Отчет по лабораторной работе №8

1. Цель работы

Цель данной лабораторной работы – реализовать быстрые алгоритмы для сложения, вычитания, умножения столбиком, быстрого умножения и деления с большими числами.

2. Задание

Алгоритм 1 (сложение неотрицательных целых чисел)

Вход. Два неотрицательных числа $\mathbf{u = u_1\ u_2\ \ldots u}$□ и $\mathbf{v = v_1\ v_2\ \ldots v}$□; разрядность чисел *n*; основание системы счисления *b*.
Выход. Сумма $\mathbf{w = w_0\ w_1\ \ldots w}$□, где $\mathbf{w_0}$ — цифра переноса — всегда равная 0 либо 1.

1. Присвоить **j := n, k := 0** (j идет по разрядам, k следит за переносом).

2. Присвоить $\mathbf{w_j = (u_j + v_j + k)\ (mod\ b)}$, где $\mathbf{w_j}$ — наименьший неотрицательный вычет в данном классе вычетов;
   $\mathbf{k = \lfloor\ (u_j + v_j + k) / b \rfloor\ .}$

3. Присвоить **j := j − 1**. Если **j > 0**, то возвращаемся на шаг 2; если **j = 0**, то присвоить $\mathbf{w_0}$ **:= k** и результат: **w**.

Алгоритм 2 (вычитание неотрицательных целых чисел)

Вход. Два неотрицательных числа $\mathbf{u = u_1\ u_2\ \ldots u}$□ и $\mathbf{v = v_1\ v_2\ \ldots v}$□, **u > v**; разрядность чисел *n*; основание системы счисления *b*.
Выход. Разность $\mathbf{w = w_1\ w_2\ \ldots w}$□ **= u − v.**

1. Присвоить **j := n, k := 0** (k — заем из старшего разряда).

2. Присвоить $\mathbf{w_j = (u_j - v_j + k)\ (mod\ b)}$, где $\mathbf{w_j}$ — наименьший неотрицательный вычет в данном классе вычетов;
   $\mathbf{k = \lfloor\ (u_j - v_j + k) / b \rfloor\ .}$

3. Присвоить **j := j − 1**. Если **j > 0**, то возвращаемся на шаг 2; если **j = 0**, то результат: **w**.

Алгоритм 3 (умножение неотрицательных целых чисел столбиком)

Вход. Числа $\mathbf{u = u_1\ u_2\ \ldots u}$□, $\mathbf{v = v_1\ v_2\ \ldots v}$□; основание системы счисления *b*.
Выход. Произведение $\mathbf{w = uv = w_1\ w_2\ \ldots w}$□$_+$ □.

1. Выполнить присвоения:
   w_{m+1} := 0, w_m := 0, …, w_1 := 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
   $t := u_i \cdot v_j + w_{i+j} + k$, $w_{i+j} := t \pmod b$, $k := \lfloor t / b \rfloor$,
   где **w_{i+j}** — наименьший неотрицательный вычет в данном классе вычетов.

5. Присвоить **i := i − 1**. Если **i > 0**, то возвращаемся на шаг 4, иначе присвоить **w_j := k**.

6. Присвоить **j := j − 1**. Если **j > 0**, то вернуться на шаг 2. Если **j = 0**, то результат: **w**.

Алгоритм 4 (быстрый столбик)

Вход. Числа **$u = u_1 u_2 \ldots u_m$, $v = v_1 v_2 \ldots v_n$**; основание системы счисления *b*.
Выход. Произведение **$w = uv = w_1 w_2 \ldots w_{m+n}$**.

1. Присвоить **t := 0**.

2. Для **s от 0 до n + m − 1 с шагом 1** выполнить шаги 3 и 4.

3. Для **t = 0** до **n** выполнить присвоение
   $t := t + u_{s-t} \cdot v_t$.

4. Вычислить
   $w_{s+1} := t \pmod b$, $t := \lfloor t / b \rfloor$,
   где **w_{s+1}** — наименьший неотрицательный вычет по модулю *b*.
   Результат: **w**.

Алгоритм 5 (деление многоразрядных целых чисел)

Вход. Числа **$u = u_m \ldots u_1 u_0$** , **$v = v_n \ldots v_1 v_0$** , **$n \geq t \geq 1$, $v_n \neq 0$**, разрядность чисел соответственно **n** и **t**.
Выход. Частное **$q = q_{m-t} \ldots q_0$** , остаток **$r = r_n \ldots r_0$** .

1. Для **j** от 0 до **n − t** присвоить **q_j := 0**.

2. Пока **$u \geq v\, b^{n-t}$**, выполнять:
   $q_{m-t} := q_{m-t} + 1$, $u := u - v\, b^{n-t}$.

3. Для **i := n, n − 1, …, t + 1** выполнять пункты 3.1 – 3.4:
   3.1. Если **$u_i \geq v_n$**, то присвоить
   $q_{i-t-1} := b - 1$,

иначе присвоить
$q_{i-□-1} := \lfloor (u_i b + u_{i-1}) / v□ \rfloor$.

3.2. Пока
$q_{i-□-1} (v□ b + v□_{-1}) > u_i b^2 + u_{i-1} b + u_{i-2}$,
выполнять
$q_{i-□-1} := q_{i-□-1} - 1$.

3.3. Присвоить
$u := u - q_{i-□-1} v b^{i-t-1}$.

3.4. Если $u < 0$, то присвоить
$u := u + v b^{i-t-1}$, $q_{i-□-1} := q_{i-□-1} - 1$.

4. **r := u.**
   Результат: **q и r.**

---

3. Теоретическое введение

Будем считать, что число записано в *b*-ичной системе счисления, *b* — натуральное число, *b* ≥ *2*. Натуральное *b*-разрядное число будем записывать в виде:

$u = u_1 u_2 \ldots u□$.

Арифметика многократной точности (или произвольной точности) позволяет выполнять операции с целыми числами, превышающими размер стандартных типов данных, предоставляемых языками программирования. Это необходимо в различных областях, включая криптографию, научные вычисления и обработку больших данных.

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. Для представления больших чисел обычно используются массивы или списки, где каждый элемент соответствует отдельному разряду числа

4. Выполнение лабораторной работы

Функции, необходимые для упрощения работы и многократно использующиеся в алгоритмах

```
# Убираем нули
function remove_leading_zeros(number::Vector{Int})
    while length(number) > 1 && number[1] == 0
        popfirst!(number)
    end
    return number
end

# Функция сравнения
```

```julia
function compare(u::Vector{Int}, v::Vector{Int})::Int
    u = remove_leading_zeros(copy(u))
    v = remove_leading_zeros(copy(v))

    if length(u) > length(v)
        return 1
    elseif length(u) < length(v)
        return -1
    else
        for (digit_u, digit_v) in zip(u, v)
            if digit_u > digit_v
                return 1
            elseif digit_u < digit_v
                return -1
            end
        end
        return 0
    end
end

# Функция добавления нулей

function pad_with_leading_zeros(u::Vector{Int},
v::Vector{Int})::Tuple{Vector{Int}, Vector{Int}}
    max_len = max(length(u), length(v))
    u_padded = vcat(zeros(Int, max_len - length(u)), u)
    v_padded = vcat(zeros(Int, max_len - length(v)), v)
    return (u_padded, v_padded)
end

# Перевод числа в вектор цифр

function number_to_digits(number::Union{String, Int})::Vector{Int}
    if isa(number, Int)
        number = string(number)
    elseif isa(number, String)
        if !all(c -> isdigit(c), number)
            error("Число должно содержать только цифры.")
        end
    else
        error("Число должно быть строкой или целым числом.")
    end
    return [parse(Int, string(c)) for c in number]
end

# Перевод цифр в число

function digits_to_number(digits::Vector{Int})::String
    return join(string.(digits))
end
function number_to_digits_func(number::Union{String, Int})::Vector{Int}
    return number_to_digits(number)
```

```
end

function digits_to_number_func(digits::Vector{Int})::String
    return digits_to_number(digits)
end
```

```
digits_to_number_func (generic function with 1 method)
```

Алгоритм 1 (сложение неотрицательных целых чисел)

```
function add(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    u_padded, v_padded = pad_with_leading_zeros(u,v)
    n = length(u_padded)
    w = zeros(Int, n+1)
    k = 0

    for j in n:-1:1
        total = u_padded[j] + v_padded[j] + k
        w[j+1] = total % b
        k = div(total,b)
    end
    w[1] = k
    return w
end
```

```
add (generic function with 1 method)
```

Пример сложения

```
u_num  = "123456789012345"
v_num = "987654321098765"
b = 10
u = number_to_digits_func(u_num)
v = number_to_digits_func(v_num)
sum_result = add(u, v, b)
sum_str = digits_to_number_func(sum_result)
println("u = $u_num")
println("v = $v_num")
println("w = $sum_str")
```

```
u = 123456789012345
v = 987654321098765
w = 1111111110111110
```

Алгоритм 2 (вычитание неотрицательных целых чисел)

```
function subtract(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    u_padded, v_padded = pad_with_leading_zeros(u, v)
    n = length(u_padded)
    w = zeros(Int, n)
```

```
    k = 0

    for j in n:-1:1
        diff = u_padded[j] - v_padded[j] + k
        if diff < 0
            diff += b
            k = -1
        else
            k = 0
        end
        w[j] = diff
    end
    return remove_leading_zeros(w)
end
```

```
subtract (generic function with 1 method)
```

Пример вычитания

```
u_num  = "987654321098765"
v_num = "123456789012345"
b = 10
u = number_to_digits_func(u_num)
v = number_to_digits_func(v_num)
diff_result = subtract(u, v, b)
diff_str = digits_to_number_func(sum_result)
println("u = $u_num")
println("v = $v_num")
println("w = $sum_str")
```

```
u = 987654321098765
v = 123456789012345
w = 1111111110111110
```

Алгоритм 3 (умножение неотрицательных целых чисел столбиком)

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

    for j in m:-1:1
        carry = 0
        for i in n:-1:1
            t = u[i] * v[j] + w[i + j] + carry
            w[i + j] = t % b
            carry = div(t, b)
        end
        w[j] += carry
    end
```

```
    return remove_leading_zeros(w)
end
```

```
multiply_long (generic function with 1 method)
```

Пример умножения

```
u_num = "123456789012345"
v_num = "678901234567890"
u = number_to_digits_func(u_num)
v = number_to_digits_func(v_num)
product_long = multiply_long(u, v, b)
product_long_str = digits_to_number_func(product_long)
println("u = $u_num")
println("v = $v_num")
println("w = $product_long_str")
```

```
u = 123456789012345
v = 678901234567890
w = 838149664762685376459506020 50
```

Алгоритм 4 (быстрый столбик)

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

    for s in (n + m):-1:1
        t = 0
        for t_idx in 1:n
            v_idx = s - t_idx + 1
            if v_idx >= 1 && v_idx <= m
                t += u[n - t_idx + 1] * v[m - v_idx + 1]
            end
        end
        w[s] += t
        carry = div(w[s], b)
        w[s] = mod(w[s], b)
        if s > 1
            w[s - 1] += carry
        end
    end

    return remove_leading_zeros(w)
end
```

```
multiply_fast (generic function with 1 method)
```

Пример быстрого умножения

```
product_fast = multiply_fast(u, v, b)
product_fast_str = digits_to_number_func(product_fast)
println("u = $u_num")
println("v = $v_num")
println("w = $product_fast_str")
println()
```

```
u = 123456789012345
v = 678901234567890
w = 536504679225115307111781741960
```

Алгоритм 5 (деление многоразрядных целых чисел)

```
function divide(u::Vector{Int}, v::Vector{Int}, b::Int)::Tuple{Vector{Int},
Vector{Int}}
    if compare(v, [0]) == 0
        error("Деление на ноль невозможно.")
    end

    if compare(u, v) < 0
        return ([0], copy(u))
    end

    n = length(u)
    t_len = length(v)
    q = zeros(Int, n - t_len + 1)
    u_copy = copy(u)

    for i in 1:(n - t_len + 1)
        start = i
        end_idx = i + t_len - 1
        if end_idx > length(u_copy)
            current = [0]
        else
            current = u_copy[start:end_idx]
        end
        current = remove_leading_zeros(current)
        if length(current) < t_len
            current = vcat(zeros(Int, t_len - length(current)), current)
        end

        if current[1] < v[1]
            q_digit = 0
        else
            q_digit = div(current[1], v[1])
            q_digit = min(q_digit, b - 1)
        end

        while true
            v_times_q = multiply_long(v, [q_digit], b)
            v_times_q = vcat(v_times_q, zeros(Int, length(current) -
```

```
length(v_times_q)))
            cmp = compare(v_times_q, current)
            if cmp > 0
                q_digit -= 1
            else
                break
            end
            if q_digit == 0
                break
            end
        end

        v_times_q = multiply_long(v, [q_digit], b)
        v_times_q = vcat(v_times_q, zeros(Int, length(current) -
length(v_times_q)))
        current = subtract(current, v_times_q, b)

        u_copy[start:end_idx] = current
        q[i] = q_digit
    end

    q = remove_leading_zeros(q)
    r = remove_leading_zeros(u_copy)
    return (q, r)
end

divide (generic function with 1 method)
```

Пример деления

```
u_num = "123456789012345678901234 5"
v_num = "123456789012345"
u = number_to_digits_func(u_num)
v = number_to_digits_func(v_num)
quotient, remainder = divide(u, v, b)
quotient_str = digits_to_number_func(quotient)
remainder_str = digits_to_number_func(remainder)
println("u = $u_num")
println("v = $v_num")
println("q = $quotient_str")
println("r = $remainder_str")
```

5. Выводы

В ходе данной лабораторной работы было рассмотрено 5 алгоритмов, обспечивающих более высокую производительность машинного сложения, вычитание, умножения и деления больших чисел. Данные алгоритмы были программно реализованы на языке ***Julia***. Данные программы могут быть использованы для работы с числами любой счетной системы (в частности десятичной) и превышающими размер стандартных типов данных.

6. Список литературы

1. Knuth, D. E. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, 1997.
2. Sedgewick, R., Wayne, K. Algorithms. Addison-Wesley, 2011.
3. Liu, Y. Chinese Remainder Theorem and Its Applications. Springer, 2008.
4. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. MIT Press, 2009.
5. McIlroy, M. D. Implementing the Chinese Remainder Theorem, 1981.
6. Буряков, В. Г. Алгоритмы и структуры данных. М.: Наука, 2004.
7. Julia Documentation. https://docs.julialang.org/