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

Целочисленная арифметика многократной точности

Терентьев Егор Дмитриевич 1132236902

Содержание

List of Figures

List of Tables

1 Цель работы

Освоить на практике целочисленную арифметику многократной точности

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

2.1 Алгоритмы

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.

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

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.

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

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: output

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. 2.4.

```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)

    # Проход по остальным разрядам
    for i in range(len(dividend) - len(divisor) + 1):
        current_quotient = quotient[i]
        current_remainder = remainder
        j = 1

        while j < len(divisor) and j + i < len(dividend):
            current_dividend = current_remainder * 10 + dividend[i + j]
            current_quotient = current_dividend // divisor[j]
            current_remainder = current_dividend % divisor[j]
            j += 1

        quotient[i] = current_quotient
        remainder = current_remainder

    # Приведение к строке
    quotient_str = ''.join(map(str, quotient)).lstrip('0') or '0'
    remainder_str = str(remainder)

    return quotient_str, remainder_str
```

Figure 2.4: output

Вывод программы: fig. 2.5.

```
Введите первое неотрицательное число: 12345
Введите второе неотрицательное число: 12
Введите основание системы счисления: 10
Сумма чисел 12345 и 12 в системе счисления с основанием 10 равна 12357
Разность чисел 12345 и 12 в системе счисления с основанием 10 равна 12333
Произведение чисел 12345 и 12 в системе счисления с основанием 10 равно 148140
Частное и остаток от деления чисел 12345 и 12 в системе счисления с основанием 10 равны ('1
```

Figure 2.5: output

3 Выводы

В результате выполнения работы я освоил на практике дискретное логарифмирование в конечном поле.

4 Список литературы

1. Методические материалы курса