Отчёт по лабораторной работе №8: Целочисленная арифметика многократной точности

Дисциплина: Математические основы защиты информации и информационной безопасности

Манаева Варвара Евгеньевна

Содержание

Список иллюстраций

1 Общая информация о задании лабораторной работы

1.1 Цель работы

Ознакомиться с целочисленной арифметикой многократной точности.

1.2 Задание [1]

1. Реализовать алгоритмы из задания лабораторной работы.

2 Выполнение лабораторной работы [1]

2.1 Алгоритм 1. Сложение неотрицательных целых чисел

Исходный код написан на языке Julia [2]. Код функции, осуществляющей сложение неотрицательных целых чисел, представлен ниже.

```
function sum_accurate(u, v, b=10)
    k = 0
    u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only
    n_u = length(u_str); n_v = length(v_str)
    j = max(n_u, n_v)
    w = zeros(Int64, j+1)
    if n_u < n_v
        temp = zeros(Int64, j)
        temp[n_v - n_u + 1:j] = u_str
        u_str = [i for i in temp]
    elseif n_v < n_u
        temp = zeros(Int64, j)
        temp[n_u - n_v + 1:j] = v_str
        v_str = [i for i in temp]
```

```
    end
    while j != 0
        k_temp = (u_str[j] + v_str[j] + k) % b
        w[j+1] = k_temp
        k = round(Int, (u_str[j] + v_str[j] + k - k_temp) / b)
        j -= 1
    end
    w[1] = k
    return parse(Int, join(string.(w)))
end
```

2.1.1 Проверка работы функции

```
sum_accurate(12533,989)
```

Результат работы кода представлен ниже (рис. 2.1).

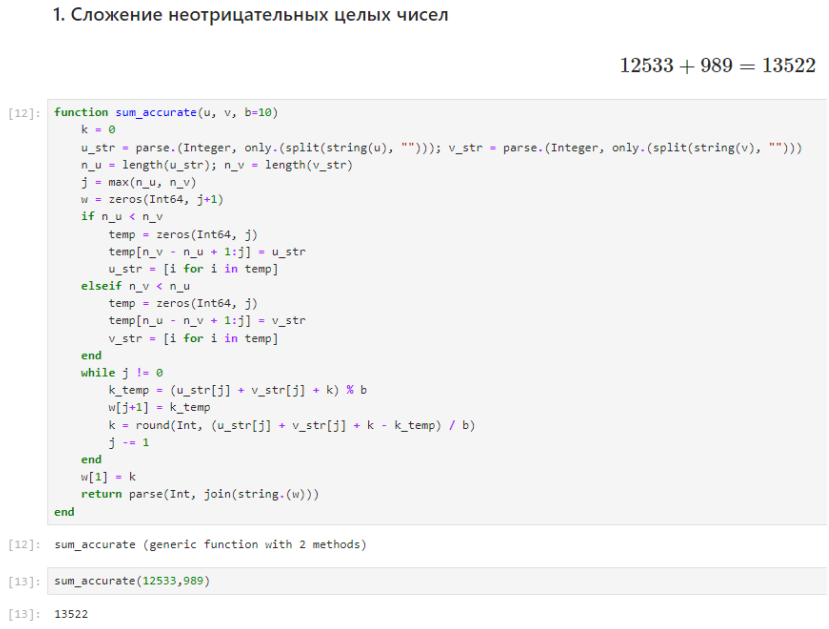

Рис. 2.1: Сумма целых неотрицательных чисел

2.2 Алгоритм 2. Вычитание неотрицательных целых чисел

Исходный код написан на языке Julia [2]. Код функции, осуществляющей вычитание целых чисел, представлен ниже.

```
function raz_accurate(u, v, b=10)
    if u < v
        return string(u) * " should be greater than " * string(v)
    end
    k = 0
    u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only
    n_u = length(u_str); n_v = length(v_str)
    j = max(n_u, n_v)
    w = zeros(Int64, j)
    if n_v < n_u
        temp = zeros(Int64, j)
        temp[n_u - n_v + 1:j] = v_str
        v_str = [i for i in temp]
    end
    while j != 0
        if u_str[j] < v_str[j]
            k = b
            u_str[j-1] -= 1
        else
            k = 0
        end
        k_temp = (u_str[j] - v_str[j] + k) % b
        w[j] += k_temp
        j -= 1
```

```
    end
    return parse(Int, join(string.(w)))
end
```

2.2.1 Проверка работы функции

```
raz_accurate(12533,989)
```

Результат работы кода представлен ниже (рис. 2.2).

2. Вычитание неотрицательных целых чисел

$$12533 - 989 = 11544$$

```
[14]: function raz_accurate(u, v, b=10)
          if u < v
              return string(u) * " should be greater than " * string(v)
          end
          k = 0
          u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only.(split(string(v), "")))
          n_u = length(u_str); n_v = length(v_str)
          j = max(n_u, n_v)
          w = zeros(Int64, j)
          if n_v < n_u
              temp = zeros(Int64, j)
              temp[n_u - n_v + 1:j] = v_str
              v_str = [i for i in temp]
          end
          while j != 0
              if u_str[j] < v_str[j]
                  k = b
                  u_str[j-1] -= 1
              else
                  k = 0
              end
              k_temp = (u_str[j] - v_str[j] + k) % b
              w[j] += k_temp
              j -= 1
          end
          return parse(Int, join(string.(w)))
      end

[14]: raz_accurate (generic function with 2 methods)

[15]: raz_accurate(12533,989)

[15]: 11544
```

Рис. 2.2: Вычитание целых чисел

2.3 Алгоритм 3. Умножение неотрицательных целых чисел

Исходный код написан на языке Julia [2]. Код функции, осуществляющей умножение неотрицательных чисел, представлен ниже.

```
function umn_accurate(u, v, b=10)
    k = 0
    u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only
    i = length(u_str); j = length(v_str)
    w = zeros(Int64, i + j)
    while j > 0
        i = length(u_str)
        k = 0
        while i > 0
            k_temp = u_str[i] * v_str[j] + w[i+j] + k
            w[i+j] = k_temp % b
            k = round(Int, (k_temp - w[i+j]) / b)
            i -= 1
        end
        w[j] = k
        j -= 1
    end
    return parse(Int, join(string.(w)))
end
```

2.3.1 Проверка работы функции

```
umn_accurate(12533,989)
```

Результат работы кода представлен ниже (рис. 2.3).

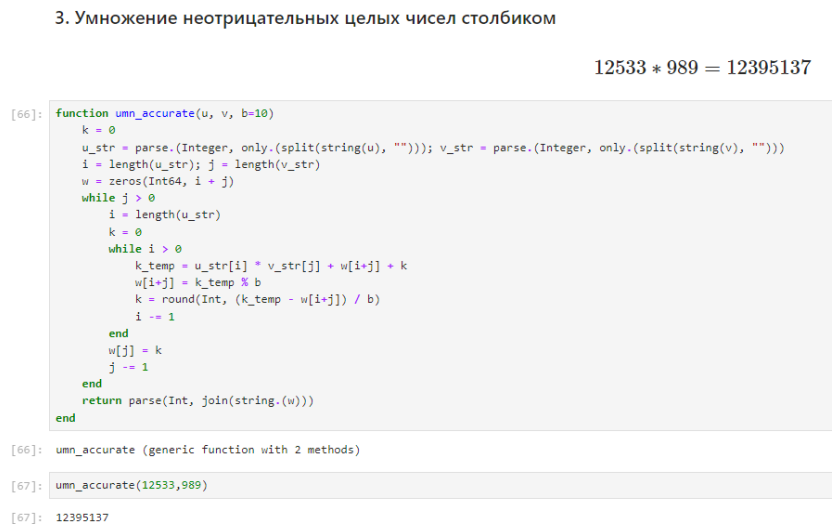

Рис. 2.3: Умножение неотрицательных чисел

2.4 Алгоритм 4. Быстрое умножение столбиком

Исходный код написан на языке Julia [2]. Код функции, осуществляющей быстрое умножение столбиком, представлен ниже.

```
function umn_fast(u, v, b=10)
    u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only
    n = length(u_str); m = length(v_str)
    w = zeros(Int64, n + m)
    t = 0
    for s in 0:m+n-1
        for i in 0:s
            if n-i <= 0 || m-s+i <= 0
                continue
            end
            t += u_str[n-i] * v_str[m-s+i]
        end
        w[n+m-s] = t % b
```

```
            t = round(Int64, (t - w[n+m-s]) / b)
        end
        return parse(Int, join(string.(w)))
    end
```

2.4.1 Проверка работы функции

```
umn_fast(12533,989)
```

Результат работы кода представлен ниже (рис. 2.4).

4. Быстрый столбик

$$12533 * 989 = 12395137$$

```
[78]: function umn_fast(u, v, b=10)
          u_str = parse.(Integer, only.(split(string(u), ""))); v_str = parse.(Integer, only.(split(string(v), "")))
          n = length(u_str); m = length(v_str)
          w = zeros(Int64, n + m)
          t = 0
          for s in 0:m+n-1
              for i in 0:s
                  if n-i <= 0 || m-s+i <= 0
                      continue
                  end
                  t += u_str[n-i] * v_str[m-s+i]
              end
              w[n+m-s] = t % b
              t = round(Int64, (t - w[n+m-s]) / b)
          end
          return parse(Int, join(string.(w)))
      end

[78]: umn_fast (generic function with 2 methods)

[79]: umn_fast(12533,989)

[79]: 12395137
```

Рис. 2.4: Быстрое умножение столбиком

3 Выводы

В результате работы мы ознакомились с целочисленной арифметикой многократной точности.

Также были записаны скринкасты:

На RuTube:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

На Платформе:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

Список литературы

1. Лабораторная работа №?. ?тема? [Электронный ресурс]. RUDN, 2024. URL: https://esystem.rudn.ru/pluginfile.php/2368510/mod_folder/content/0/lab03.pdf.
2. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL: https://docs.julialang.org/en/v1/.