Лабораторная работа 8

Попов Дмитрий Павлович, НФИмд-01-23

Содержание

List of Figures

List of Tables

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

дисциплина: Математические основы защиты информации и информационной безопасности

Преподователь: Кулябов Дмитрий Сергеевич

Студент: Попов Дмитрий Павлович

Группа: НФИмд-01-23

МОСКВА

2023 г.

1 Цель работы

Освоить на практике целочисленную арифметику многократной точности.[1]

2 Выполнение лабораторной работы

Требуется реализовать:

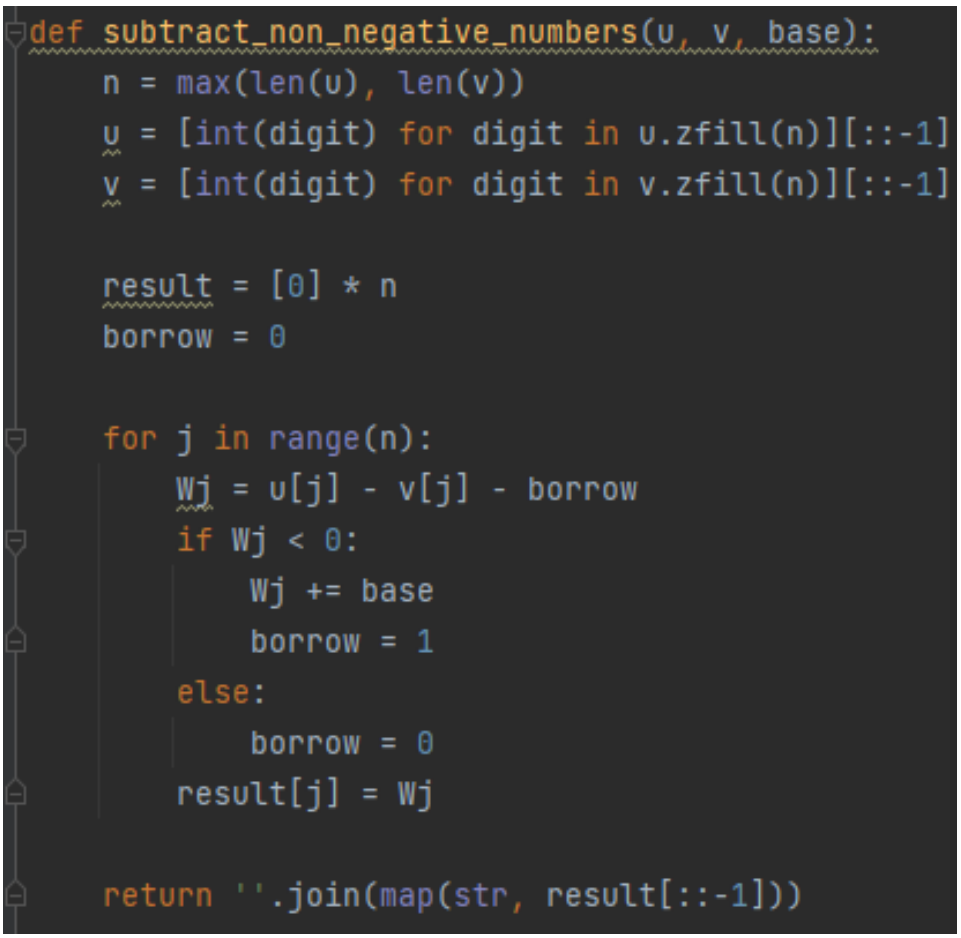
1. Алгоритм сложения неотрицательных целых чисел
2. Алгоритм вычитания неотрицательных целых чисел
3. Алгоритм умножения неотрицательных целых чисел столбиком
4. Алгоритм деления многоразрядных целых чисел

Алгоритм сложения неотрицательных целых чисел основан на стандартном методе сложения в столбик. Две числа выравниваются по разрядам, затем происходит поэлементное сложение с учетом переносов. Результат представляется в виде строки. fig. 2.1.

```
def add_non_negative_numbers(u, v, base):
    n = max(len(u), len(v))
    u = [int(digit) for digit in u.zfill(n)][::-1]
    v = [int(digit) for digit in v.zfill(n)][::-1]

    result = [0] * n
    carry = 0

    for j in range(n):
        Wj = u[j] + v[j] + carry
        result[j] = Wj % base
        carry = Wj // base

    return ''.join(map(str, result[::-1]))
```

Figure 2.1: add

Алгоритм вычитания неотрицательных целых чисел основан на стандартном методе вычитания в столбик. Два числа выравниваются по разрядам, и происходит поэлементное вычитание с учетом заемов. Результат представляется в виде строки. fig. 2.2.



```
def subtract_non_negative_numbers(u, v, base):
    n = max(len(u), len(v))
    u = [int(digit) for digit in u.zfill(n)][::-1]
    v = [int(digit) for digit in v.zfill(n)][::-1]

    result = [0] * n
    borrow = 0

    for j in range(n):
        Wj = u[j] - v[j] - borrow
        if Wj < 0:
            Wj += base
            borrow = 1
        else:
            borrow = 0
        result[j] = Wj

    return ''.join(map(str, result[::-1]))
```

Figure 2.2: sub

Алгоритм умножения неотрицательных чисел столбиком базируется на стандартном методе умножения в столбик. Два числа представлены в виде списков цифр, и происходит поэлементное умножение с учетом позиции разрядов. Промежуточные результаты суммируются, и конечный результат представляется в виде строки. fig. 2.3.

```python
def multiply_non_negative_numbers(u, v):
    len_u, len_v = len(u), len(v)
    result = [0] * (len_u + len_v)

    for i in range(len_u - 1, -1, -1):
        carry = 0
        for j in range(len_v - 1, -1, -1):
            temp = int(u[i]) * int(v[j]) + carry + result[i + j + 1]
            result[i + j + 1] = temp % 10
            carry = temp // 10
        result[i] += carry

    result_str = ''.join(map(str, result))
    return result_str.lstrip('0') or '0'
```

Figure 2.3: mult

Алгоритм деления многоразрядных целых чисел основан на делении в столбик. Делимое и делитель представлены в виде списков цифр. Алгоритм пошагово вычисляет цифры частного и остаток, используя текущие разряды. Результаты объединяются в строки для представления частного и остатка. fig. **??** fig. **??**.

```python
def divide_large_numbers(dividend, divisor):
    # Преобразование строк в списки цифр
    dividend = [int(digit) for digit in str(dividend)]
    divisor = [int(digit) for digit in str(divisor)]

    quotient = []  # Частное
    remainder = 0  # Остаток

    for digit in dividend:
        current_dividend = remainder * 10 + digit
        current_quotient = current_dividend // divisor[0]
        remainder = current_dividend % divisor[0]
        quotient.append(current_quotient)
```

```python
    # Проход по остальным разр
    for i in range(len(divider
        current_quotient = quo
        current_remainder = re
        j = 1

        while j < len(divisor)
            current_dividend =
            current_quotient =
            current_remainder
            j += 1

        quotient[i] = current_
        remainder = current_re

    # Приведение к строке
    quotient_str = ''.join(map
    remainder_str = str(remain

    return quotient_str, remai
```

Вывод программы: fig. 2.4.

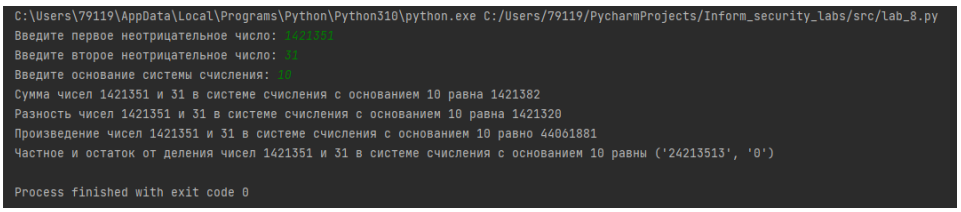


Figure 2.4: output

3 Выводы

В результате выполнения работы я освоил на практике целочисленную арифметику многократной точности.

4 Список литературы

1. Методические материалы курса