

## Лабораторная работа №8

Математические основы защиты информации и информационной безопасности

---

Мануэл Марсия Педру

2026

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

:::::::::: {.columns align=center} :::{.column width="70%"}

- Мануэл Марсия Педру
- Студент группы НФИмд-02-25
- Студ. билет 1032255503
-

## Цель лабораторной работы

---

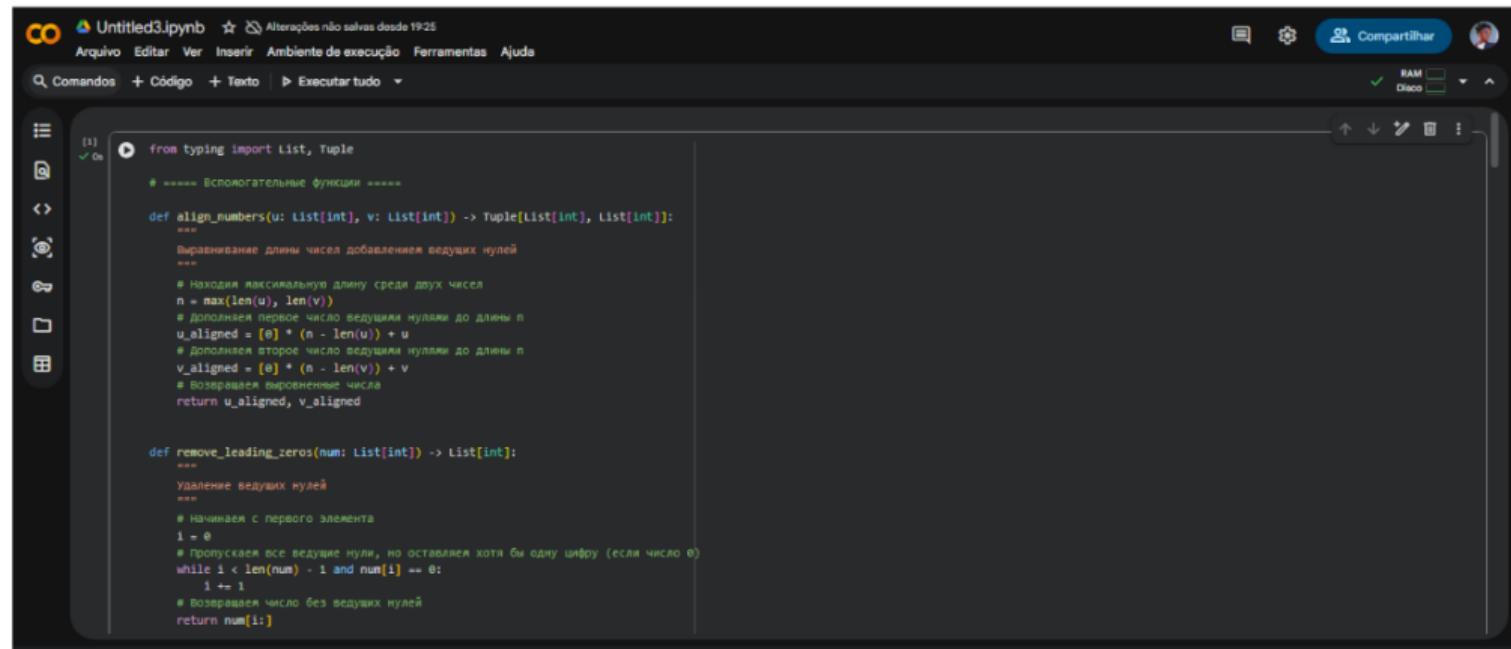
- Изучить алгоритмы для выполнения арифметических операций с большими целыми числами и научиться их реализовывать.

## Выполнение лабораторной работы

---

## Реализация алгоритма 1 (сложение неотрицательных целых чисел)

Выполним реализацию этого алгоритма на языке python (рис. (fig:001?)) и (рис. (fig:002?)):



The screenshot shows a Jupyter Notebook cell with the following Python code:

```
[1]: from typing import List, Tuple

# ===== Вспомогательные функции =====

def align_numbers(u: List[int], v: List[int]) -> Tuple[List[int], List[int]]:
    """
    Выравнивание длины чисел добавлением ведущих нулей
    """

    # Нахождение максимальную длину среди двух чисел
    n = max(len(u), len(v))
    # Дополняем первое число ведущими нулями до длины n
    u_aligned = [0] * (n - len(u)) + u
    # Дополняем второе число ведущими нулями до длины n
    v_aligned = [0] * (n - len(v)) + v
    # Возвращаем выровненные числа
    return u_aligned, v_aligned

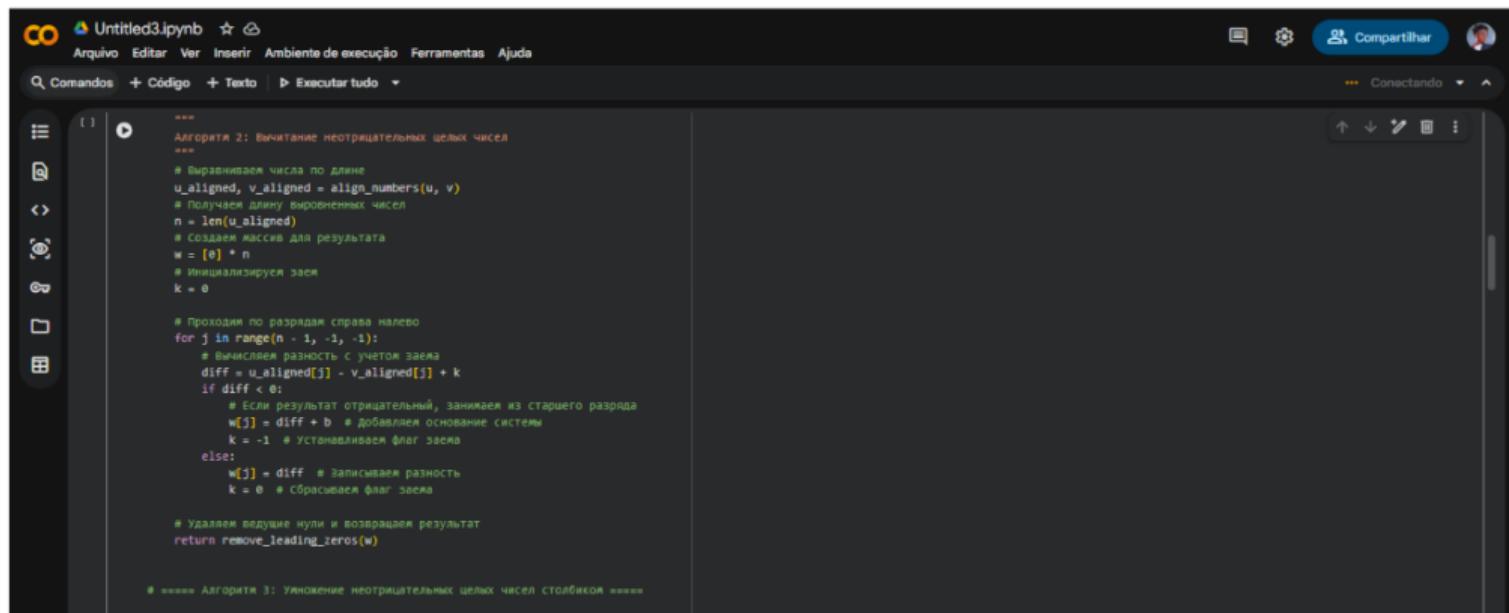
def remove_leading_zeros(num: List[int]) -> List[int]:
    """
    Удаление ведущих нулей
    """

    # Начинаем с первого элемента
    i = 0
    # Пропускаем все ведущие нули, но оставляем хотя бы одну цифру (если число 0)
    while i < len(num) - 1 and num[i] == 0:
        i += 1
    # Возвращаем число без ведущих нулей
    return num[i:]
```

Рис. 1: Реализация алгоритма 1

## Реализация алгоритма 2 (вычитание неотрицательных целых чисел)

Выполним реализацию этого алгоритма на языке python (рис. (fig:003?)):



The screenshot shows a Jupyter Notebook cell with the following code:

```
Untitled3.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda
Q Comandos + Código + Texto Executar tudo
Compartilhar Conectando


"""
Алгоритм 2: Вычитание неотрицательных целых чисел
"""

# Выравниваем числа по длине
u_aligned, v_aligned = align_numbers(u, v)
# Получаем длину выровненных чисел
n = len(u_aligned)
# Создаем массив для результата
w = [0] * n
# Инициализируем заем
k = 0

# Проходим по разрядам справа налево
for j in range(n - 1, -1, -1):
    # Вычисляем разность с учетом заема
    diff = u_aligned[j] - v_aligned[j] + k
    if diff < 0:
        # Если результат отрицательный, занимаем из старшего разряда
        w[j] = diff + b  # добавляем основание системы
        k = -1  # Устанавливаем флаг заема
    else:
        w[j] = diff  # записываем разность
        k = 0  # Сбрасываем флаг заема

# Удаляем ведущие нули и возвращаем результат
return remove_leading_zeros(w)

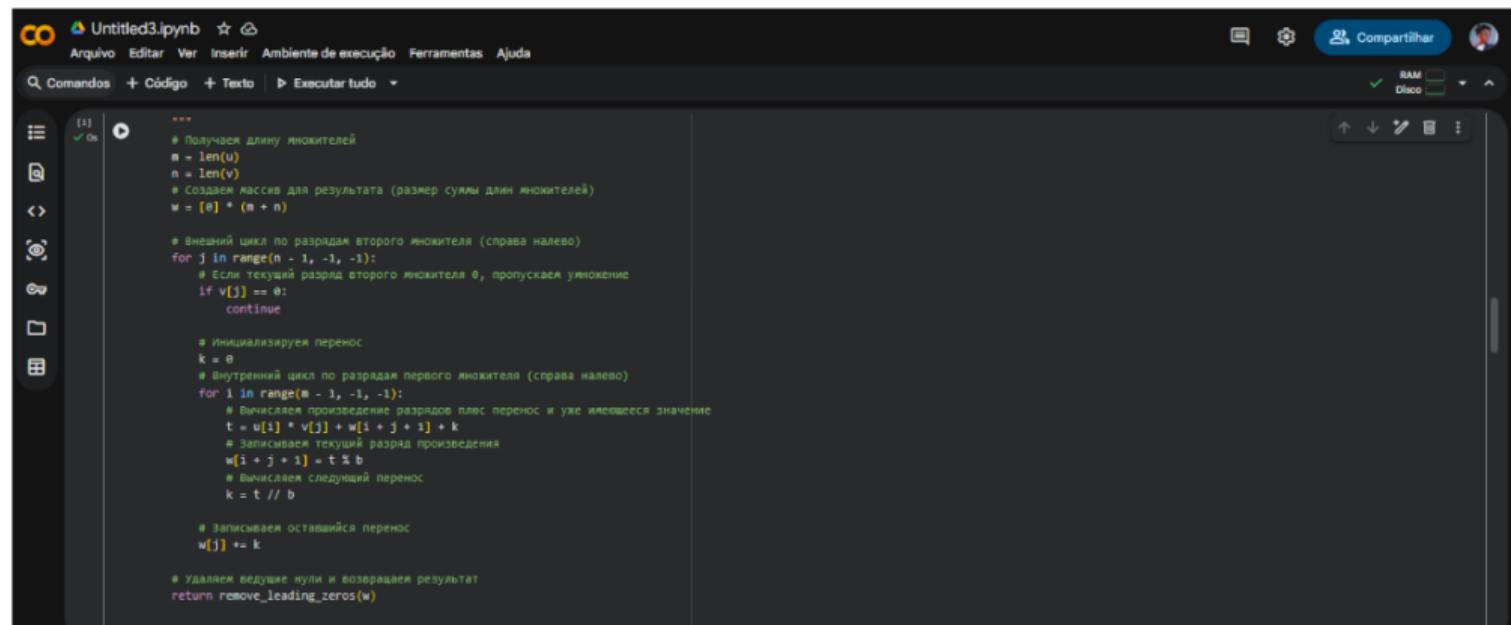
# ===== Алгоритм 3: Умножение неотрицательных целых чисел столбиком =====


```

Рис. 3: Реализация алгоритма 2

## Реализация алгоритма 3 (умножение неотрицательных целых чисел столбиком)

Выполним реализацию этого алгоритма на языке python (рис. (fig:004?)):

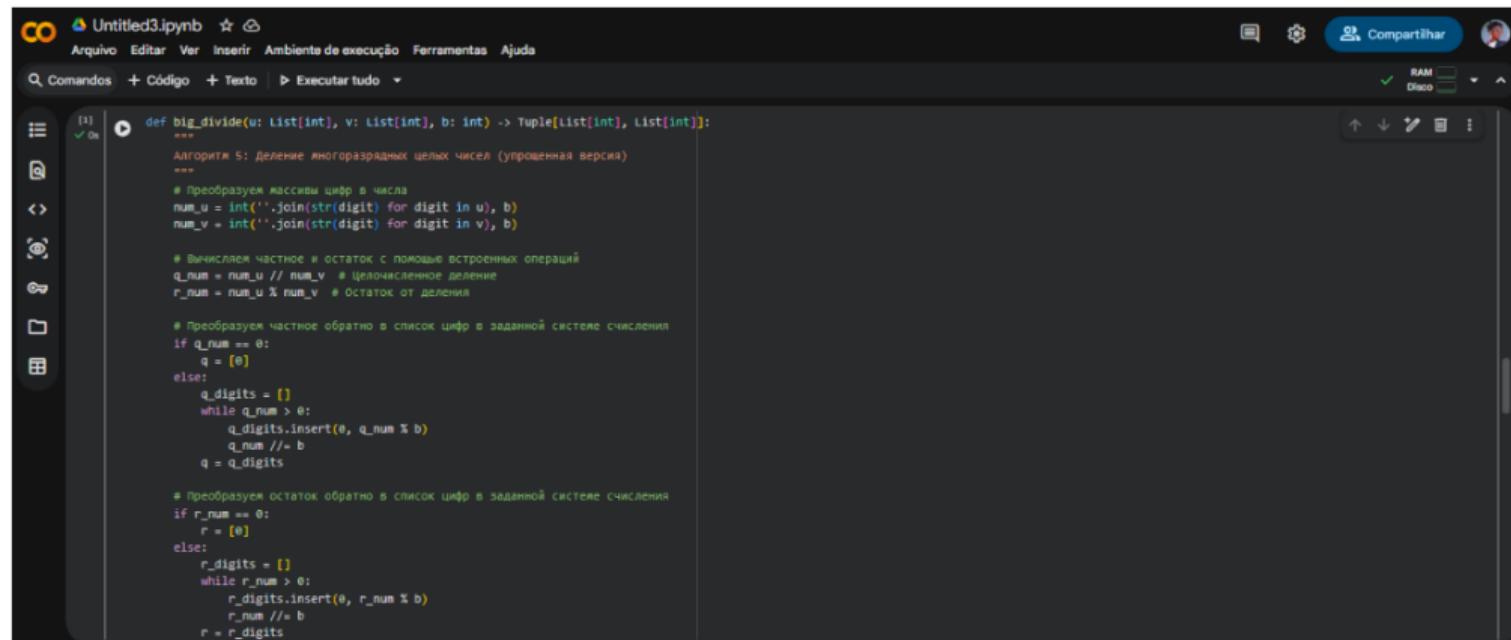


```
Untitled3.ipynb Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda
Comandos + Código + Texto Executar tudo RAM Disco
[1] Os
***  
# Получаем длину множителей  
n = len(u)  
m = len(v)  
# Создаем массив для результата (размер суммы длин множителей)  
w = [0] * (n + m)  
  
# Внешний цикл по разрядам второго множителя (справа налево)  
for j in range(m - 1, -1, -1):  
    # Если текущий разряд второго множителя 0, пропускаем умножение  
    if v[j] == 0:  
        continue  
  
    # Инициализируем перенос  
    k = 0  
    # Внутренний цикл по разрядам первого множителя (справа налево)  
    for i in range(n - 1, -1, -1):  
        # Вычисляем произведение разрядов плюс перенос и уже имеющееся значение  
        t = u[i] * v[j] + w[i + j + 1] + k  
        # Записываем текущий разряд произведения  
        w[i + j + 1] = t % b  
        # Вычисляем следующий перенос  
        k = t // b  
  
    # Записываем оставшийся перенос  
    w[j] += k  
  
# Удаляем ведущие нули и возвращаем результат  
return remove_leading_zeros(w)
```

Рис. 4: Реализация алгоритма 3

## Реализация алгоритма 5 (деление многоразрядных целых чисел)

Выполним реализацию этого алгоритма на языке python (рис. (fig:005?)):



The screenshot shows a Jupyter Notebook cell with the code for the division algorithm. The code is well-commented and follows the pseudocode provided in the slide.

```
[1]: def big_divide(u: List[int], v: List[int], b: int) -> Tuple[List[int], List[int]]:
    """
    Алгоритм 5: Деление многоразрядных целых чисел (упрощенная версия)
    """
    # Преобразуем массивы цифр в числа
    num_u = int(''.join(str(digit) for digit in u), b)
    num_v = int(''.join(str(digit) for digit in v), b)

    # Вычисляем частное и остаток с помощью встроенных операций
    q_num = num_u // num_v # Целочисленное деление
    r_num = num_u % num_v # Остаток от деления

    # Преобразуем частное обратно в список цифр в заданной системе счисления
    if q_num == 0:
        q = [0]
    else:
        q_digits = []
        while q_num > 0:
            q_digits.insert(0, q_num % b)
            q_num //= b
        q = q_digits

    # Преобразуем остаток обратно в список цифр в заданной системе счисления
    if r_num == 0:
        r = [0]
    else:
        r_digits = []
        while r_num > 0:
            r_digits.insert(0, r_num % b)
            r_num //= b
        r = r_digits
```

Рис. 5: Реализация алгоритма 5

## Вывод

---

## Вывод

---

- В ходе выполнения лабораторной работы были изучены алгоритмы для выполнения арифметических операций с большими целыми числами, а также написаны их алгоритмы на языке python.