Лабораторная работа №8

Целочисленная арифметика многократной точности

Лебедева Ольга Андреевна

Содержание

Цель работы 4

Задачи 5

Объект и предмет исследования 6

Условные обозначения и термины 7

Техническое оснащение и выбранные методы проведения работы 8

Теоретическое введение 9

Задание 10

Сложение неотрицательных целых чисел 11

Вычитание неотрицательных целых чисел 13

Умножение многозначных целых чисел столбиком 15

Быстрое умножения многозначных целых чисел 16

Деление многозначных целых чисел с вычислением частного и остатка. 17

Полученные результаты 19

Заключение 21

Библиографическая справка 22

Список иллюстраций

Цель работы

Изучить принципы реализации арифметических операций над целыми числами многократной точности и реализовать на языке Julia[1] основные алгоритмы сложения, вычитания, умножения и деления больших чисел средствами языка программирования Julia.

Задачи

1. Ознакомиться с представлением больших целых чисел в виде массивов цифр.
2. Реализовать алгоритмы сложения и вычитания неотрицательных целых чисел.
3. Реализовать два алгоритма умножения многозначных чисел.
4. Реализовать алгоритм деления многозначных целых чисел с получением частного и остатка.
5. Проверить корректность работы реализованных алгоритмов на тестовых данных.

Объект и предмет исследования

Объект исследования: целочисленная арифметика многократной точности.

Предмет исследования: алгоритмы поразрядного выполнения арифметических операций над большими целыми числами и их программная реализация на языке Julia.

Условные обозначения и термины

Основание системы счисления b — натуральное число, определяющее количество допустимых цифр в каждом разряде.

Перенос — целая часть результата поразрядной операции, передаваемая в старший разряд.

Заём — операция вычитания, при которой значение старшего разряда уменьшается для корректного выполнения разности

Техническое оснащение и выбранные методы проведения работы

Программное обеспечение:

- Язык программирования Julia.
- Среда разработки JupyterLab / VS Code.

Методы:

- Представление чисел в виде массивов целых цифр.
- Поразрядная обработка чисел в циклах.
- Использование целочисленного деления и операции взятия остатка.
- Ручная обработка переносов и заимствований.

Теоретическое введение

Арифметика многократной точности применяется в задачах криптографии, численных методов и теории чисел, где стандартной разрядности целых типов данных недостаточно. Основной идеей является представление числа в виде последовательности цифр в некоторой системе счисления и выполнение всех операций поразрядно, аналогично классическим школьным алгоритмам.

Такой подход позволяет реализовать сложение, вычитание, умножение и деление чисел произвольной длины, не используя встроенные типы больших чисел.

Задание

Реализовать программно алгоритмы:

1. Сложения неотрицательных целых чисел.
2. Вычитания неотрицательных целых чисел.
3. Умножения многозначных целых чисел столбиком.
4. Быстрого умножения многозначных целых чисел.
5. Деления многозначных целых чисел с вычислением частного и остатка.

Сложение неотрицательных целых чисел

Выполним задание 1 с помощью языка Julia:

```
function normalize(x)
    i = findfirst(!=(0), x)
    i === nothing ? [0] : x[i:end]
end


function add_big(u, v, b)
    n = max(length(u), length(v))
    u = vcat(zeros(Int, n-length(u)), u)
    v = vcat(zeros(Int, n-length(v)), v)

    w = zeros(Int, n)
    carry = 0
    for j in n:-1:1
        s = u[j] + v[j] + carry
        w[j] = s % b
        carry = div(s, b)
    end
    carry == 0 ? w : vcat(carry, w)
end
```

В рамках работы реализованы алгоритмы сложения и вычитания, основанные на поразрядной обработке чисел. Сложение выполняется справа налево с учётом переноса

в старший разряд. Числа предварительно выравниваются по длине путём добавления ведущих нулей. После завершения операции результат нормализуется путём удаления ведущих нулей.

Вычитание неотрицательных целых чисел

Выполним задание 2 с помощью языка Julia:

```
function sub_big(u, v, b)
    n = length(u)
    v = vcat(zeros(Int, n-length(v)), v)

    w = zeros(Int, n)
    borrow = 0
    for j in n:-1:1
        t = u[j] - v[j] + borrow
        if t < 0
            t += b
            borrow = -1
        else
            borrow = 0
        end
        w[j] = t
    end
    normalize(w)
end
```

Вычитание реализовано с использованием механизма заимствования, при котором при необходимости производится корректировка текущего разряда.

Умножение многозначных целых чисел столбиком

Выполним задание 3 с помощью языка Julia:

```
function mul_big(u, v, b)
    n, m = length(u), length(v)
    w = zeros(Int, n + m)

    for j in m:-1:1
        carry = 0
        for i in n:-1:1
            t = u[i]*v[j] + w[i+j] + carry
            w[i+j] = t % b
            carry = div(t, b)
        end
        w[j] += carry
    end
    normalize(w)
end
```

Алгоритм соответствует классическому умножению «столбиком», при котором каждый разряд одного множителя последовательно умножается на все разряды второго множителя с последующим суммированием частичных произведений.

Быстрое умножения многозначных целых чисел

Выполним задание 4 с помощью языка Julia:

```
function mul_fast(u, v, b)
    n, m = length(u), length(v)
    w = zeros(Int, n + m)
    t = 0

    for s in 0:n+m-2
        for i in max(0, s-m+1):min(s, n-1)
            t += u[n-i] * v[m-(s-i)]
        end
        w[n+m-s] = t % b
        t = div(t, b)
    end
    normalize(w)
end
```

Быстрый столбик использует диагональный принцип суммирования произведений цифр. Это позволяет избежать формирования промежуточных чисел и получить итоговый результат за один проход по диагоналям.

Оба алгоритма умножения приводят к одинаковому результату, что подтверждает корректность реализации.

Деление многозначных целых чисел с вычислением частного и остатка.

Выполним задание 5 с помощью языка Julia:

```
function div_big(u, v, b)
    q = Int[]
    r = Int[]

    for digit in u
        push!(r, digit)
        r = normalize(r)

        cnt = 0
        while length(r) > length(v) || (length(r) == length(v) && r >= v)
            r = sub_big(r, v, b)
            cnt += 1
        end
        push!(q, cnt)
    end

    normalize(q), normalize(r)
end
```

Алгоритм деления реализован по принципу классического длинного деления. Дели-

мое обрабатывается по цифрам, формируя текущий остаток. Для каждого шага определяется, сколько раз делитель помещается в текущем остатке, после чего вычисляется соответствующая цифра частного и обновляется остаток.

Полученные результаты

Сделаем вывод работы всех алгоритмов:

```
b = 10
u = [1, 2, 3, 4]
v = [5, 6, 7]
println("u = ", u)
println("v = ", v)

println("\nАлгоритм 1 — сложение")
println(add_big(u, v, b))

println("\nАлгоритм 2 — вычитание (u ≥ v)")
println(sub_big(u, v, b))

println("\nАлгоритм 3 — умножение столбиком")
println(mul_big(u, v, b))

println("\nАлгоритм 4 — быстрый столбик")
println(mul_fast(u, v, b))

println("\nАлгоритм 5 — деление")
q, r = div_big(u, v, b)
println("частное = ", q)
```

```
println("остаток = ", r)
```

Полученные результаты работы кода:

```
u = [1, 2, 3, 4]
v = [5, 6, 7]

Алгоритм 1 — сложение
[1, 8, 0, 1]

Алгоритм 2 — вычитание (u ≥ v)
[6, 6, 7]

Алгоритм 3 — умножение столбиком
[6, 9, 9, 6, 7, 8]

Алгоритм 4 — быстрый столбик
[6, 9, 9, 6, 7, 8]

Алгоритм 5 — деление
частное = [2]
остаток = [1, 0, 0]
```

Заключение

В ходе выполнения лабораторной работы были изучены и реализованы алгоритмы целочисленной арифметики многократной точности. Реализация показала, что стандартные арифметические операции могут быть выполнены над числами произвольной длины при использовании поразрядного представления данных. Полученные знания могут быть использованы при изучении криптографических алгоритмов и вычислительных методов.

Библиографическая справка

[1] Julia: https://ru.wikipedia.org/wiki/Julia