Лабораторная работа №8

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

21 декабря 2024

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Прагматика выполнения

Прагматика выполнения

- Освоение алгоритмов алгоритмов целочисленной арифметики многократной точности — сложение неотрицательных целых чисел, вычитание неотрицательных целых чисел, умножение неотрицательных целых чисел столбиком, быстрый столбик, деление многоразрядных целых чисел.

Цели

Цели

Изучить работу алгоритмов целочисленной арифметики многократной точности: сложение неотрицательных целых чисел; вычитание неотрицательных целых чисел; умножение неотрицательных целых чисел столбиком; быстрый столбик; деление многоразрядных целых чисел; а также реализовать их программно.

Задачи

Задачи

1. Реализовать рассмотренные алгоритмы программно на языке Julia.

Выполнение работы

Алгоритмы сложения и вычитания многоразрядных неотрицательных целых чисел

```julia
"""Алгоритм 1. Сложение неотрицательных целых чисел"""
function Add_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    # u, v - складываемые числа, записаны в виде массивов цифр, b - основание системы счисления
    n = length(u)
    m = length(v)
    w = zeros(Int, n)
    k = 0
    j = n
    while j > 0
        temp = u[j] + v[j] + k
        w[j] = mod(temp, b)
        k = div(temp, b)
        j -= 1
    end
    w[1] += k
    return w
end

"""Алгоритм 2. Вычитание неотрицательных целых чисел"""
function Subtract_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    # u, v - вычитаемые числа, записаны в виде массивов цифр, b - основание системы счисления
    n = length(u)
    w = zeros(Int, n)
    k = 0
    j = n
    while j > 0
        temp = u[j] - v[j] + k
        w[j] = mod(temp, b)
        k = div(temp, b)
        j -= 1
    end
    w[1] += k
    return w
end
```

Алгоритм умножения многоразрядных неотрицательных целых чисел столбиком

```julia
"""Алгоритм 3. Умножение неотрицательных целых чисел столбиком"""
function Multiply_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)
    j = m
    while j > 0
        if v[j] == 0
            w[j] = 0
            j -= 1
        else
            i = n
            k = 0
            while i > 0
                temp = w[i+j] + u[i] * v[j] + k
                w[i+j] = mod(temp, b)
                k = div(temp, b)
                i -= 1
            end
            w[j] = k
            j -= 1
        end
    end
    return w
end
```

Рис. 2: Код алгоритма умножения неотрицательных целых чисел столбиком на Julia

Умножение многоразрядных неотрицательных целых чисел алгоритмом быстрого столбика

```julia
function Fast_Multiply_Large_Positive_Numbers(u::Vector{Int}, v::Vector{Int}, b::Int)::Vector{Int}
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)  # Результат будет длиной n + m
    t = 0  # Переменная для переноса

    for s in 1:(n + m)  # Итерации по разрядам результата
        for i in max(1, s - m + 1):min(n, s)
            t += u[n - i + 1] * v[m - (s - i)]
        end
        w[n + m - s + 1] = mod(t, b)  # Записываем младший разряд
        t = div(t, b)  # Переносим в старший разряд
    end

    w[1] += t  # Добавляем оставшийся перенос, если он есть
    return w
end
```

Рис. 3: Код умножения неотрицательных целых чисел алгоритмом быстрого столбика на Julia

Алгоритм деления многоразрядных неотрицательных целых чисел

```
function Divide_Large_Positive_Numbers(u::Int, v::Int, b::Int)
    # Преобразуем числа в массивы цифр
    u_digits = reverse(digits(u, base=b))
    v_digits = reverse(digits(v, base=b))

    # Инициализация переменных
    n = length(u_digits)
    t = length(v_digits)
    q = zeros(Int, max(0, n - t + 1)) # Вычисление размера массива q
    r = 0

    # Основной цикл деления
    for i in 1:n
        # Обновляем остаток, добавляя очередную цифру u_digits
        r = r * b + u_digits[i]

        # Определяем текущую цифру частного
        if r >= v
            if i - t + 1 > 0
                q[i - t + 1] = div(r, v)
            else
                q[i - t + 1] = 0 # Избегаем отрицательных индексов
            end
            r %= v
        else
            if i - t + 1 > 0
                q[i - t + 1] = 0
            end
        end
    end
```

Начальные данные (1/2)

```
# Пример использования 1
b = 10  # Основание системы счисления

# Пример чисел для тестирования
u = [6, 9, 5, 7]  # Число u = 6957
v = [1, 4, 2, 3]  # Число v = 1423
u1, v1 = 6957, 1423

println("Число u: ", u, "\t Число v:", v)

# Алгоритм 1: Сложение
sum_result = Add_Large_Positive_Numbers(u, v, b)
println("Сумма: ", sum_result)

# Алгоритм 2: Вычитание
subtract_result = Subtract_Large_Positive_Numbers(u, v, b)
println("Разность: ", subtract_result)

# Алгоритм 3: Умножение
multiply_result = Multiply_Large_Positive_Numbers(u, v, b)
println("Произведение: ", multiply_result)

# Алгоритм 4: Быстрое умножение
fast_multiply_result = Fast_Multiply_Large_Positive_Numbers(u, v, b)
println("Быстрый столбик: ", fast_multiply_result)

# Алгоритм 5: Деление
quotient, remainder = Divide_Large_Positive_Numbers(u1, v1, b)
println("Частное: ", quotient)
println("Остаток: ", remainder)
```

Начальные данные (2/2)

```
297  # Пример использования 2
298  b = 10  # Основание системы счисления
299
300  # Пример чисел для тестирования
301  u = [6, 9, 5, 7, 1, 4, 2]  # Число u = 6957142
302  v = [1, 4, 2, 3]  # Число v = 1423
303
304  println("\nЧисло u: ", u, "\t Число v:", v)
305
306  # Алгоритм 3: Умножение
307  multiply_result = Multiply_Large_Positive_Numbers(u, v, b)
308  println("Произведение: ", multiply_result)
309
310  # Алгоритм 4: Быстрое умножение
311  fast_multiply_result = Fast_Multiply_Large_Positive_Numbers(u, v, b)
312  println("Быстрый столбик: ", fast_multiply_result)
313
314  # Алгоритм 5: Деление
315  u2, v2 = 6957142, 1423
316  #quotient, remainder = Divide_Large_Positive_Numbers(u, v, b)
317  quotient, remainder = Divide_Large_Positive_Numbers(u2, v2, b)
318  println("Частное: ", quotient)
319  println("Остаток: ", remainder)
```

Результат работы алгоритмов арифметических операций с многоразрядными целыми числами

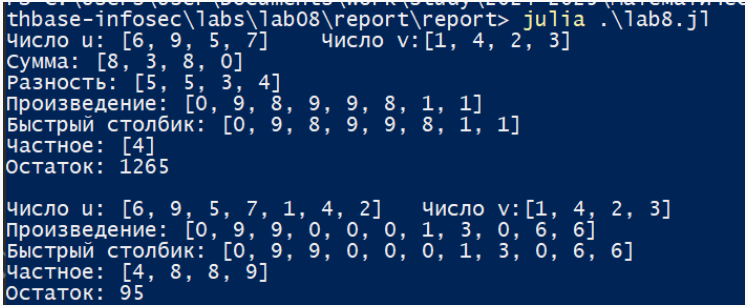

Рис. 7: Результат выполнения кода алгоритмов арифметических операций с многоразрядными целыми числами на Julia

Результаты

Результаты

По результатам работы, я изучил работу алгоритмов целочисленной арифметики многократной точности: сложение неотрицательных целых чисел; вычитание неотрицательных целых чисел; умножение неотрицательных целых чисел столбиком; быстрый столбик; деление многоразрядных целых чисел; а также реализовал их программно.