    BASE = 10
    println("Система счисления: ", BASE)
    println("-----")

    a_val, b_val = 12345, 6789
    u = number_to_digits(a_val, BASE)
    v = number_to_digits(b_val, BASE)
    res_add = add_long(u, v, BASE)
    println("Сложение: $a_val + $b_val")
    println(" Ожидается: ", a_val + b_val)
    println(" Получено: ", digits_to_number(res_add, BASE))

    a_val, b_val = 12345, 6789

```

```

u = number_to_digits(a_val, BASE)
v = number_to_digits(b_val, BASE)
res_sub = sub_long(u, v, BASE)
println("\nВычитание: $a_val - $b_val")
println("  Ожидается: ", a_val - b_val)
println("  Получено: ", digits_to_number(res_sub, BASE))

```

```

a_val, b_val = 123, 45
u = number_to_digits(a_val, BASE)
v = number_to_digits(b_val, BASE)
res_mul = mul_long_classic(u, v, BASE)
println("\nУмножение (школьное): $a_val * $b_val")
println("  Ожидается: ", a_val * b_val)
println("  Получено: ", digits_to_number(res_mul, BASE))

```

```

res_mul_fast = mul_long_fast(u, v, BASE)
println("\nУмножение (быстрое): $a_val * $b_val")
println("  Ожидается: ", a_val * b_val)
println("  Получено: ", digits_to_number(res_mul_fast, BASE))

```

```

a_val, b_val = 12345, 45
u = number_to_digits(a_val, BASE)
v = number_to_digits(b_val, BASE)
q_res, r_res = div_long(u, v, BASE)
println("\nДеление: $a_val / $b_val")
println("  Ожидается: Частное = $(div(a_val, b_val)), Остаток = $(mod(a_val, b_val))")
println("  Получено: Частное = $(digits_to_number(q_res, BASE)), Остаток = $(digits_to_number(r_res, BASE))")

```

end

Результат работы

Система счисления: 10

Сложение: $12345 + 6789$

Ожидается: 19134

Получено: 19134

Вычитание: $12345 - 6789$

Ожидается: 5556

Получено: 5556

Умножение (школьное): $123 \star 45$

Ожидается: 5535

Получено: 5535

Умножение (быстрое): $123 \star 45$

Ожидается: 5535

Получено: 5535

Деление: $12345 / 45$

Ожидается: Частное = 274, Остаток = 15

Получено: Частное = 274, Остаток = 15

3 Выводы

Мы изучили работу алгоритмов, а также реализовали их на языке Julia.