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

Целочисленная арифметика многократной точности

Лубышева Ярослава Михайловна

Содержание

List of Figures

List of Tables

1 Цель работы

Выполнить задание к лабораторной работе № 8 [1].

2 Задание

1. Ознакомиться с алгоритмами: сложение неотрицательных целых чисел, вычитание неотрицательных целых чисел, умножение неотрицательных целых чисел столбиком, умножение неотрицательных целых чисел столбиком (быстрый столбик), деление многоразрядных целых чисел.
2. Реализовать алгоритмы программно.

3 Выполнение лабораторной работы

Для реализации алгоритмов была написана программа на языке программирования Python (fig. 3.1 - fig. 3.6).

```
# Сложение неотрицательных целых чисел
# Вход: два неотрицательных числа u=u_1u_2...u_n и v=v_1v_2...v_n;
# разрядность чисел n; основание системы счисления b;
# Выход: сумма w=w_0w_1...w_n, где w_0 - цифра переноса - всегда равна 0 или 1.
def plus(u, v, n, b):
  j = n-1  # идет по разрядам (-1 так как индексирование списков начинается с 0)
  k = 0  # следит за переносом
  w = [None] * n

  while j >= 0:
    w[j] = (u[j]+v[j]+k) % b # наименьший неотриц. вычет в данном классе вычетов
    k = (u[j]+v[j]+k) // b
    j -= 1

  w.insert(0, k)
  return w
```

Figure 3.1: Программная реализация алгоритма сложения неотрицательных целых чисел

```python
# Вычитание неотрицательных целых чисел
# Вход: два неотрицательных числа u=u_1u_2...u_n и v=v_1v_2...v_n, u > v;
# разрядность чисел n; основание системы счисления b;
# Выход: разрядность w=w_1w_2...w_n = u - v.
def minus(u, v, n, b):
  if u <= v:
    return "Введите другие числа: первое число должно быть больше второго"

  j = n-1  # так как индексирование списков начинается с 0
  k = 0  #заем из старшего разряда
  w = [None] * n

  while j >= 0:
    w[j] = (u[j]-v[j]+k) % b # наименьший неотриц. вычет в данном классе вычетов
    k = (u[j]-v[j]+k) // b
    j -= 1

  return w
```

Figure 3.2: Программная реализация алгоритма вычитания неотрицательных целых чисел

```
# Умножение неотрицательных целых чисел столбиком
# Вход: числа u=u_1u_2...u_n и v=v_1v_2...v_m; основание системы счисления b;
# Выход: произведение w = u*v = w_1w_2...w_(m+n).
def long_multiplication(u, v, b):
  n = len(u)
  m = len(v)
  w = [0] * (m+n)
  # j перемещается по номерам разрядов числа v от младших к старшим
  # условие шага 6
  for j in range(m-1, -1, -1):
    # шаг 2
    if v[j] != 0:
      # шаг 3
      k = 0  # отвечает за перенос
      # условие шага 5
      for i in range(n-1, -1, -1):
        # шаг 4
        t = u[i] * v[j] + w[i+j+1] + k
        w[i+j+1] = t % b  # наименьший неотриц. вычет в данном классе вычетов
        k = t // b
      # если i==0
      w[j] = k
  return w
```

Figure 3.3: Программная реализация алгоритма умножения неотрицательных целых чисел столбиком

```python
# Умножение неотрицательных целых чисел столбиком (быстрый столбик)
# Вход: числа u=u_1u_2...u_n и v=v_1v_2...v_m; основание системы счисления b;
# Выход: произведение w = u*v = w_1w_2...w_(m+n).
def quick_long_multiplication(u, v, b):
  n = len(u)
  m = len(v)
  w = [0] * (m+n)
  # шаг 1
  t = 0
  # шаг 2
  for s in range(0, m+n):
    # шаг 3
    for i in range(0, s+1):
      if 0<=n-i-1<n and 0<=m-s+i-1<m:
        t += u[n-i-1] * v[m-s+i-1]
    # шаг 4
    w[m+n-s-1] = t % b  #наименьший неотрицательный вычет по модулю b
    t //= b
  return w
```

Figure 3.4: Программная реализация алгоритма умножения неотрицательных целых чисел столбиком (быстрый столбик)

```python
# Преобразование числа-списка в int
def list_to_int(lst):
  return int(''.join(map(str, lst)))

# преобразование int в число-строку
# len - количество цифр в числе-строке до преобазования в int
def int_to_list(num, num_size):
  num_list = list(map(int, str(num)))
  # добавляем недостающие нули перед значением
  num_list = [0]*(num_size - len(num_list)) + num_list
  return num_list
```

Figure 3.5: Программная реализация преобразования числа-списка в int и обратно (для алгоритма деления)

```python
# Деление многоразрядных целых чисел
# Вход: числа u=u_n...u_1u_0, v=v_t...v_1v_0, n>=t>=1, v_t!=0,
# где n и t -разрядность чисел; основание системы счисления b;
# Выход: частное q=q_(n-t)...q_0, остаток r=r_t...r_0.
def dividing_multi_digit_integers(u, v, b):
  n = len(u) - 1
  t = len(v) - 1
  q = [0] * (n-t+1)
  # шаг 2
  u, v = list_to_int(u), list_to_int(v)  # преобразуем в int
  while u >= v*b**(n-t):
    q[n-t] += 1
    u -= v*b**(n-t)
  # шаг 3
  for i in range(n, t, -1):
    u, v = int_to_list(u, n+1), int_to_list(v, t+1)  # преобразуем в list
    # шаг 3.1
    if u[n-i]>=v[0]:
      q[i-t-1] = b - 1
    else:
      q[i-t-1] = (u[n-i]*b + u[n-i+1]) // v[0]
    # шаг 3.2
    while q[i-t-1]*(v[0]*b + v[1]) > u[n-i]*b**2 + u[n-i+1]*b + u[n-i+2]:
      q[i-t-1] -= 1
    # шаг 3.3
    u, v = list_to_int(u), list_to_int(v)  # преобразуем в int
    u -= q[i-t-1] * b**(i-t-1) * v
    # шаг 3.4
    if u<0:
      u += v*b**(i-t-1)
      q[i-t-1] -= 1
  # шаг 4
  r = u
  return q, r
```

Figure 3.6: Программная реализация алгоритма деления многоразрядных целых чисел

Результаты работы алгоритмов представлены на рисунке ниже (fig. 3.3).

```
u = [3, 0, 6]
v = [1, 0, 4]
n = 3
b = 10

print(f"{plus(u, v, n, b)}")
print(f"{minus(u, v, n, b)}")
print(f"{long_multiplication([1, 1], [1, 1], b)}")
print(f"{quick_long_multiplication([1, 1], [1, 1], b)}")
print(f"{dividing_multi_digit_integers([1, 2, 7, 8], [4, 1, 9], b)}")
```

```
[0, 4, 1, 0]
[2, 0, 2]
[0, 1, 2, 1]
[0, 1, 2, 1]
([3, 0], 21)
```

Figure 3.7: Результаты работы алгоритмов целочисленной арифметики многократной точности

4 Выводы

Выполнено задание к лабораторной работе № 8.

5 Список литературы

1. Методические материалы курса