Целочисленная арифметика многократной точности

Лабораторная работа №8

Данилова А.С.

Цели и задачи

- Изучить теоретическую часть о предложенных алгоритмах;
- Реализовать алгоритмы программно.

Целочисленная арифметика многократной точности

Целочисленная арифметика многократной точности — это алгоритмы для выполнения арифметических операций с большими целыми числами.

Примеры алгоритмов

1. Сложение неотрицательных целых чисел
2. Вычитание неотрицательных целых чисел
3. Умножение неотрицательных целых чисел столбиком
4. Быстрый столбик
5. Деление многоразрядных целых чисел

Выполнение работы

```julia
function plus(a::Vector{Int}, b::Vector{Int})
    result = Int[]
    carry = 0
    max_length = max(length(a), length(b))

    for i in 1:max_length
        digit_a = i <= length(a) ? a[end - i + 1] : 0
        digit_b = i <= length(b) ? b[end - i + 1] : 0
        sum = digit_a + digit_b + carry
        push!(result, sum % 10)
        carry = div(sum, 10)
    end

    if carry > 0
        push!(result, carry)
    end

    return reverse(result)
end
a = [1, 2, 3, 5, 4, 2]
b = [4, 5, 6, 4, 2, 7]
sum_res = plus(a, b)
println("сумма: ", sum_res)
```

Рис. 1: Сложение неотрицательных целых чисел

Полученный результат

```
  Activating new project at `C:\Users\nastd\.julia\environments\v1.11`
Сумма: [5, 7, 9]
  *  Terminal will be reused by tasks, press any key to close it.

  Activating new project at `C:\Users\nastd\.julia\environments\v1.11`
Сумма: [5, 7, 9, 9, 6, 9]
  *  Terminal will be reused by tasks, press any key to close it.
```

Рис. 2: Результат

Выполнение работы

```julia
function minus(a::Vector{Int}, b::Vector{Int})
    result = Int[]
    borrow = 0

    for i in 1:length(a)
        digit_a = a[end - i + 1]
        digit_b = i <= length(b) ? b[end - i + 1] : 0
        diff = digit_a - digit_b - borrow

        if diff < 0
            diff += 10
            borrow = 1
        else
            borrow = 0
        end

        push!(result, diff)
    end

    while length(result) > 1 && result[end] == 0
        pop!(result)
    end

    return reverse(result)
end
b = [1, 2, 3, 5, 4, 2]
a = [4, 5, 6, 6, 5, 7]
min_res = minus(a, b)
println("Разность: ", min_res)
```

Рис. 3: Вычитание неотрицательных целых чисел

Полученный результат

```
  Activating new project at `C:\Users\nastd\.julia\environments\v1.11`
Разность: [3, 3, 3, 1, 1, 5]
  *  Terminal will be reused by tasks, press any key to close it.
```

Рис. 4: Результат

Выполнение работы

```julia
function multiply(a::Vector{Int}, b::Vector{Int})
    result = zeros(Int, length(a) + length(b))

    for i in 1:length(b)
        carry = 0
        for j in 1:length(a)
            product = b[end - i + 1] * a[end - j + 1] + result[end - (i + j - 2)] + carry
            result[end - (i + j - 2)] = product % 10
            carry = div(product, 10)
        end
        result[end - (i + length(a) - 1)] += carry
    end

    while length(result) > 1 && result[1] == 0
        result = result[2:end]
    end

    return result
end
b = [1, 2, 3]
a = [4, 5, 6]
mul_res = multiply(a, b)
println("Произведение: ", mul_res)
```

Рис. 5: Умножение неотрицательных целых чисел столбиком

Полученный результат

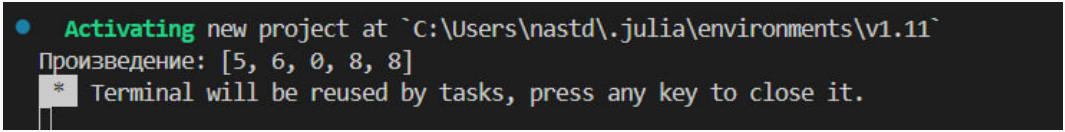

Рис. 6: Результат

Выполнение работы

```julia
function multiply(a::String, b::String)::String
    digit_a = reverse(parse.(Int, collect(a)))
    digit_b = reverse(parse.(Int, collect(b)))

    m = length(digit_a)
    n = length(digit_b)

    result = zeros(Int64, m + n)

    for i in 1:m
        for j in 1:n
            result[i + j - 1] += digit_a[i] * digit_b[j]
        end
    end

    carry = 0
    for k in 1:length(result)
        result[k] += carry
        carry = div(result[k], 10)
        result[k] %= 10
    end

    while length(result) > 1 && result[end] == 0
        pop!(result)
    end

    return join(reverse(result))
end

a = "1234"
b = "98765"
result = multiply(a, b)
println("Результат: ", result)
```

Рис. 7: Быстрый столбик

Полученный результат

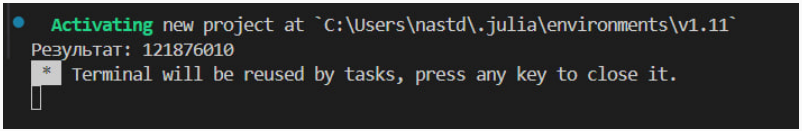

Рис. 8: Результат

Выводы

Мы изучили 5 алгоритмов целочисленной арифметики многократной точности и реализовали их программно на языке программмирования Julia.