Лабораторна работа № 8

Целочисленная арифметика многократной точности

Покрас Илья Михайлович

Содержание

Список иллюстраций

1 Цель работы

Реализовать алгоритмы целочисленной арифметики многократной точности

2 Задание

- Реализовать алгоритм сложения неотрицательных чисел
- Реализовать алгоритм вычитания неотрицательных чисел
- Реализовать алгоритм умножения неотрицательных чисел
- Реализовать алгоритм “быстрый столбик”
- Реализовать алгоритм деления многоразрядных целых чисел

3 Теоретическое введение

- Алгоритм сложения неотрицательных чисел заключается в пошаговом суммировании цифр чисел, начиная с младших разрядов и перенося разряды при необходимости. Этот алгоритм позволяет эффективно выполнять операцию сложения.
- Алгоритм вычитания неотрицательных чисел включает поэтапное вычитание цифр чисел, начиная с младших разрядов и заемом разрядов, если необходимо. Он обеспечивает эффективное выполнение операции вычитания.
- Алгоритм умножения неотрицательных чисел базируется на методе пошагового умножения цифр чисел и последующем сложении результатов. Он предоставляет эффективный способ выполнения операции умножения.
- Алгоритм “быстрый столбик” представляет собой метод многоразрядного умножения чисел с использованием оптимизаций. Этот алгоритм значительно ускоряет выполнение операции умножения.
- Алгоритм деления многоразрядных целых чисел основывается на методе пошагового нахождения частного и остатка при делении чисел. Он позволяет эффективно и точно выполнять операцию деления.

4 Выполнение лабораторной работы

4.1 Алгоритм сложения неотрицательных чисел

Я реализовал функцию алгоритма сложения неотрицательных чисел. Она принимает два массива цифр u и v, их длину n и основание системы счисления b, складывает числа, представленные массивами u и v, с учетом переносов и возвращает результат в виде нового массива цифр w. (рис. 4.1).

```
function alg_add(u, v, n, b)
    k = 0
    w = zeros(Int, n)
    j = n
    while j > 0
        k = div(u[j] + v[j] + k, b)
        w[j] = (u[j] + v[j] + k) % b
        j -= 1
    end
    return w
end
```

Рис. 4.1: Алгоритм сложения неотрицательных чисел

4.2 Алгоритм вычитания неотрицательных чисел

Я реализовал функцию алгоритма вычитания неотрицательных чисел, принцип которого схож с алгоритмом сложения - она также принимает два массива цифр u и v, их длину n и основание системы счисления b. Она вычитает число, представленное массивом v, из числа, представленного массивом u, с учетом

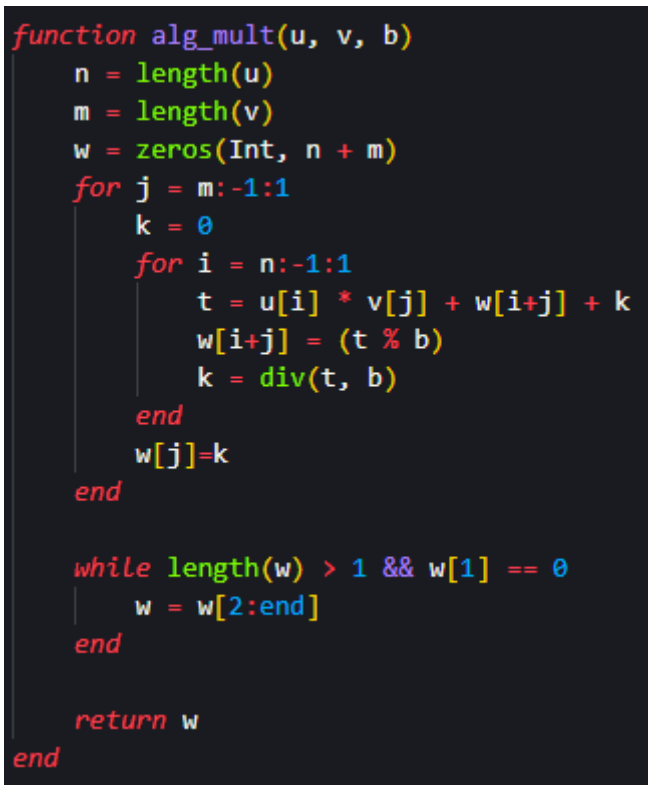
переносов и возвращает результат в виде нового массива цифр w (рис. 4.2).

```
function alg_sub(u, v, n, b)
    k = 0
    w = zeros(Int, n)
    j = n
    while j > 0
        k = div(u[j] - v[j] + k, b)
        w[j] = (u[j] - v[j] + k) % b
        j -= 1
    end
    return w
end
```

Рис. 4.2: Алгоритм вычитания неотрицательных чисел

4.3 Алгоритм умножения неотрицательных чисел

Далее я реализовал функцию умножения неотрицательных чисел реализует умножение u и v с основанием системы счисления b. В циклах происходит умножение коэффициентов, вычисление остатка от деления и переноса, а также добавление переноса к более старшему разряду. После умножения удаляются ведущие нули и возвращается результат(рис. 4.3).



```
function alg_mult(u, v, b)
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)
    for j = m:-1:1
        k = 0
        for i = n:-1:1
            t = u[i] * v[j] + w[i+j] + k
            w[i+j] = (t % b)
            k = div(t, b)
        end
        w[j]=k
    end

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

    return w
end
```

Рис. 4.3: Алгоритм умножения неотрицательных чисел

4.4 Алгоритм “быстрый столбик”

Реализованная мной функция алгоритма “быстрый столбик” также умножает u и v с основанием системы счисления b, но использует более оптимизированный подход. Она выполняет те же операции, но использует вложенные циклы для суммирования произведений коэффициентов во время умножения. Результат также корректируется и возвращается после удаления ведущих нулей (рис. 4.4).

```
function alg_fast_mult(u, v, b)
    n = length(u)
    m = length(v)
    w = zeros(Int, n + m)
    t = 0
    for s = 0:(n + m - 1)
        for i = 0:s
            if (1 <= n - i <= n) && (1<=m-s+i<=m)
                t += u[n-i] * v[m-s+i]
            end
        end
        w[m + n - s] = mod(t, b)
        t = div(t, b)
    end

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

    return w
end
```

Рис. 4.4: Алгоритм “быстрый столбик”

4.5 Алгоритм деления многоразрядных целых чисел

Сама мною реализованная функция деления многоразрядных целых чисел принимает три параметра: два массива чисел u и v и число b. Внутри функции vec_convert используется для конвертации массивов чисел u и v в целочисленные значения. Затем в функции alg_div происходит деление u на v в b-ичной системе счисления. Деление выполняется сначала с помощью оператора ÷, затем находится остаток от деления с помощью оператора %. Далее результат деления d раскладывается на цифры и сохраняется в массив r1, а затем выполняются b итераций, при которых остаток m умножается на 10, далее находится целая часть от деления и добавляется в массив r2. В конце функция возвращает массивы r1 и r2 (рис. 4.5).

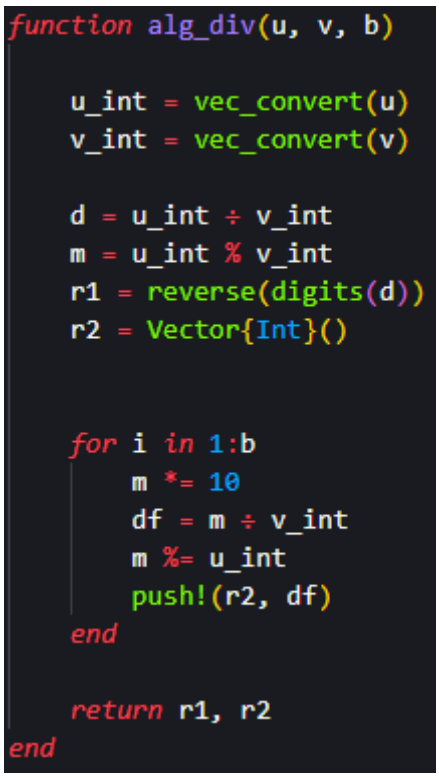


Рис. 4.5: Алгоритм деления многоразрядных целых чисел

Также Для данного алгоритма была реализована функция конвертации массива в переменную типа BigInt (рис. 4.6).

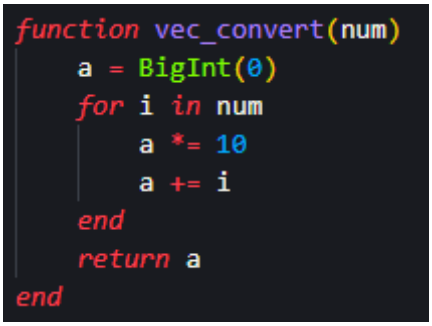


Рис. 4.6: Функция конвертации массивов

4.6 Инициализация переменных и вызов функции

Далее я инициализировал переменные, которые будут входными параметрами вызванных далее функций (рис. 4.7).

```
u = [4, 4, 4, 4]
v = [2, 2, 2, 2]
n = length(u)
b = 10

println(alg_add(u, v, n, b))
println(alg_sub(u, v, n, b))
println(alg_mult(u, v, b))
println(alg_fast_mult(u, v, b))
println(alg_div(u, v, b))
```

Рис. 4.7: Инициализация переменных и вызов функции

И получил следующей результат (рис. 4.8).

```
[6, 6, 6, 6]
[2, 2, 2, 2]
[9, 8, 7, 4, 5, 6, 8]
[9, 8, 7, 4, 5, 6, 8]
([2], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Рис. 4.8: Результат выполнения кода

5 Выводы

Я реализовал алгоритмы целочисленной арифметики многократной точности

Список Литературы

1. Julia - Control Flow
2. Julia - Mathematical Operations
3. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone - Handbook of Applied Cryptography