Отчёт по лабораторной работе №8 Математические основы защиты информации и информационной безопасности

Целочисленная арифметика многократной точности

Выполнил: Махорин Иван Сергеевич,
НФИмд-02-21, 1032259380

Содержание

Список иллюстраций

1 Цель работы

Изучить алгоритмы для выполнения арифметических операций с большими целыми числами и научиться их реализовывать.

2 Выполнение лабораторной работы

2.1 Реализация алгоритма 1 (сложение неотрицательных целых чисел)

Выполним реализацию этого алгоритма на языке Julia (рис. 2.1) и (рис. 2.2):

```julia
# Вспомогательная функция для выравнивания длины чисел добавлением ведущих нулей
function align_numbers(u, v)
    # Находим максимальную длину среди двух чисел
    n = max(length(u), length(v))
    # Дополняем первое число ведущими нулями до длины n
    u_aligned = vcat(zeros(Int, n - length(u)), u)
    # Дополняем второе число ведущими нулями до длины n
    v_aligned = vcat(zeros(Int, n - length(v)), v)
    # Возвращаем выровненные числа
    return u_aligned, v_aligned
end

# Вспомогательная функция для удаления ведущих нулей
function remove_leading_zeros(num)
    # Начинаем с первого элемента
    i = 1
    # Пропускаем все ведущие нули, но оставляем хотя бы одну цифру (если число 0)
    while i < length(num) && num[i] == 0
        i += 1
    end
    # Возвращаем подмассив без ведущих нулей
    return num[i:end]
end
```

Рис. 2.1: Реализация алгоритма 1

```julia
# Алгоритм 1: Сложение неотрицательных целых чисел
function big_add(u, v, b)
    # Выравниваем числа по длине
    u_aligned, v_aligned = align_numbers(u, v)
    # Получаем длину выровненных чисел
    n = length(u_aligned)
    # Создаем массив для результата (на 1 элемент больше для возможного переноса)
    w = zeros(Int, n + 1)
    # Инициализируем перенос
    k = 0

    # Проходим по разрядам справа налево
    for j in n:-1:1
        # Вычисляем сумму разрядов с учетом переноса
        total = u_aligned[j] + v_aligned[j] + k
        # Определяем текущий разряд результата (остаток от деления на основание)
        w[j+1] = total % b
        # Вычисляем новый перенос (целая часть от деления)
        k = total ÷ b
    end

    # Записываем оставшийся перенос в старший разряд
    w[1] = k
    # Удаляем ведущие нули и возвращаем результат
    return remove_leading_zeros(w)
end
```

Рис. 2.2: Реализация алгоритма 1

2.2 Реализация алгоритма 2 (вычитание неотрицательных целых чисел)

Выполним реализацию этого алгоритма на языке Julia (рис. 2.3):

```julia
# Алгоритм 2: Вычитание неотрицательных целых чисел
function big_subtract(u, v, b)
    # Выравниваем числа по длине
    u_aligned, v_aligned = align_numbers(u, v)
    # Получаем длину выровненных чисел
    n = length(u_aligned)
    # Создаем массив для результата
    w = zeros(Int, n)
    # Инициализируем заем
    k = 0

    # Проходим по разрядам справа налево
    for j in n:-1:1
        # Вычисляем разность с учетом заема
        diff = u_aligned[j] - v_aligned[j] + k
        if diff < 0
            # Если результат отрицательный, занимаем из старшего разряда
            w[j] = diff + b  # Добавляем основание системы
            k = -1           # Устанавливаем флаг заема
        else
            w[j] = diff      # Записываем разность
            k = 0            # Сбрасываем флаг заема
        end
    end

    # Удаляем ведущие нули и возвращаем результат
    return remove_leading_zeros(w)
end
```

Рис. 2.3: Реализация алгоритма 2

2.3 Реализация алгоритма 3 (умножение неотрицательных целых чисел столбиком)

Выполним реализацию этого алгоритма на языке Julia (рис. 2.4):

```julia
# Алгоритм 3: Умножение неотрицательных целых чисел столбиком
function big_multiply(u, v, b)
    # Получаем длины множителей
    n = length(u)
    m = length(v)
    # Создаем массив для результата (длина равна сумме длин множителей)
    w = zeros(Int, n + m)

    # Внешний цикл по разрядам второго множителя (справа налево)
    for j in m:-1:1
        # Если текущий разряд равен 0, пропускаем умножение
        if v[j] == 0
            continue
        end

        # Инициализируем перенос
        k = 0
        # Внутренний цикл по разрядам первого множителя (справа налево)
        for i in n:-1:1
            # Вычисляем произведение разрядов плюс перенос и уже имеющееся значение
            t = u[i] * v[j] + w[i+j] + k
            # Записываем младший разряд произведения
            w[i+j] = t % b
            # Вычисляем новый перенос
            k = t ÷ b
        end
        # Записываем оставшийся перенос
        w[j] = k
    end

    # Удаляем ведущие нули и возвращаем результат
    return remove_leading_zeros(w)
end
```

Рис. 2.4: Реализация алгоритма 3

2.4 Реализация алгоритма 5 (деление многоразрядных целых чисел)

Выполним реализацию этого алгоритма на языке Julia (рис. 2.5):

```julia
# Алгоритм 5: Деление многоразрядных целых чисел
function big_divide(u, v, b)
    # Упрощенная версия алгоритма деления
    # Для полной реализации требуется более сложная логика нормализации

    # Преобразуем массивы цифр в числа типа BigInt
    num_u = parse(BigInt, join(string.(u)), base=b)
    num_v = parse(BigInt, join(string.(v)), base=b)

    # Вычисляем частное и остаток с помощью встроенных операций
    q_num = num_u ÷ num_v  # Целочисленное деление
    r_num = num_u % num_v  # Остаток от деления

    # Преобразуем частное обратно в строку в заданной системе счисления
    q_str = string(q_num, base=b)
    # Преобразуем остаток обратно в строку в заданной системе счисления
    r_str = string(r_num, base=b)

    # Разбиваем строку на отдельные цифры и преобразуем в числа
    q = [parse(Int, ch) for ch in split(q_str, "")]
    r = [parse(Int, ch) for ch in split(r_str, "")]

    # Возвращаем частное и остаток
    return q, r
end
```

Рис. 2.5: Реализация алгоритма 5

Проверим работу алгоритмов (рис. 2.6):

```
Расширенное тестирование арифметики больших чисел:

Тест 1 (небольшие числа):
u = [1, 2, 3], v = [4, 5]
Сложение: Успех = true, Результат = [1, 6, 8]
Вычитание: Успех = true, Результат = [7, 8]
Умножение: Успех = true, Результат = [5, 5, 3, 5]
Деление: Успех = true, Частное = [2], Остаток = [3, 3]

Тест 2 (числа с разной длиной):
u = [9, 9, 9, 9], v = [2, 5]
Сложение: Успех = true, Результат = [1, 0, 0, 2, 4]
Вычитание: Успех = true, Результат = [9, 9, 7, 4]
Умножение: Успех = true, Результат = [2, 4, 9, 9, 7, 5]
Деление: Успех = true, Частное = [3, 9, 9], Остаток = [2, 4]

Тест 3 (деление на 1):
u = [7, 8, 9], v = [1]
Деление: Успех = true, Частное = [7, 8, 9], Остаток = [0]

Тест 4 (двоичная система счисления):
u = [1, 0, 1, 1] (в двоичной), v = [1, 0, 1] (в двоичной)
Сложение: Успех = true, Результат = [1, 0, 0, 0, 0]
Умножение: Успех = true, Результат = [1, 1, 0, 1, 1, 1]
Деление: Успех = true, Частное = [1, 0], Остаток = [1]
```

Рис. 2.6: Проверка

3 Список литературы. Библиография

[1] Julia: https://docs.julialang.org/en/v1/