

# Лабораторная работа №8

Тема: Целочисленная арифметика многократной точности

Выполнила: Исламова Сания Маратовна

Группа: НПИмд-01-24

Студ.билет: 1132249576

## Задача лабораторной работы:

Реализовать рассмотренные алгоритмы программно: Сложение неотрицательных целых чисел, Вычитание неотрицательных целых чисел, Умножение неотрицательных целых чисел столбиком, Быстрый столбик, Деление многоразрядных целых числе

## Описание хода выполнения лабораторной работы:

```
# Функция преобразования массива цифр в строку в системе счисления b
function digits_to_str(dig::Vector{Int}, b::Int)::String
    # Если массив пустой или состоит только из нулей — число равно 0
    if isempty(dig) || all==(0), dig)
        return "0"
    end
    # Собираем строку из цифр, начиная со старшего разряда (reverse)
    # Если цифра ≥10, преобразуем в букву A–Z (для оснований >10)
    join([d < 10 ? string(d) : string(Char('A' + d - 10)) for d in reverse(dig)], "")
    # Результат: строка, представляющая число в основании b, без ведущих нулей
end

# Функция преобразования строки в массив цифр (младший разряд — индекс 1)
function str_to_digits(s::String, b::Int)::Vector{Int}
    # Убираем пробелы и приводим к нижнему регистру для удобства
    s = lowercase(s)
    # Пустая строка или "0" → представляем как [0]
    if s == "" || s == "0" return [0] end
    digits = Int[] # Создаём пустой массив для цифр
    # Проходим по символам строки справа налево (младшие разряды первые)
    for c in reverse(s)
        # Преобразуем символ в цифру: 0–9 или A–Z → 10–35
        d = isdigit(c) ? c - '0' : (uppercase(c) - 'A' + 10)
        # Проверяем, что цифра допустима в данном основании
        if d < 0 || d >= b
            error("Недопустимая цифра '$c' в основании $b")
        end
        push!(digits, d) # Добавляем цифру в массив (младшая первая)
    end
    # Удаляем ведущие нули (кроме случая, когда число 0)
    while length(digits) > 1 && digits[end] == 0
        pop!(digits)
    end
    digits # Результат : массив цифр , младший разряд — digits[1]
end

# Удаление ведущих нулей из массива цифр
function trim(dig::Vector{Int})::Vector{Int}
    # Пока больше одного разряда и старший разряд нулевой — удаляем его
    while length(dig) > 1 && dig[end] == 0
        pop!(dig)
    end
    dig # Результат: массив без ведущих нулей (кроме [0] для нуля)
end

# Алгоритм 1: Сложение неотрицательных чисел (по лабораторной)
```

```

function add_big(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = max(length(u), length(v)) # Определяем максимальную разрядность
    w = zeros(Int, n + 1) # Результирующий массив на один разряд больше (для переноса)
    k = 0 # Перенос, изначально 0
    for j = 1:n # Проходим по разрядам от младшего к старшему
        uj = j <= length(u) ? u[j] : 0 # Берем цифру из u или 0, если разряд кончился
        vj = j <= length(v) ? v[j] : 0 # Аналогично для v
        s = uj + vj + k # Сумма цифр + перенос
        w[j] = s % b # Записываем младшую часть суммы в текущий разряд
        k = s ÷ b # Новый перенос в старший разряд
    end
    w[n+1] = k # Записываем финальный перенос (w[0] в алгоритме)
    trim(w) # Убираем ведущие нули
    # Результат: сумма u + v в основании b
end

```

# Алгоритм 2: Вычитание u - v ( $u \geq v \geq 0$ )

```

function sub_big(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = length(u) # Разрядность берём по первому числу ( $u \geq v$ )
    w = zeros(Int, n) # Результирующий массив
    k = 0 # Заём из старшего разряда, изначально 0
    for j = 1:n
        uj = u[j] # Цифра из уменьшаемого
        vj = j <= length(v) ? v[j] : 0 # Цифра из вычитаемого или 0
        s = uj - vj - k # Разность с учётом займа
        if s < 0 # Если получилась отрицательная цифра
            s += b # Занимаем из старшего разряда (добавляем основание)
            k = 1 # Устанавливаем заём для следующего разряда
        else
            k = 0 # Заём не нужен
        end
        w[j] = s # Записываем цифру результата
    end
    trim(w) # Убираем ведущие нули
    # Результат: разность u - v в основании b
end

```

# Алгоритм 3: Умножение "столбиком"

```

function mul_classic(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n, m = length(u), length(v) # Разрядности множимого и множителя
    w = zeros(Int, n + m) # Результат до n+m разрядов
    for j = 1:m # По разрядам множителя v (от младшего)
        vjj == 0 && continue # Если цифра 0 — пропускаем (оптимизация)
        k = 0 # Перенос для текущего "столбика"
        for i = 1:n # По разрядам множимого u
            t = u[i] * v[j] + w[i+j-1] + k # Произведение + уже накопленное + перенос
            w[i+j-1] = t % b # Записываем в текущий разряд
            k = t ÷ b # Перенос в следующий разряд
        end
        w[n+j] = k # Записываем оставшийся перенос
    end
    trim(w) # Убираем ведущие нули
    # Результат: произведение u × v в основании b
end

```

# Алгоритм 4: Умножение "быстрым столбиком"

```

function mul_fast(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n, m = length(u), length(v)

```

```

w = zeros(Int, n + m)      # Результат до n+m разрядов
t = 0                      # Накопитель промежуточной суммы
for s = 0:n+m-2            # s — диагональ в "столбике"
    low = max(0, s - m + 1) # Нижняя граница индекса i
    high = min(s, n - 1)   # Верхняя граница индекса i
    for i = low:high        # Суммируем все произведения на этой диагонали
        t += u[i+1] * v[s-i+1]
    end
    w[n+m-s-1] = t % b    # Записываем цифру в соответствующий разряд
    t ÷= b                # Переносим остаток в следующую диагональ
end
w[1] = t                  # Последний перенос в старший разряд
trim(w)
# Результат: произведение u × v (тот же, что и в mul_classic, но другой алгоритм)
end

```

```

# Алгоритм 5: Деление с остатком
function div_big(u_::Vector{Int}, v_::Vector{Int}, b::Int)
    u = copy(u_)          # Копируем, чтобы не изменять оригинал
    v = trim(copy(v_))    # Копируем и убираем ведущие нули у делителя
    n, t = length(u), length(v) # Разрядности делимого и делителя
    if t == 0 || all(==(0), v) # Проверка деления на ноль
        error("Деление на ноль")
    end
    q = zeros(Int, n - t + 1) # Массив для частного (максимальная длина)
    for i = n:-1:t+1         # По разрядам делимого от старшего
        hi = i <= n ? u[i] : 0 # Текущий старший разряд
        mi = i-1 >= 1 ? u[i-1] : 0 # Следующий разряд
        lo = i-2 >= 1 ? u[i-2] : 0 # Ещё один для точной оценки
        # Оценка цифры частного
        qhat = hi >= v[t] ? b - 1 : (hi * b + mi) ÷ v[t]
        v1 = v[t] * b + (t >= 2 ? v[t-1] : 0) # Для проверки переполнения
        # Корректируем оценку вниз, если слишком большая
        while qhat > 0 && qhat * v1 > hi * b*b + mi * b + lo
            qhat -= 1
        end
        borrow = 0              # Заём при вычитании
        for j = 1:t              # Вычитаем qhat × v × b^(i-t)
            pos = i - t + j
            if pos > length(u) # Если нужно — расширяем массив u
                resize!(u, pos)
                u[pos] = 0
            end
            temp = qhat * v[j] + borrow
            u[pos] -= temp % b
            borrow = temp ÷ b
            if u[pos] < 0        # Если отрицательно — занимаем
                u[pos] += b
                borrow += 1
            end
        end
        pos_carry = i + 1
        if pos_carry <= length(u)
            u[pos_carry] -= borrow # Вычитаем заём из старшего разряда
            if u[pos_carry] < 0    # Если переполнение — корректируем
                u[pos_carry] += b
                qhat -= 1
            end
        end
    end
end

```

```

end
q[i - t] = qhat      # Записываем цифру частного
end
r = length(u) >= t ? u[1:t] : u # Остаток — младшие t разрядов
trim(q), trim(r)      # Убираем ведущие нули
# Результат: кортеж (частное, остаток)
end

println ( "Лабораторная работа №8: Арифметика многократной точности")
println ( "0 — Выход")
println ( "1 — Сложение")
println ( "2 — Вычитание")
println ( "3 — Умножение столбиком")
println ( "4 — Умножение быстрым столбиком")
println ( "5 — Деление с остатком\n")

# Бесконечный цикл — программа работает, пока пользователь не выберет выход.
while true
    print("Выберите алгоритм (0 для выхода): ")
    input = strip(readline())          # Считываем и убираем лишние пробелы
    if input == "0" && (println("До свидания!"); break) # Выход по 0

    # Преобразуем ввод в число, если ошибка — сообщаем и продолжаем цикл
    alg = try parse(Int, input) catch
        println("Неверный выбор\n"); continue
    end
    if !(1 <= alg <= 5)            # Проверяем диапазон
        println("Выберите от 1 до 5\n"); continue
    end

    print("Основание | b (2–36): ")
    b = try parse(Int, readline()) catch    # Считываем основание
        println("Неверное основание\n"); continue
    end
    if !(2 <= b <= 36)            # Проверяем допустимость
        println("b должно быть от 2 до 36\n"); continue
    end

    print("Первое число (в системе $b): ")
    s1 = readline()                # Ввод первого числа как строки
    print("Второе число (в системе $b): ")
    s2 = readline()                # Ввод второго числа

    try
        # Преобразуем строки в массивы цифр
        u = str_to_digits(s1, b)
        v = str_to_digits(s2, b)

        # Выполняем выбранный алгоритм
        if alg == 1
            res = add_big(u, v, b)
            println("Сумма : $(digits_to_str(res, b))\n")
        elseif alg == 2
            # Проверка условия u ≥ v для вычитания
            if length(u) < length(v) || (length(u) == length(v) && u < v)
                println("Ошибка: первое число должно быть ≥ второго\n")
            else

```

```

res = sub_big(u, v, b)
println("Разность : $(digits_to_str(res, b))\n")
end
elseif alg == 3
    res = mul_classic(u, v, b)
    println("Произведение (столбиком) : $(digits_to_str(res, b))\n")
elseif alg == 4
    res = mul_fast(u, v, b)
    println("Произведение (быстро) : $(digits_to_str(res, b))\n")
elseif alg == 5
    q, r = div_big(u, v, b)
    println("Частное : $(digits_to_str(q, b))")
    println("Остаток : $(digits_to_str(r, b))\n")
end
catch e
    # Ловим все ошибки ввода (недопустимые цифры и т.д.)
    println( "Ошибка : $(sprint(showerror, e))\n")
end
end

```

## Результат реализации рассмотренного алгоритма

```

Lab08j
C:\Users\4ekal0\Downloads> Lab08j > ...
1  # Лабораторная работа №8:
2  # Тема: Целочисленная арифметика многократной точности
3  # Выполнена: Исламова Сания Маратовна
4  # Группа: НИФД-01-24
5

6  # Функция преобразования массива цифр в строку в системе счисления b
7  function digits_to_str(dig:vector[Int], b:Int):String
8      # Если массив пустой или состоит только из нулей – число равно 0
9      if isempty(dig) || all(==0, dig)
10         return "0"
11     end
12
13     # Собираем строку из цифр, начиная со старшего разряда (reverse)
14     # Если цифра >10, преобразуем в буквы А-З (для оснований >10)
15     join((d > 10 ? str(d) : string(Char("A" + d - 10))) for d in reverse(dig), "")
16     # Результат: строка, представляющая число в основании b, без ведущих нулей
17 end
18
19 # Функция преобразования строки в массив цифр (младший разряд – индекс 1)
20 function str_to_digits(::String, b:Int)::Vector[Int]
21     # Убираем пробелы и приводим к нижнему регистру для удобства
22     s = lowercase(strip(s))
23     # Пускем цикл, считывая цифры – представляем как [0]
24     if s == "" || s == "0" return [0] end
25     digits = Int[] # Создаём пустой массив для цифр
26     # Проходим по символам строки справа налево (младшие разряды первые)
27     for c in reverse(s)
28         # Преобразуем символ в цифру: 0-9 или A-Z → 10-35
29         d = isdigit(c) ? c - '0' : (uppercase(c) - 'A' + 10)
30         # Проверяем, что цифра допустима в данном основании
31         if d < 0 || d > b
32             error("Недопустимая цифра '$c' в основании $b")
33         end
34         push!(digits, d) # Добавляем цифру в массив (младшая первая)
35     end
36     # Удаляем ведущие нули (кроме случая, когда число 0)
37     while length(digits) > 1 && digits[end] == 0
38         pop!(digits)
39     end
40     digits # Результат: массив цифр, младший разряд – digits[1]
41 end
42
43 # Удаление ведущих нулей из массива цифр
44 function trim(dig:Vector[Int]):Vector[Int]
45     # Пока больше одного разряда и старший разряд нулевой – удаляем его
46     while length(dig) > 1 && dig[end] == 0
47         pop!(dig)
48     end
49     dig # Результат: массив без ведущих нулей (кроме [0] для нуля)
50 end
51
52 # Алгоритм 1: Сложение неотрицательных чисел (по лабораторной)
53 function add_big(u:Vector[Int], v:Vector[Int], b:Int)::Vector[Int]
54     n = max(length(u), length(v)) # Определяем максимальную разность
55     w = zeros(Int, n + 1) # Результирующий массив на один разряд больше (для переноса)
56     k = 0 # Перенос, изначально 0
57     for j = 1:n
58         uj = j <= length(u) ? u[j] : 0 # Берём цифру u или 0, если разряд кончился
59         vj = j <= length(v) ? v[j] : 0 # Аналогично для v
60         s = uj + vj + k # Сумма цифр + перенос
61         w[j] = s % b # Записываем младшую часть суммы в текущий разряд
62         k = s ÷ b # Новый перенос в старший разряд
63     end
64     w[n+1] = k # Записываем финальный перенос (что в алгоритме)
65     trim(w) # Убираем ведущие нули
66     # Результат: сумма u + v в основании b
67 end
68
69 # Алгоритм 2: Вычитание u - v (u ≥ v)
70 function sub_big(u:Vector[Int], v:Vector[Int], b:Int)::Vector[Int]
71     n = length(u) # Разность берётся по первому числу (u ≥ v)
72     w = zeros(Int, n) # Результирующий массив
73     k = 0 # Займ из старшего разряда, изначально 0
74     for j = 1:n
75         uj = u[j] # Цифра из уменьшаемого
76         vj = j <= length(v) ? v[j] : 0 # Цифра из вычитаемого или 0
77         s = uj - vj - k # Разность с учётом займа
78         if s < 0 # Если получилась отрицательная цифра
79             s += b # Занимаем из старшего разряда (добавляем основание)
80             k = 1 # Устанавливаем займ для следующего разряда
81         else
82             k = 0 # Займ не нужен
83         end
84         w[j] = s # Записываем цифру результата
85     end
86     trim(w) # Убираем ведущие нули
87     # Результат: разность u - v в основании b
88 end

```

```

Lab08j
C:\Users\4ekal0\Downloads> Lab08j > ...
44 function trim(dig:Vector[Int]):Vector[Int]
45     # Пока больше одного разряда и старший разряд нулевой – удаляем его
46     while length(dig) > 1 && dig[end] == 0
47         pop!(dig)
48     end
49     dig # Результат: массив без ведущих нулей (кроме [0] для нуля)
50 end
51
52 # Алгоритм 1: Сложение неотрицательных чисел (по лабораторной)
53 function add_big(u:Vector[Int], v:Vector[Int], b:Int)::Vector[Int]
54     n = max(length(u), length(v)) # Определяем максимальную разность
55     w = zeros(Int, n + 1) # Результирующий массив на один разряд больше (для переноса)
56     k = 0 # Перенос, изначально 0
57     for j = 1:n
58         uj = j <= length(u) ? u[j] : 0 # Берём цифру u или 0, если разряд кончился
59         vj = j <= length(v) ? v[j] : 0 # Аналогично для v
60         s = uj + vj + k # Сумма цифр + перенос
61         w[j] = s % b # Записываем младшую часть суммы в текущий разряд
62         k = s ÷ b # Новый перенос в старший разряд
63     end
64     w[n+1] = k # Записываем финальный перенос (что в алгоритме)
65     trim(w) # Убираем ведущие нули
66     # Результат: сумма u + v в основании b
67 end
68
69 # Алгоритм 2: Вычитание u - v (u ≥ v)
70 function sub_big(u:Vector[Int], v:Vector[Int], b:Int)::Vector[Int]
71     n = length(u) # Разность берётся по первому числу (u ≥ v)
72     w = zeros(Int, n) # Результирующий массив
73     k = 0 # Займ из старшего разряда, изначально 0
74     for j = 1:n
75         uj = u[j] # Цифра из уменьшаемого
76         vj = j <= length(v) ? v[j] : 0 # Цифра из вычитаемого или 0
77         s = uj - vj - k # Разность с учётом займа
78         if s < 0 # Если получилась отрицательная цифра
79             s += b # Занимаем из старшего разряда (добавляем основание)
80             k = 1 # Устанавливаем займ для следующего разряда
81         else
82             k = 0 # Займ не нужен
83         end
84         w[j] = s # Записываем цифру результата
85     end
86     trim(w) # Убираем ведущие нули
87     # Результат: разность u - v в основании b
88 end

```

```

1 # Lab05
2
3 # C:\Users\Aselat> delphi7 Downloads> Lab05\obj\... 
4
5 # Алгоритм 1: Умножение "столбиком"
6 function mul_classic( Vector1:vector[int], b1size:integer; Vector2:vector[int] )
7 begin
8   var s:integer;
9   w:array[0..n-1] of integer;
10  t:integer;
11  x:integer;
12  i,j,k:integer;
13
14  if b1size <= 0 then
15    begin
16      writeln('Результат для пустого разряда');
17      exit;
18    end;
19
20  for j := 1 to n do
21    begin
22      s := 0;
23      for i := 1 to b1size do
24        begin
25          k := 0;
26          for x := 0 to m-1 do
27            begin
28                k := k + w[x]*b1[i,x];
29            end;
30            s := s + k;
31            k := 0;
32        end;
33        w[x] := s;
34        s := 0;
35      end;
36      if w[0] > 9 then
37        begin
38          t := w[0] div 10;
39          w[0] := w[0] mod 10;
40          w[1] := w[1] + t;
41        end;
42      end;
43      if w[0] > 9 then
44        begin
45          t := w[0] div 10;
46          w[0] := w[0] mod 10;
47          w[1] := w[1] + t;
48        end;
49      end;
50      if w[1] > 9 then
51        begin
52          t := w[1] div 10;
53          w[1] := w[1] mod 10;
54          w[2] := w[2] + t;
55        end;
56      end;
57      if w[2] > 9 then
58        begin
59          t := w[2] div 10;
60          w[2] := w[2] mod 10;
61          w[3] := w[3] + t;
62        end;
63      end;
64      if w[3] > 9 then
65        begin
66          t := w[3] div 10;
67          w[3] := w[3] mod 10;
68          w[4] := w[4] + t;
69        end;
70      end;
71      if w[4] > 9 then
72        begin
73          t := w[4] div 10;
74          w[4] := w[4] mod 10;
75          w[5] := w[5] + t;
76        end;
77      end;
78      if w[5] > 9 then
79        begin
80          t := w[5] div 10;
81          w[5] := w[5] mod 10;
82          w[6] := w[6] + t;
83        end;
84      end;
85      if w[6] > 9 then
86        begin
87          t := w[6] div 10;
88          w[6] := w[6] mod 10;
89          w[7] := w[7] + t;
90        end;
91      end;
92      if w[7] > 9 then
93        begin
94          t := w[7] div 10;
95          w[7] := w[7] mod 10;
96          w[8] := w[8] + t;
97        end;
98      end;
99      if w[8] > 9 then
100        begin
101          t := w[8] div 10;
102          w[8] := w[8] mod 10;
103          w[9] := w[9] + t;
104        end;
105      end;
106      if w[9] > 9 then
107        begin
108          t := w[9] div 10;
109          w[9] := w[9] mod 10;
110          w[10] := w[10] + t;
111        end;
112      end;
113      if w[10] > 9 then
114        begin
115          t := w[10] div 10;
116          w[10] := w[10] mod 10;
117          w[11] := w[11] + t;
118        end;
119      end;
120      if w[11] > 9 then
121        begin
122          t := w[11] div 10;
123          w[11] := w[11] mod 10;
124          w[12] := w[12] + t;
125        end;
126      end;
127      if w[12] > 9 then
128        begin
129          t := w[12] div 10;
130          w[12] := w[12] mod 10;
131          w[13] := w[13] + t;
132        end;
133      end;
134      if w[13] > 9 then
135        begin
136          t := w[13] div 10;
137          w[13] := w[13] mod 10;
138          w[14] := w[14] + t;
139        end;
140      end;
141      if w[14] > 9 then
142        begin
143          t := w[14] div 10;
144          w[14] := w[14] mod 10;
145          w[15] := w[15] + t;
146        end;
147      end;
148      if w[15] > 9 then
149        begin
150          t := w[15] div 10;
151          w[15] := w[15] mod 10;
152          w[16] := w[16] + t;
153        end;
154      end;
155      if w[16] > 9 then
156        begin
157          t := w[16] div 10;
158          w[16] := w[16] mod 10;
159          w[17] := w[17] + t;
160        end;
161      end;
162      if w[17] > 9 then
163        begin
164          t := w[17] div 10;
165          w[17] := w[17] mod 10;
166          w[18] := w[18] + t;
167        end;
168      end;
169      if w[18] > 9 then
170        begin
171          t := w[18] div 10;
172          w[18] := w[18] mod 10;
173          w[19] := w[19] + t;
174        end;
175      end;
176      if w[19] > 9 then
177        begin
178          t := w[19] div 10;
179          w[19] := w[19] mod 10;
180          w[20] := w[20] + t;
181        end;
182      end;
183      if w[20] > 9 then
184        begin
185          t := w[20] div 10;
186          w[20] := w[20] mod 10;
187          w[21] := w[21] + t;
188        end;
189      end;
190      if w[21] > 9 then
191        begin
192          t := w[21] div 10;
193          w[21] := w[21] mod 10;
194          w[22] := w[22] + t;
195        end;
196      end;
197      if w[22] > 9 then
198        begin
199          t := w[22] div 10;
200          w[22] := w[22] mod 10;
201          w[23] := w[23] + t;
202        end;
203      end;
204      if w[23] > 9 then
205        begin
206          t := w[23] div 10;
207          w[23] := w[23] mod 10;
208          w[24] := w[24] + t;
209        end;
210      end;
211      if w[24] > 9 then
212        begin
213          t := w[24] div 10;
214          w[24] := w[24] mod 10;
215          w[25] := w[25] + t;
216        end;
217      end;
218      if w[25] > 9 then
219        begin
220          t := w[25] div 10;
221          w[25] := w[25] mod 10;
222          w[26] := w[26] + t;
223        end;
224      end;
225      if w[26] > 9 then
226        begin
227          t := w[26] div 10;
228          w[26] := w[26] mod 10;
229          w[27] := w[27] + t;
230        end;
231      end;
232      if w[27] > 9 then
233        begin
234          t := w[27] div 10;
235          w[27] := w[27] mod 10;
236          w[28] := w[28] + t;
237        end;
238      end;
239      if w[28] > 9 then
240        begin
241          t := w[28] div 10;
242          w[28] := w[28] mod 10;
243          w[29] := w[29] + t;
244        end;
245      end;
246      if w[29] > 9 then
247        begin
248          t := w[29] div 10;
249          w[29] := w[29] mod 10;
250          w[30] := w[30] + t;
251        end;
252      end;
253      if w[30] > 9 then
254        begin
255          t := w[30] div 10;
256          w[30] := w[30] mod 10;
257          w[31] := w[31] + t;
258        end;
259      end;
260      if w[31] > 9 then
261        begin
262          t := w[31] div 10;
263          w[31] := w[31] mod 10;
264          w[32] := w[32] + t;
265        end;
266      end;
267      if w[32] > 9 then
268        begin
269          t := w[32] div 10;
270          w[32] := w[32] mod 10;
271          w[33] := w[33] + t;
272        end;
273      end;
274      if w[33] > 9 then
275        begin
276          t := w[33] div 10;
277          w[33] := w[33] mod 10;
278          w[34] := w[34] + t;
279        end;
280      end;
281      if w[34] > 9 then
282        begin
283          t := w[34] div 10;
284          w[34] := w[34] mod 10;
285          w[35] := w[35] + t;
286        end;
287      end;
288      if w[35] > 9 then
289        begin
290          t := w[35] div 10;
291          w[35] := w[35] mod 10;
292          w[36] := w[36] + t;
293        end;
294      end;
295      if w[36] > 9 then
296        begin
297          t := w[36] div 10;
298          w[36] := w[36] mod 10;
299          w[37] := w[37] + t;
300        end;
301      end;
302      if w[37] > 9 then
303        begin
304          t := w[37] div 10;
305          w[37] := w[37] mod 10;
306          w[38] := w[38] + t;
307        end;
308      end;
309      if w[38] > 9 then
310        begin
311          t := w[38] div 10;
312          w[38] := w[38] mod 10;
313          w[39] := w[39] + t;
314        end;
315      end;
316      if w[39] > 9 then
317        begin
318          t := w[39] div 10;
319          w[39] := w[39] mod 10;
320          w[40] := w[40] + t;
321        end;
322      end;
323      if w[40] > 9 then
324        begin
325          t := w[40] div 10;
326          w[40] := w[40] mod 10;
327          w[41] := w[41] + t;
328        end;
329      end;
330      if w[41] > 9 then
331        begin
332          t := w[41] div 10;
333          w[41] := w[41] mod 10;
334          w[42] := w[42] + t;
335        end;
336      end;
337      if w[42] > 9 then
338        begin
339          t := w[42] div 10;
340          w[42] := w[42] mod 10;
341          w[43] := w[43] + t;
342        end;
343      end;
344      if w[43] > 9 then
345        begin
346          t := w[43] div 10;
347          w[43] := w[43] mod 10;
348          w[44] := w[44] + t;
349        end;
350      end;
351      if w[44] > 9 then
352        begin
353          t := w[44] div 10;
354          w[44] := w[44] mod 10;
355          w[45] := w[45] + t;
356        end;
357      end;
358      if w[45] > 9 then
359        begin
360          t := w[45] div 10;
361          w[45] := w[45] mod 10;
362          w[46] := w[46] + t;
363        end;
364      end;
365      if w[46] > 9 then
366        begin
367          t := w[46] div 10;
368          w[46] := w[46] mod 10;
369          w[47] := w[47] + t;
370        end;
371      end;
372      if w[47] > 9 then
373        begin
374          t := w[47] div 10;
375          w[47] := w[47] mod 10;
376          w[48] := w[48] + t;
377        end;
378      end;
379      if w[48] > 9 then
380        begin
381          t := w[48] div 10;
382          w[48] := w[48] mod 10;
383          w[49] := w[49] + t;
384        end;
385      end;
386      if w[49] > 9 then
387        begin
388          t := w[49] div 10;
389          w[49] := w[49] mod 10;
390          w[50] := w[50] + t;
391        end;
392      end;
393      if w[50] > 9 then
394        begin
395          t := w[50] div 10;
396          w[50] := w[50] mod 10;
397          w[51] := w[51] + t;
398        end;
399      end;
400      if w[51] > 9 then
401        begin
402          t := w[51] div 10;
403          w[51] := w[51] mod 10;
404          w[52] := w[52] + t;
405        end;
406      end;
407      if w[52] > 9 then
408        begin
409          t := w[52] div 10;
410          w[52] := w[52] mod 10;
411          w[53] := w[53] + t;
412        end;
413      end;
414      if w[53] > 9 then
415        begin
416          t := w[53] div 10;
417          w[53] := w[53] mod 10;
418          w[54] := w[54] + t;
419        end;
420      end;
421      if w[54] > 9 then
422        begin
423          t := w[54] div 10;
424          w[54] := w[54] mod 10;
425          w[55] := w[55] + t;
426        end;
427      end;
428      if w[55] > 9 then
429        begin
430          t := w[55] div 10;
431          w[55] := w[55] mod 10;
432          w[56] := w[56] + t;
433        end;
434      end;
435      if w[56] > 9 then
436        begin
437          t := w[56] div 10;
438          w[56] := w[56] mod 10;
439          w[57] := w[57] + t;
440        end;
441      end;
442      if w[57] > 9 then
443        begin
444          t := w[57] div 10;
445          w[57] := w[57] mod 10;
446          w[58] := w[58] + t;
447        end;
448      end;
449      if w[58] > 9 then
450        begin
451          t := w[58] div 10;
452          w[58] := w[58] mod 10;
453          w[59] := w[59] + t;
454        end;
455      end;
456      if w[59] > 9 then
457        begin
458          t := w[59] div 10;
459          w[59] := w[59] mod 10;
460          w[60] := w[60] + t;
461        end;
462      end;
463      if w[60] > 9 then
464        begin
465          t := w[60] div 10;
466          w[60] := w[60] mod 10;
467          w[61] := w[61] + t;
468        end;
469      end;
470      if w[61] > 9 then
471        begin
472          t := w[61] div 10;
473          w[61] := w[61] mod 10;
474          w[62] := w[62] + t;
475        end;
476      end;
477      if w[62] > 9 then
478        begin
479          t := w[62] div 10;
480          w[62] := w[62] mod 10;
481          w[63] := w[63] + t;
482        end;
483      end;
484      if w[63] > 9 then
485        begin
486          t := w[63] div 10;
487          w[63] := w[63] mod 10;
488          w[64] := w[64] + t;
489        end;
490      end;
491      if w[64] > 9 then
492        begin
493          t := w[64] div 10;
494          w[64] := w[64] mod 10;
495          w[65] := w[65] + t;
496        end;
497      end;
498      if w[65] > 9 then
499        begin
500          t := w[65] div 10;
501          w[65] := w[65] mod 10;
502          w[66] := w[66] + t;
503        end;
504      end;
505      if w[66] > 9 then
506        begin
507          t := w[66] div 10;
508          w[66] := w[66] mod 10;
509          w[67] := w[67] + t;
510        end;
511      end;
512      if w[67] > 9 then
513        begin
514          t := w[67] div 10;
515          w[67] := w[67] mod 10;
516          w[68] := w[68] + t;
517        end;
518      end;
519      if w[68] > 9 then
520        begin
521          t := w[68] div 10;
522          w[68] := w[68] mod 10;
523          w[69] := w[69] + t;
524        end;
525      end;
526      if w[69] > 9 then
527        begin
528          t := w[69] div 10;
529          w[69] := w[69] mod 10;
530          w[70] := w[70] + t;
531        end;
532      end;
533      if w[70] > 9 then
534        begin
535          t := w[70] div 10;
536          w[70] := w[70] mod 10;
537          w[71] := w[71] + t;
538        end;
539      end;
540      if w[71] > 9 then
541        begin
542          t := w[71] div 10;
543          w[71] := w[71] mod 10;
544          w[72] := w[72] + t;
545        end;
546      end;
547      if w[72] > 9 then
548        begin
549          t := w[72] div 10;
550          w[72] := w[72] mod 10;
551          w[73] := w[73] + t;
552        end;
553      end;
554      if w[73] > 9 then
555        begin
556          t := w[73] div 10;
557          w[73] := w[73] mod 10;
558          w[74] := w[74] + t;
559        end;
560      end;
561      if w[74] > 9 then
562        begin
563          t := w[74] div 10;
564          w[74] := w[74] mod 10;
565          w[75] := w[75] + t;
566        end;
567      end;
568      if w[75] > 9 then
569        begin
570          t := w[75] div 10;
571          w[75] := w[75] mod 10;
572          w[76] := w[76] + t;
573        end;
574      end;
575      if w[76] > 9 then
576        begin
577          t := w[76] div 10;
578          w[76] := w[76] mod 10;
579          w[77] := w[77] + t;
580        end;
581      end;
582      if w[77] > 9 then
583        begin
584          t := w[77] div 10;
585          w[77] := w[77] mod 10;
586          w[78] := w[78] + t;
587        end;
588      end;
589      if w[78] > 9 then
590        begin
591          t := w[78] div 10;
592          w[78] := w[78] mod 10;
593          w[79] := w[79] + t;
594        end;
595      end;
596      if w[79] > 9 then
597        begin
598          t := w[79] div 10;
599          w[79] := w[79] mod 10;
600          w[80] := w[80] + t;
601        end;
602      end;
603      if w[80] > 9 then
604        begin
605          t := w[80] div 10;
606          w[80] := w[80] mod 10;
607          w[81] := w[81] + t;
608        end;
609      end;
610      if w[81] > 9 then
611        begin
612          t := w[81] div 10;
613          w[81] := w[81] mod 10;
614          w[82] := w[82] + t;
615        end;
616      end;
617      if w[82] > 9 then
618        begin
619          t := w[82] div 10;
620          w[82] := w[82] mod 10;
621          w[83] := w[83] + t;
622        end;
623      end;
624      if w[83] > 9 then
625        begin
626          t := w[83] div 10;
627          w[83] := w[83] mod 10;
628          w[84] := w[84] + t;
629        end;
630      end;
631      if w[84] > 9 then
632        begin
633          t := w[84] div 10;
634          w[84] := w[84] mod 10;
635          w[85] := w[85] + t;
636        end;
637      end;
638      if w[85] > 9 then
639        begin
640          t := w[85] div 10;
641          w[85] := w[85] mod 10;
642          w[86] := w[86] + t;
643        end;
644      end;
645      if w[86] > 9 then
646        begin
647          t := w[86] div 10;
648          w[86] := w[86] mod 10;
649          w[87] := w[87] + t;
650        end;
651      end;
652      if w[87] > 9 then
653        begin
654          t := w[87] div 10;
655          w[87] := w[87] mod 10;
656          w[88] := w[88] + t;
657        end;
658      end;
659      if w[88] > 9 then
660        begin
661          t := w[88] div 10;
662          w[88] := w[88] mod 10;
663          w[89] := w[89] + t;
664        end;
665      end;
666      if w[89] > 9 then
667        begin
668          t := w[89] div 10;
669          w[89] := w[89] mod 10;
670          w[90] := w[90] + t;
671        end;
672      end;
673      if w[90] > 9 then
674        begin
675          t := w[90] div 10;
676          w[90] := w[90] mod 10;
677          w[91] := w[91] + t;
678        end;
679      end;
680      if w[91] > 9 then
681        begin
682          t := w[91] div 10;
683          w[91] := w[91] mod 10;
684          w[92] := w[92] + t;
685        end;
686      end;
687      if w[92] > 9 then
688        begin
689          t := w[92] div 10;
690          w[92] := w[92] mod 10;
691          w[93] := w[93] + t;
692        end;
693      end;
694      if w[93] > 9 then
695        begin
696          t := w[93] div 10;
697          w[93] := w[93] mod 10;
698          w[94] := w[94] + t;
699        end;
700      end;
701      if w[94] > 9 then
702        begin
703          t := w[94] div 10;
704          w[94] := w[94] mod 10;
705          w[95] := w[95] + t;
706        end;
707      end;
708      if w[95] > 9 then
709        begin
710          t := w[95] div 10;
711          w[95] := w[95] mod 10;
712          w[96] := w[96] + t;
713        end;
714      end;
715      if w[96] > 9 then
716        begin
717          t := w[96] div 10;
718          w[96] := w[96] mod 10;
719          w[97] := w[97] + t;
720        end;
721      end;
722      if w[97] > 9 then
723        begin
724          t := w[97] div 10;
725          w[97] := w[97] mod 10;
726          w[98] := w[98] + t;
727        end;
728      end;
729      if w[98] > 9 then
730        begin
731          t := w[98] div 10;
732          w[98] := w[98] mod 10;
733          w[99] := w[99] + t;
734        end;
735      end;
736      if w[99] > 9 then
737        begin
738          t := w[99] div 10;
739          w[99] := w[99] mod 10;
740          w[100] := w[100] + t;
741        end;
742      end;
743      if w[100] > 9 then
744        begin
745          t := w[100] div 10;
746          w[100] := w[100] mod 10;
747          w[101] := w[101] + t;
748        end;
749      end;
750      if w[101] > 9 then
751        begin
752          t := w[101] div 10;
753          w[101] := w[101] mod 10;
754          w[102] := w[102] + t;
755        end;
756      end;
757      if w[102] > 9 then
758        begin
759          t := w[102] div 10;
760          w[102] := w[102] mod 10;
761          w[103] := w[103] + t;
762        end;
763      end;
764      if w[103] > 9 then
765        begin
766          t := w[103] div 10;
767          w[103] := w[103] mod 10;
768          w[104] := w[104] + t;
769        end;
770      end;
771      if w[104] > 9 then
772        begin
773          t := w[104] div 10;
774          w[104] := w[104] mod 10;
775          w[105] := w[105] + t;
776        end;
777      end;
778      if w[105] > 9 then
779        begin
780          t := w[105] div 10;
781          w[105] := w[105] mod 10;
782          w[106] := w[106] + t;
783        end;
784      end;
785      if w[106] > 9 then
786        begin
787          t := w[106] div 10;
788          w[106] := w[106] mod 10;
789          w[107] := w[107] + t;
790        end;
791      end;
792      if w[107] > 9 then
793        begin
794          t := w[107] div 10;
795          w[107] := w[107] mod 10;
796          w[108] := w[108] + t;
797        end;
798      end;
799      if w[108] > 9 then
800        begin
801          t := w[108] div 10;
802          w[108] := w[108] mod 10;
803          w[109] := w[109] + t;
804        end;
805      end;
806      if w[109] > 9 then
807        begin
808          t := w[109] div 10;
809          w[109] := w[109] mod 10;
810          w[110] := w[110] + t;
811        end;
812      end;
813      if w[110] > 9 then
814        begin
815          t := w[110] div 10;
816          w[110] := w[110] mod 10;
817          w[111] := w[111] + t;
818        end;
819      end;
820      if w[111] > 9 then
821        begin
822          t := w[111] div 10;
823          w[111] := w[111] mod 10;
824          w[112] := w[112] + t;
825        end;
826      end;
827      if w[112] > 9 then
828        begin
829          t := w[112] div 10;
830          w[112] := w[112] mod 10;
831          w[113] := w[113] + t;
832        end;
833      end;
834      if w[113] > 9 then
835        begin
836          t := w[113] div 10;
837          w[113] := w[113] mod 10;
838          w[114] := w[114] + t;
839        end;
840      end;
841      if w[114] > 9 then
842        begin
843          t := w[114] div 10;
844          w[114] := w[114] mod 10;
845          w[115] := w[115] + t;
846        end;
847      end;
848      if w[115] > 9 then
849        begin
850          t := w[115] div 10;
851          w[115] := w[115] mod 10;
852          w[116] := w[116] + t;
853        end;
854      end;
855      if w[116] > 9 then
856        begin
857          t := w[116] div 10;
858          w[116] := w[116] mod 10;
859          w[117] := w[117] + t;
860        end;
861      end;
862      if w[117] > 9 then
863        begin
864          t := w[117] div 10;
865          w[117] := w[117] mod 10;
866          w[118] := w[118] + t;
867        end;
868      end;
869      if w[118] > 9 then
870        begin
871          t := w[118] div 10;
872          w[118] := w[118] mod 10;
873          w[119] := w[119] + t;
874        end;
875      end;
876      if w[119] > 9 then
877        begin
878          t := w[119] div 10;
879          w[119] := w[119] mod 10;
880          w[120] := w[120] + t;
881        end;
882      end;
883      if w[120] > 9 then
884        begin
885          t := w[120] div 10;
886          w[120] := w[120] mod 10;
887          w[121] := w[121] + t;
888        end;
889      end;
890      if w[121] > 9 then
891        begin
892          t := w[121] div 10;
893          w[121] := w[121] mod 10;
894          w[122] := w[122] + t;
895        end;
896      end;
897      if w[122] > 9 then
898        begin
899          t := w[122] div 10;
900          w[122] := w[122] mod 10;
901          w[123] := w[123] + t;
902        end;
903      end;
904      if w[123] > 9 then
905        begin
906          t := w[123] div 10;
907          w[123] := w[123] mod 10;
908          w[124] := w[124] + t;
909        end;
910      end;
911      if w[124] > 9 then
912        begin
913          t := w[124] div 10;
914          w[124] := w[124] mod 10;
915          w[125] := w[125] + t;
916        end;
917      end;
918      if w[125] > 9 then
919        begin
920          t := w[125] div 10;
921          w[125] := w[125] mod 10;
922          w[126] := w[126] + t;
923        end;
924      end;
925      if w[126] > 9 then
926        begin
927          t := w[126] div 10;
928          w[126] := w[126] mod 10;
929          w[127] := w[127] + t;
930        end;
931      end;
932      if w[127] > 9 then
933        begin
934          t := w[127] div 10;
935          w[127] := w[127] mod 10;
936          w[128] := w[128] + t;
937        end;
938      end;
939      if w[128] > 9 then
940        begin
941          t := w[128] div 10;
942          w[128] := w[128] mod 10;
943          w[129] := w[129] + t;
944        end;
945      end;
946      if w[129] > 9 then
947        begin
948          t := w[129] div 10;
949          w[129] := w[129] mod 10;
950          w[130] := w[130] + t;
951        end;
952      end;
953      if w[130] > 9 then
954        begin
955          t := w[130] div 10;
956          w[130] := w[130] mod 10;
957          w[131] := w[131] + t;
958        end;
959      end;
960      if w[131] > 9 then
961        begin
962          t := w[131] div 10;
963          w[131] := w[131] mod 10;
964          w[132] := w[132] + t;
965        end;
966      end;
967      if w[132] > 9 then
968        begin
969          t := w[132] div 10;
970          w[132] := w[132] mod 10;
971          w[133] := w[133] + t;
972        end;
973      end;
974      if w[133] > 9 then
975        begin
976          t := w[133] div 10;
97
```

```

1 # задача
2
3 // Назад > Решить > Задачи > № 1089 > ...
4
5 // Время выполнения: 0.000
6 // Проверка: Правильный
7
8 // Установите значение для переменных
9 // в соответствии с условиями задачи
10
11 // Время выполнения: 0.000
12 // Проверка: Правильный
13
14 // Установите значение для переменных
15 // в соответствии с условиями задачи
16
17 // Время выполнения: 0.000
18 // Проверка: Правильный
19
20 // Установите значение для переменных
21 // в соответствии с условиями задачи
22
23 // Время выполнения: 0.000
24 // Проверка: Правильный
25
26 // Установите значение для переменных
27 // в соответствии с условиями задачи
28
29 // Время выполнения: 0.000
30 // Проверка: Правильный
31
32 // Установите значение для переменных
33
34 // Время выполнения: 0.000
35
36 // Установите значение для переменных
37
38 // Время выполнения: 0.000
39
40 // Установите значение для переменных
41
42 // Время выполнения: 0.000
43
44 // Установите значение для переменных
45
46 // Время выполнения: 0.000
47
48 // Установите значение для переменных
49
50 // Время выполнения: 0.000
51
52 // Установите значение для переменных
53
54 // Время выполнения: 0.000
55
56 // Установите значение для переменных
57
58 // Время выполнения: 0.000
59
60 // Установите значение для переменных
61
62 // Время выполнения: 0.000
63
64 // Установите значение для переменных
65
66 // Время выполнения: 0.000
67
68 // Установите значение для переменных
69
70 // Время выполнения: 0.000
71
72 // Установите значение для переменных
73
74 // Время выполнения: 0.000
75
76 // Установите значение для переменных
77
78 // Время выполнения: 0.000
79
80 // Установите значение для переменных
81
82 // Время выполнения: 0.000
83
84 // Установите значение для переменных
85
86 // Время выполнения: 0.000
87
88 // Установите значение для переменных
89
90 // Время выполнения: 0.000
91
92 // Установите значение для переменных
93
94 // Время выполнения: 0.000
95
96 // Установите значение для переменных
97
98 // Время выполнения: 0.000
99
100 // Установите значение для переменных
101
102 // Время выполнения: 0.000
103
104 // Установите значение для переменных
105
106 // Время выполнения: 0.000
107
108 // Установите значение для переменных
109
110 // Время выполнения: 0.000
111
112 // Установите значение для переменных
113
114 // Время выполнения: 0.000
115
116 // Установите значение для переменных
117
118 // Время выполнения: 0.000
119
120 // Установите значение для переменных
121
122 // Время выполнения: 0.000
123
124 // Установите значение для переменных
125
126 // Время выполнения: 0.000
127
128 // Установите значение для переменных
129
130 // Время выполнения: 0.000
131
132 // Установите значение для переменных
133
134 // Время выполнения: 0.000
135
136 // Установите значение для переменных
137
138 // Время выполнения: 0.000
139
140 // Установите значение для переменных
141
142 // Время выполнения: 0.000
143
144 // Установите значение для переменных
145
146 // Время выполнения: 0.000
147
148 // Установите значение для переменных
149
150 // Время выполнения: 0.000
151
152 // Установите значение для переменных
153
154 // Установите значение для переменных
155
156 // Установите значение для переменных
157
158 // Установите значение для переменных
159
160 // Установите значение для переменных
161
162 // Установите значение для переменных
163
164 // Установите значение для переменных
165
166 // Установите значение для переменных
167
168
169
170 // Установите значение для переменных

```

```
C:\Users\Asus\Downloads\Лабы\Лаба 9 -  
128 function div big(u:Vector<int>, v:Vector<int>, b:int)  
129 {  
130     int res = 0;  
131     int r = length(v);  
132     if (r >= 1) {  
133         res = 0;  
134         for (int i = 0; i < r; i++) {  
135             res = res * 10 + v[i];  
136         }  
137     }  
138     return res;  
139 }  
140  
141 printLn("Лабораторная работа №9. Деление многочленов по точности")  
142 printLn("1 - Ввод коэффициентов")  
143 printLn("2 - Вычисление")  
144 printLn("3 - Умножение")  
145 printLn("4 - Деление (частное и остаток)")  
146 printLn("5 - Деление с остатком")  
147 printLn("6 - Выход")  
148  
149 // Бессструктурный цикл - программа работает, пока пользователь не выберет выход  
150 while true  
151 {  
152     printLn("Выберите действие (0 для выхода): ")  
153     input = stringParse(input);  
154     input = input[0] == '0' ? "" : input[0] == '6' ? "6" : input;  
155     if (input == "") break;  
156  
157     printLn("Введено значение n, если ошибки - сообщение и продолжим цикл");  
158     alg = try parseInt(input);  
159     if (alg < 0 || alg > 6) {  
160         printLn("Неверное значение алгоритма");  
161         continue;  
162     }  
163     if ((1 <= alg <= 5)) {  
164         printLn("Введено значение от 1 до 5");  
165         continue;  
166     }  
167  
168     printLn("Введено значение b (2-35): ");  
169     b = try parseInt(readline());  
170     if (b < 0 || b > 35) {  
171         printLn("Неверное значение b");  
172         continue;  
173     }  
174     if ((0 < b < 10)) {  
175         printLn("Введенное значение b недопустимо");  
176         continue;  
177     }  
178  
179     printLn("Введено число (в системе 10): ");  
180     x1 = readline();  
181     printLn("Введено число (в системе 10): ");  
182     x2 = readline();  
183  
184     if (x1 == "" || x2 == "") {  
185         printLn("Одно из чисел пусто");  
186         continue;  
187     }  
188  
189     printLn("Введено число (в системе 10): ");  
190     y1 = readline();  
191     printLn("Введено число (в системе 10): ");  
192     y2 = readline();  
193  
194     if (y1 == "" || y2 == "") {  
195         printLn("Одно из чисел пусто");  
196         continue;  
197     }  
198  
199     if (alg == 1) {  
200         printLn("Результат деления: " + div(x1, y1, b));  
201     }  
202     else if (alg == 2) {  
203         printLn("Результат умножения: " + mult(x1, y1, b));  
204     }  
205     else if (alg == 3) {  
206         printLn("Результат деления: " + div(x1, y1, b, 1));  
207     }  
208     else if (alg == 4) {  
209         printLn("Результат деления: " + div(x1, y1, b, 2));  
210     }  
211     else if (alg == 5) {  
212         printLn("Результат деления с остатком: " + div(x1, y1, b, 3));  
213     }  
214 }
```

```
C:\> Users> 4ek4o> Downloads > ▾ Lab05gj > ...
213     try
214         # Преобразуем строки в массивы цифр
215         s = str_to_digits(str1, v)
216         r = str_to_digits(str2, b)
217
218         # Выполним выбранный алгоритм
219         if alg == 1:
220             add(big(u, v, b))
221             printin("Сумма: ${digits_to_str(res, b)}")n
222         elif alg == 2:
223             # Проверим условия u <= v для вычитания
224             if length(u) < length(v) || (length(u) == length(v) && u < v):
225                 printin("Ошибка: первое число должно быть > второго!")n
226             else:
227                 res = sub(big(u, v, b))
228                 printin("Разность: ${digits_to_str(res, b)}")n
229             end
230         else:
231             alg == 3
232             res = mul(classic(u, v, b))
233             printin("Произведение (столбиком): ${digits_to_str(res, b)}")n
234             alg == 4
235             res = fast(u, v, b)
236             printin("Произведение (быстро): ${digits_to_str(res, b)}")n
237             alg == 5
238             q, r = div(big(u, v, b))
239             printin("Частное: ${digits_to_str(q, b)}")
240             printin("Остаток: ${digits_to_str(r, b)}")n
241         end
242     catch e:
243         # Ловим все ошибки входа (недопустимые цифры и т.д.)
244         printin("Ошибка: ${print(showererr, e)}")n
245     end
246 end
```

Лабораторная работа №8: Арифметика многократной точности  
0 – Выход  
1 – Сложение  
2 – Вычитание  
3 – Умножение быстрым столбиком  
4 – Умножение быстрым столбиком  
5 – Деление с остатком

**julia> 1**

Основание b (2-36): 6  
Первое число (в системе 6): 4  
Второе число (в системе 6): 2  
Сумма: 10

Выберите алгоритм (0 для выхода): 2

Основание b (2-36): 29  
Первое число (в системе 29): 23  
Второе число (в системе 29): 17  
Ошибка: первое число должно быть ≥ второго

Выберите алгоритм (0 для выхода): 2  
Основание b (2-36): 30  
Первое число (в системе 30): 29  
Второе число (в системе 30): 10  
Разность: 19

Выберите алгоритм (0 для выхода): 3  
Основание b (2-36): 33  
Первое число (в системе 33): 32  
Второе число (в системе 33): 19  
Произведение (столбиком): 371

Выберите алгоритм (0 для выхода): 4  
Основание b (2-36): 10  
Первое число (в системе 10): 2  
Второе число (в системе 10): 8  
Произведение (быстро): 16

Выберите алгоритм (0 для выхода): 4  
Основание b (2-36): 16  
Первое число (в системе 16): 15  
Второе число (в системе 16): 12  
Частное: 0  
Остаток: 15

Выберите алгоритм (0 для выхода): 5  
Основание b (2-36): 16

Первое число (в системе 16): 15

Второе число (в системе 16): 12

Частное: 0

Остаток: 15

Выберите алгоритм (0 для выхода): 0

До свидания!