

Лабораторная работа 8

Серенко Данил Сергеевич, НФИмд-01-23

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

дисциплина: Математические основы защиты информации и информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Серенко Данил Сергеевич

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения 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

Цель работы

Освоить на практике целочисленную арифметику многократной точности

Выполнение лабораторной работы

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

```
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 в список цифр, выравниваем по длине и разворачиваем
    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]))
```

main_func

2. 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.

```
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 строку в список цифр, выравниваем по длине и разворачиваем
    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]))
```

output

3. 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.

```
4
5 def multiply_non_negative_numbers(u, v):
6     len_u, len_v = len(u), len(v)
7     result = [0] * (len_u + len_v)
8
9     for i in range(len_u - 1, -1, -1):
10        carry = 0
11        for j in range(len_v - 1, -1, -1):
12            temp = int(u[i]) * int(v[j]) + carry + result[i + j + 1]
13            result[i + j + 1] = temp % 10
14            carry = temp // 10
15        result[i] += carry
16
17    result_str = ''.join(map(str, result))
18    return result_str.lstrip('0') or '0'
```

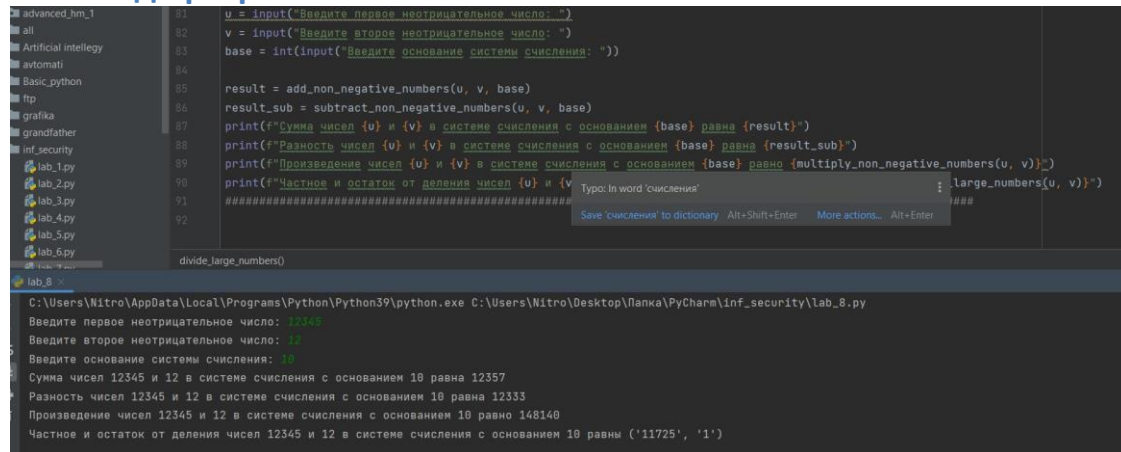
output

4. 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.

```
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
```

output

5. Вывод программы:



```
91 u = input("Введите первое неотрицательное число: ")
92 v = input("Введите второе неотрицательное число: ")
93 base = int(input("Введите основание системы счисления: "))
94
95 result = add_non_negative_numbers(u, v, base)
96 result_sub = subtract_non_negative_numbers(u, v, base)
97 print(f"Сумма чисел {u} и {v} в системе счисления с основанием {base} равна {result}")
98 print(f"Разность чисел {u} и {v} в системе счисления с основанием {base} равна {result_sub}")
99 print(f"Произведение чисел {u} и {v} в системе счисления с основанием {base} равно {multiply_non_negative_numbers(u, v)}")
100 print(f"Частное и остаток от деления чисел {u} и {v} в системе счисления с основанием {base} равны {divide_large_numbers(u, v)}")
101 #####
102
103 Save 'счисления' to dictionary Alt+Shift+Enter More actions... Alt+Enter
104
105 divide_large_numbers()
```

lab.8 >

C:\Users\Nitro\AppData\Local\Programs\Python\Python39\python.exe C:\Users\Nitro\Desktop\Панка\PyCharm\inf_security\lab_8.py

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 ('11725', '1')

output

Выводы

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