Лабораторная работа №8

Презентация

Миличевич Александра

15 февраля 2025

Российский университет дружбы народов, Москва, Россия

Цель работы

Арифметика в системах счисления

Ознакомление с алгоритмами целочисленной арифметики многократной точности, а также их последующая программная реализация.

Длинная арифметика

Высокоточная (длинная) арифметика — это операции (базовые арифметические действия, элементарные математические функции и пр.) над числами большой разрядности (многоразрядными числами), т.е. числами, разрядность которых превышает длину машинного слова универсальных процессоров общего назначения (более 128 бит).

Арифметическое сложение

Арифметическое сложение неотрицательных чисел — это операция, при которой два числа складываются поразрядно, начиная с младших разрядов. Если сумма цифр в текущем разряде превышает основание системы счисления, происходит перенос единицы в следующий, более старший разряд. Результатом сложения является новое число, представленное в той же системе счисления. Этот процесс повторяется для всех разрядов, включая возможный перенос на последнем шаге.

1. Функция `addition(num1, num2, base)`

Описание:

Вход: - `num1` (str): Первое число в виде строки. - `num2` (str): Второе число в виде строки. - `base` (int): Основание системы счисления.

Выход: - `list`: Список цифр, представляющий сумму двух чисел.

Как работает: 1. **Инициализация:** - Определяется разрядность чисел `n`. - Переменная `carry` устанавливается в 0. - Создается пустой список `result`.

2. **Цикл по разрядам:**
   - Цикл проходит по разрядам чисел от младшего к старшему.
   - На каждой итерации:
     - Вычисляется сумма цифр и перенос.
     - Младшая цифра результата добавляется в `result`.
     - Вычисляется новый перенос.

3. **Разворот результата:** Список `result` разворачивается.

4. **Возврат результата:** Возвращается список `result`.

```python
import math

# --- Алгоритм 1: Сложение неотрицательных целых чисел ---
def addition(num1, num2, base):
    """
    Выполняет сложение двух неотрицательных целых чисел в заданной системе счисления.

    Args:
        num1 (str): Первое число в виде строки.
        num2 (str): Второе число в виде строки.
        base (int): Основание системы счисления.

    Returns:
        list: Список цифр, представляющий сумму двух чисел.
    """
    n = len(num1)  # Разрядность чисел (предполагаем, что num1 и num2 имеют одинаковую длину)
    carry = 0      # Перенос
    result = []    # Список для хранения цифр результата

    for i in range(n - 1, -1, -1):  # Идем по разрядам от младшего к старшему
        digit_sum = int(num1[i]) + int(num2[i]) + carry  # Складываем цифры и перенос
        result.append(digit_sum % base)        # Добавляем младшую цифру результата
        carry = digit_sum // base                # Вычисляем новый перенос

    result.reverse() # Разворачиваем результат, так как цифры добавлялись в обратном порядке.
    return result
```


Рис. 1: addition

Арифметическое вычитание

Арифметическое вычитание неотрицательных чисел — это операция, при которой из одного числа (уменьшаемого) вычитается другое число (вычитаемое) поразрядно, начиная с младших разрядов. Если цифра в уменьшаемом меньше цифры в вычитаемом, происходит заимствование единицы из следующего, более старшего разряда. Результатом вычитания является новое число, которое может быть меньше исходного или равно нулю. Э тот процесс повторяется для всех разрядов, включая возможное заимствование на каждом шаге.

2. Функция `subtraction(num1, num2, base)`

Описание:

Вход: - `num1` (str): Уменьшаемое в виде строки. - `num2` (str): Вычитаемое в виде строки. - `base` (int): Основание системы счисления.

Выход: - `list`: Список цифр, представляющий разность двух чисел.

Как работает: 1. **Инициализация:** - Определяется разрядность чисел `n`. - Переменная `borrow` устанавливается в 0. - Создается пустой список `result`.

2. **Цикл по разрядам:**
   - Цикл проходит по разрядам чисел от младшего к старшему.
   - На каждой итерации:
     - Вычисляется разность цифр и заем.
     - Цифра результата добавляется в `result`.

3. **Разворот результата:** Список `result` разворачивается.

4. **Возврат результата:** Возвращается список `result`.

```python
# --- Алгоритм 2: Вычитание неотрицательных целых чисел ---
def subtraction(num1, num2, base):
    """
    Выполняет вычитание двух неотрицательных целых чисел в заданной системе счисления.
    Предполагается, что num1 >= num2.

    Args:
        num1 (str): Уменьшаемое в виде строки.
        num2 (str): Вычитаемое в виде строки.
        base (int): Основание системы счисления.

    Returns:
        list: Список цифр, представляющий разность двух чисел.
    """
    n = len(num1) # Разрядность чисел
    borrow = 0 # Заем
    result = [] # Список для хранения цифр результата

    for i in range(n - 1, -1, -1): # Идем по разрядам от младшего к старшему
        digit_diff = int(num1[i]) - int(num2[i]) + borrow # Вычисляем разность цифр и заем

        if digit_diff < 0: # Если результат отрицательный
            digit_diff += base # Добавляем основание, чтобы получить неотрицательную цифру
            borrow = -1 # Устанавливаем заем в -1
        else:
            borrow = 0 # Иначе, заем равен 0

        result.append(digit_diff) # Добавляем цифру результата
```

Арифметическое умножение неотрицательных чисел по столбику

Арифметическое умножение неотрицательных чисел по столбику — это операция, при которой одно число (множимое) у множается на каждую цифру другого числа (множителя) по отдельности, начиная с младшего разряда. Результаты каждого умножения записываются со сдвигом влево на соответствующее количество разрядов, после чего все промежуточные результаты складываются. Если произведение цифр превышает основание системы счисления, старшая часть записывается как перенос и добавляется к следующему разряду. Итогом является новое число, представляющее произведение исходных чисел.

3. Функция `multiplication(num1, num2, base)`

Эта функция выполняет умножение двух неотрицательных чисел в заданной системе счисления.

Описание:

Вход:

- `num1` (str): Первое число в виде строки.
- `num2` (str): Второе число в виде строки.
- `base` (int): Основание системы счисления.

Выход:

- `list`: Список цифр, представляющий произведение двух чисел.

Как работает:

Инициализация: - Определяются длины чисел `n` (для `num1`) и `m` (для `num2`). - Создается список `result` длиной `n + m`, заполненный нулями, для хранения результата.

Внешний цикл по разрядам второго числа:

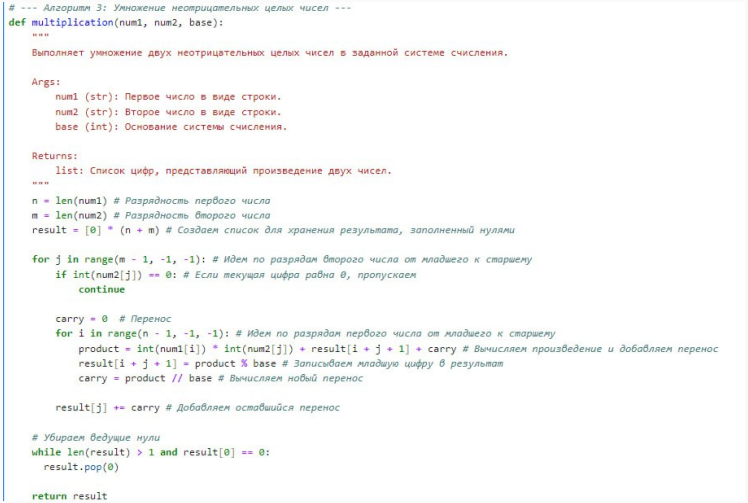
- Цикл `for` проходит по разрядам `num2` от младшего к старшему (справа налево).
- Если текущая цифра `num2[j]` равна 0, то итерация пропускается.

Внутренний цикл по разрядам первого числа:

- Цикл `for` проходит по разрядам `num1` от младшего к старшему.

На каждой итерации:

- Вычисляется произведение цифр на текущих разрядах и добавляется к предыдущему результату и переносу:
  `product = int(num1[i]) * int(num2[j]) + result[i + j + 1] + carry`.
- Младшая цифра произведения записывается в `result[i + j + 1]`:
  `result[i + j + 1] = product % base`.
- Вычисляется новый перенос:
  `carry = product // base`.



```python
# --- Алгоритм 3: Умножение неотрицательных целых чисел ---
def multiplication(num1, num2, base):
    """
    Выполняет умножение двух неотрицательных целых чисел в заданной системе счисления.

    Args:
        num1 (str): Первое число в виде строки.
        num2 (str): Второе число в виде строки.
        base (int): Основание системы счисления.

    Returns:
        list: Список цифр, представляющий произведение двух чисел.
    """
    n = len(num1) # Разрядность первого числа
    m = len(num2) # Разрядность второго числа
    result = [0] * (n + m) # Создаем список для хранения результата, заполненный нулями

    for j in range(m - 1, -1, -1): # Идем по разрядам второго числа от младшего к старшему
        if int(num2[j]) == 0: # Если текущая цифра равна 0, пропускаем
            continue

        carry = 0  # Перенос
        for i in range(n - 1, -1, -1): # Идем по разрядам первого числа от младшего к старшему
            product = int(num1[i]) * int(num2[j]) + result[i + j + 1] + carry # Вычисляем произведение и добавляем перенос
            result[i + j + 1] = product % base # Записываем младшую цифру в результат
            carry = product // base # Вычисляем новый перенос

        result[j] += carry # Добавляем оставшийся перенос

    # Убираем ведущие нули
    while len(result) > 1 and result[0] == 0:
      result.pop(0)

    return result
```

Рис. 3: multiplication

Арифметическое деление неотрицательных чисел — это операция, которая определяет, сколько раз одно число (делимое) может быть разделено на другое число (делитель) без остатка. Результатом деления является частное, которое представляет собой количество полных делений, и остаток, который показывает, сколько осталось после деления. Если делимое меньше делителя, то частное равно нулю , а остаток равен делимому. Деление на ноль не определено и приводит к ошибке, так как невозможно разделить число на ноль.

5. Функция `division(dividend, divisor, base)`

Эта функция выполняет деление неотрицательных целых чисел в заданной системе счисления.

Описание:
-Вход:

- `dividend` (str): Делимое в виде строки.
- `divisor` (str): Делитель в виде строки.
- `base` (int): Основание системы счисления.

Выход:

- `tuple`: Кортеж, содержащий частное (list) и остаток (str).

Как работает:
Инициализация: - Определяется длина делимого `dividend_len` и делителя `divisor_len`. - Проверяется, что делитель не равен нулю. - Проверяется, что

Предварительное деление:

Выполняется проверка, на случай, если делимое больше делителя на целую степень base.

Основной цикл деления:

Цикл проходит по разрядам делимого (справа налево).

На каждой итерации: Проверяется, что текущий разряд делимого входит в границы остатка. Определяется начальное значение для цифры частного. Цикл `while` уточняет значение цифры частного. Обновляется остаток с учетом новой цифры частного.

Возврат результата:

Возвращается кортеж, содержащий частное (`quotient`) и остаток (`remainder`).

```python
def division(dividend, divisor, base):
    """
    Выполняет деление неотрицательных целых чисел в заданной системе счисления.

    Args:
        dividend (str): Делимое в виде строки.
        divisor (str): Делитель в виде строки.
        base (int): Основание системы счисления.

    Returns:
        tuple: Кортеж, содержащий частное (list) и остаток (str).
    """
    dividend_len = len(dividend)  # Длина делимого
    divisor_len = len(divisor)  # Длина делителя

    # Проверяем, что делитель не ноль
    if int(divisor) == 0:
      return [], "Деление на ноль!"

    # Проверяем, что делимое больше или равно делителя
    if int(dividend) < int(divisor):
      return [0], dividend

    # Инициализация частного и остатка
    quotient = [0] * (dividend_len - divisor_len + 1)
    remainder = dividend  # Изначально остаток - это делимое

    # Условие деления делимого на делитель
    while int(remainder) >= int(divisor) * (base ** (dividend_len - divisor_len)):
       quotient[dividend_len - divisor_len] += 1
       remainder = int(remainder) - int(divisor) * (base ** (dividend_len - divisor_len))
    remainder = str(remainder) # Конвертируем обратно в строку

    # Основной цикл деления
    for i in range(dividend_len -1, divisor_len - 2, -1):
      if i >= len(remainder) :
          continue

      if i >= len(remainder):
          continue

      if int(remainder[i]) > int(divisor[divisor_len-1]):
          quotient[i - divisor_len] = base - 1
      else:
          if i - 1 < 0:
              if int(divisor[divisor_len - 1]) != 0:
                  quotient[i - divisor_len] = math.floor(int(remainder[i]) / int(divisor[divisor_len - 1]))
              else:
                  quotient[i - divisor_len] = 0
          else:
            if int(divisor[divisor_len - 1]) != 0 :
                quotient[i - divisor_len] = math.floor((int(remainder[i]) * base + int(remainder[i-1])) / int(divisor[divisor_len - 1]))
            else:
                quotient[i - divisor_len] = 0
      # Уточнение частного
      if i - 2 < 0:
          if i - 1 < 0:
              continue
```

Рис. 4: division

Заключение

Эта лабораторная описывает функции для выполнения арифметических операций в системах счисления.