Цель работы

Основной целью работы является реализовать разными алгоритмами целочисленную арифметику многократной точности.

Выполнение лабораторной работы

Алгоритм 1 (сложение неотрицательных целых чисел)

```
function add_bigint(u::Vector{Int}, v::Vector{Int}, b::Int)
    n = max(length(u), length(v))
    u = vcat(zeros(Int, n - length(u)), u)
    v = vcat(zeros(Int, n - length(v)), v)
    w = Vector{Int}(undef, n)
    k = 0  # перенос (carry)
    for j in n:-1:1
        s = u[j] + v[j] + k
        w[j] = s % b         # цифра в j-ом разряде
        k = s ÷ b            # перенос
    end
    return k > 0 ? vcat([k], w) : w
end
b = 10
u = [1, 7, 1, 1, 1, 2]  # число 171112
v = [9, 8, 7, 6, 5, 4]  # число 987654
result = add_bigint(u, v, b)
println("Сумма = ", result, " → ", join(result, ""))  # для красоты печатаем слитно
```

```
Сумма = [1, 1, 5, 8, 7, 6, 6] → 1158766
```

Алгоритм 2 (вычитание неотрицательных целых чисел)

```
function sub_bigint(u::Vector{Int}, v::Vector{Int}, b::Int)
    # 1. Выравниваем длины чисел (добавляем ведущие нули к v)
    n = max(length(u), length(v))
    u = vcat(zeros(Int, n - length(u)), u)
    v = vcat(zeros(Int, n - length(v)), v)
    w = Vector{Int}(undef, n)
    k = 0  # заем (может быть -1, 0)
    for j in n:-1:1
        diff = u[j] - v[j] + k
        if diff < 0
            diff += b    # добавляем "заем" из старшего разряда
            k = -1       # следующий разряд уменьшится на 1
        else
            k = 0
        end
        w[j] = diff
    end
    while length(w) > 1 && w[1] == 0
        popfirst!(w)
    end
    return w
end
b = 10
u = [1, 7, 1, 1, 1, 2]  # 171112
v = [1, 2, 1, 1, 1, 2]  # 121112
result = sub_bigint(u, v, b)
println("Разность = ", result, " → ", join(result, ""))  # для красоты печатаем слитно
```

```
Разность = [5, 0, 0, 0, 0] → 50000
```

Алгоритм 3 (умножение неотрицательных целых чисел столбиком)

```
function multiply_column(u::Vector{Int}, v::Vector{Int}, b::Int)
    m = length(u)
    n = length(v)
    w = zeros(Int, m + n)
    for j = n:-1:1
        if v[j] == 0
            continue
        end
        K = 0  # перенос
        for i = m:-1:1
            t = u[i] * v[j] + w[i+j] + K
            w[i+j] = t % b        # остаток от деления на основание
            K = div(t, b)         # новый перенос
        end
        w[j] += K
    end
    return w
end
u = [1,2,3]  # 123
v = [4,5]    # 45
b = 10
result = multiply_column(u, v, b)
println("Результат: ", result)  # Должно вывести цифры числа 5535 ([5,5,3,5])
```

```
Результат: [0, 5, 5, 3, 5]
```

Алгоритм 4 (быстрый столбик)

```
function fast_column_multiply(u::Vector{Int}, v::Vector{Int}, b::Int)
    m = length(u)
    n = length(v)
    w = zeros(Int, m + n)  # массив для результата
    for j = n:-1:1
        if v[j] == 0
            continue
        end
        K = 0  # перенос
        for i = m:-1:1
            t = u[i] * v[j] + w[i+j] + K
            w[i+j] = t % b
            K = div(t, b)
        end
        w[j] += K  # добавляем остаток переноса
    end
    while length(w) > 1 && w[1] == 0
        w = w[2:end]
    end
    return w
end
u = [1,2,3]  # число 123
v = [4,5]    # число 45
b = 10
result = fast_column_multiply(u, v, b)
println("Результат: ", result)  # [5,5,3,5] -> 5535
```

```
Результат: [5, 5, 3, 5]
```

Алгоритм 5 (деление многоразрыдных целых чисел)

```
        end
        v_num = 0
        for d in v
            v_num = v_num * b + d
        end
        qt = div(u_num, v_num)
        qt = min(qt, b-1)  # не больше основания
        carry = 0
        for j = 1:n
            idx = i - n + j
            r[idx] -= qt * v[j] + carry
            if r[idx] < 0
                carry = div(-r[idx] + b - 1, b)
                r[idx] += carry * b
            else
                carry = 0
            end
        end
        while r[i-n+1] < 0
            carry = 0
            for j = 1:n
                idx = i - n + j
                r[idx] += v[j] + carry
                if r[idx] >= b
                    carry = div(r[idx], b)
                    r[idx] %= b
                else
                    carry = 0
                end
            end
            qt -= 1
        end
        q[i-n+1] = qt
    end
    return q, r
end
u = [1,2,3,4]  # делимое 1234
v = [1,2]      # делитель 12
b = 10
q, r = long_division(u, v, b)
println("Частное: ", q)
println("Остаток: ", r)
```

```
Частное: [0, 1, 2]
Остаток: [1, 1, 9, 0]
```

Вывод

В ходе выполнения лабораторной работы было реализовано разными алгоритмами целочисленную арифметику многократной точности.