МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе № 1

по дисциплине «Информатика»

Тема: **Системы счисления**

Студент гр. 1305 Смирнов М.О.
Студент гр. 1305 Смирнов В.А.
Студент гр. 1305 Чибисов А.А.
Преподаватель _____Перязева Ю.В.

Санкт-Петербург

2021

Введение

Цель работы состоит в закреплении знаний основных принципов представления чисел в различных системах счисления, которые необходимы для понимания того, как числовые данные хранятся и обрабатываются в компьютере.

Для достижения цели необходимо выполнить следующие задачи:

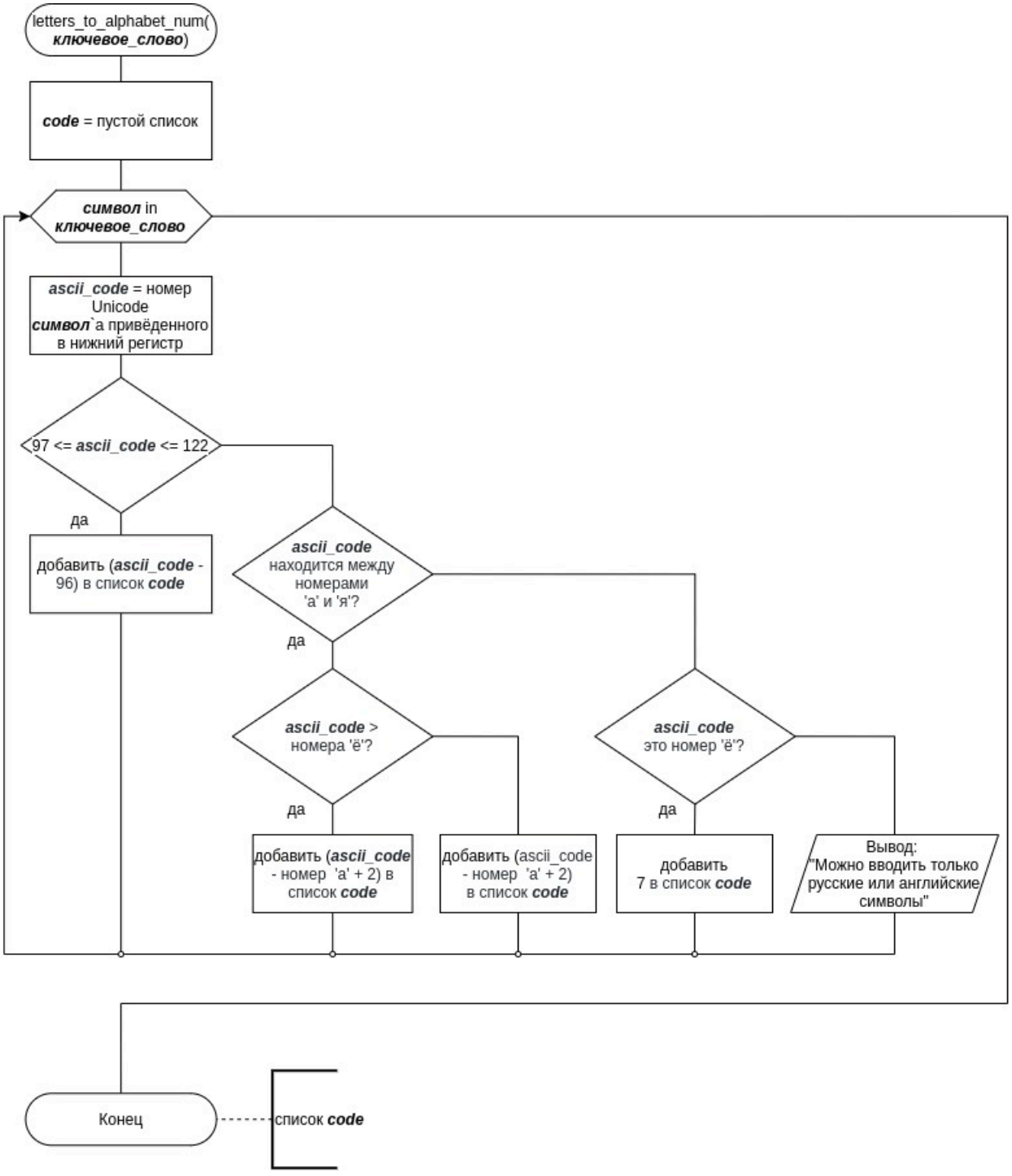
- изучить учебные материалы, посвященные системам счисления и хранению числовых данных в компьютере;
- разработать алгоритм по предложенному заданию, оформить в текстовом или графическом виде;
- разработать программу на языке программирования Python реализующую разработанный алгоритм;
- разработать контрольные примеры, выполнить их решение с помощью программы и ручной расчет, отладить программу;

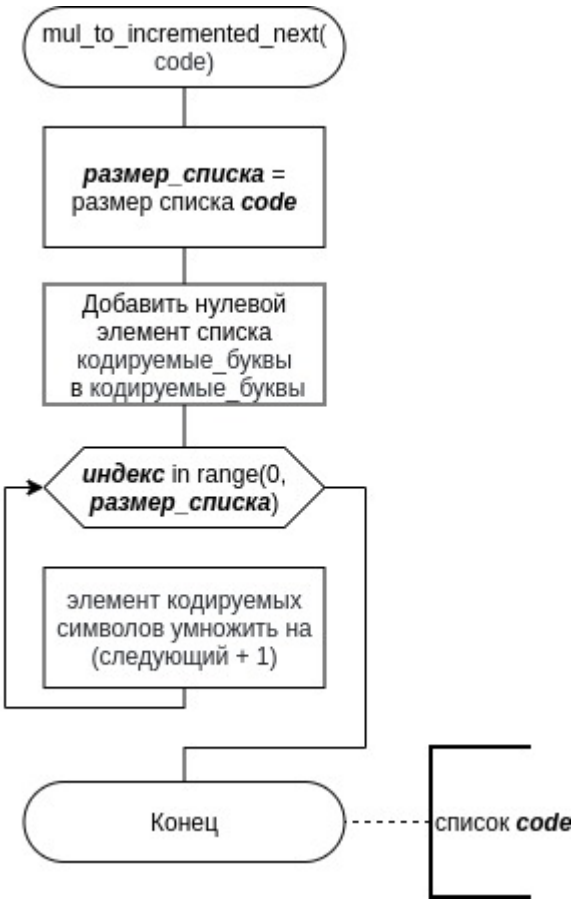
Постановка задачи и описание решения

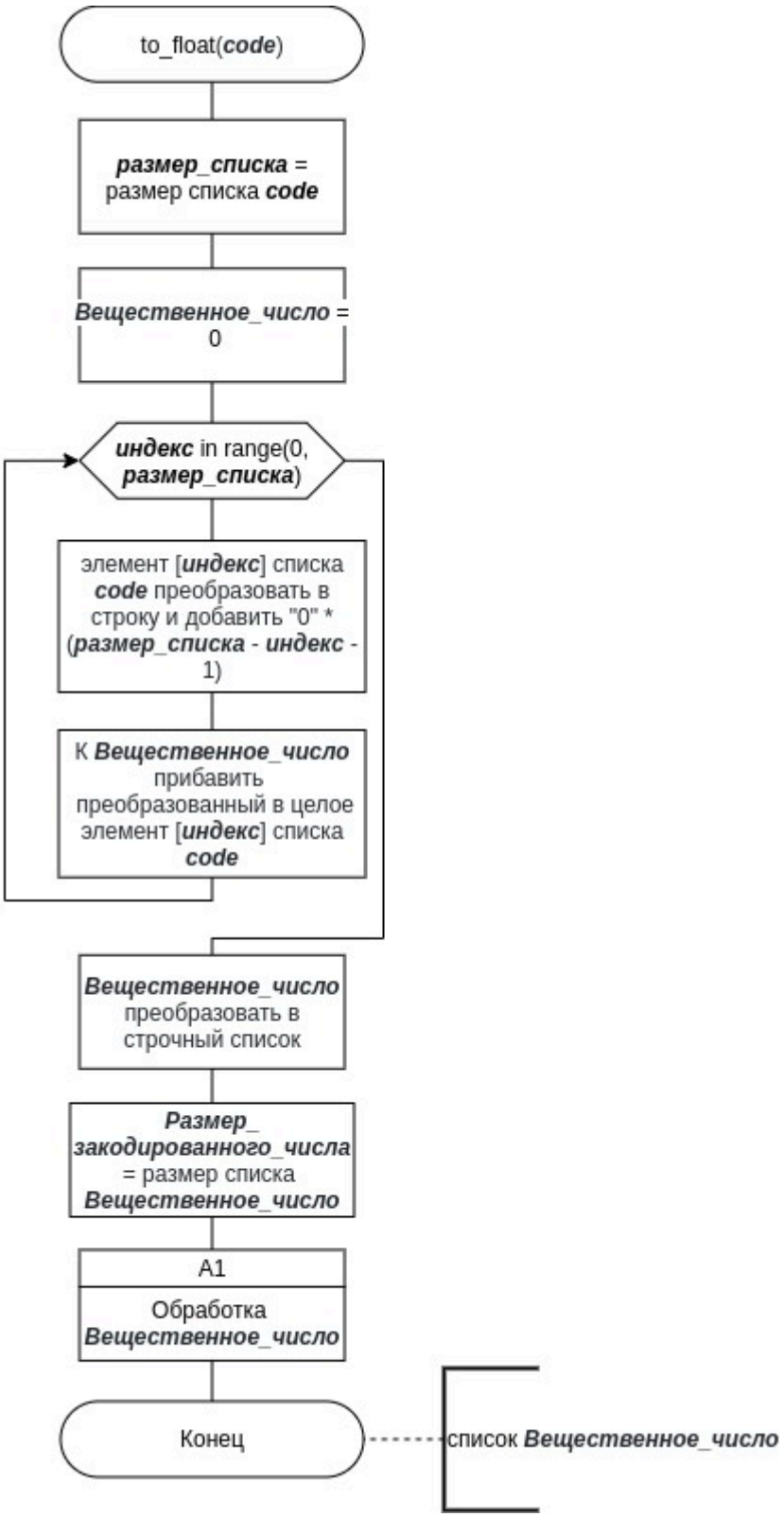
Необходимо разработать алгоритм формирования кода для кодового замка, который состоит из n символов (цифры, латинские буквы и символ #) по слову-ключу. Разработка алгоритма должна быть основана на теоретических положениях представления чисел в различных системах счисления.

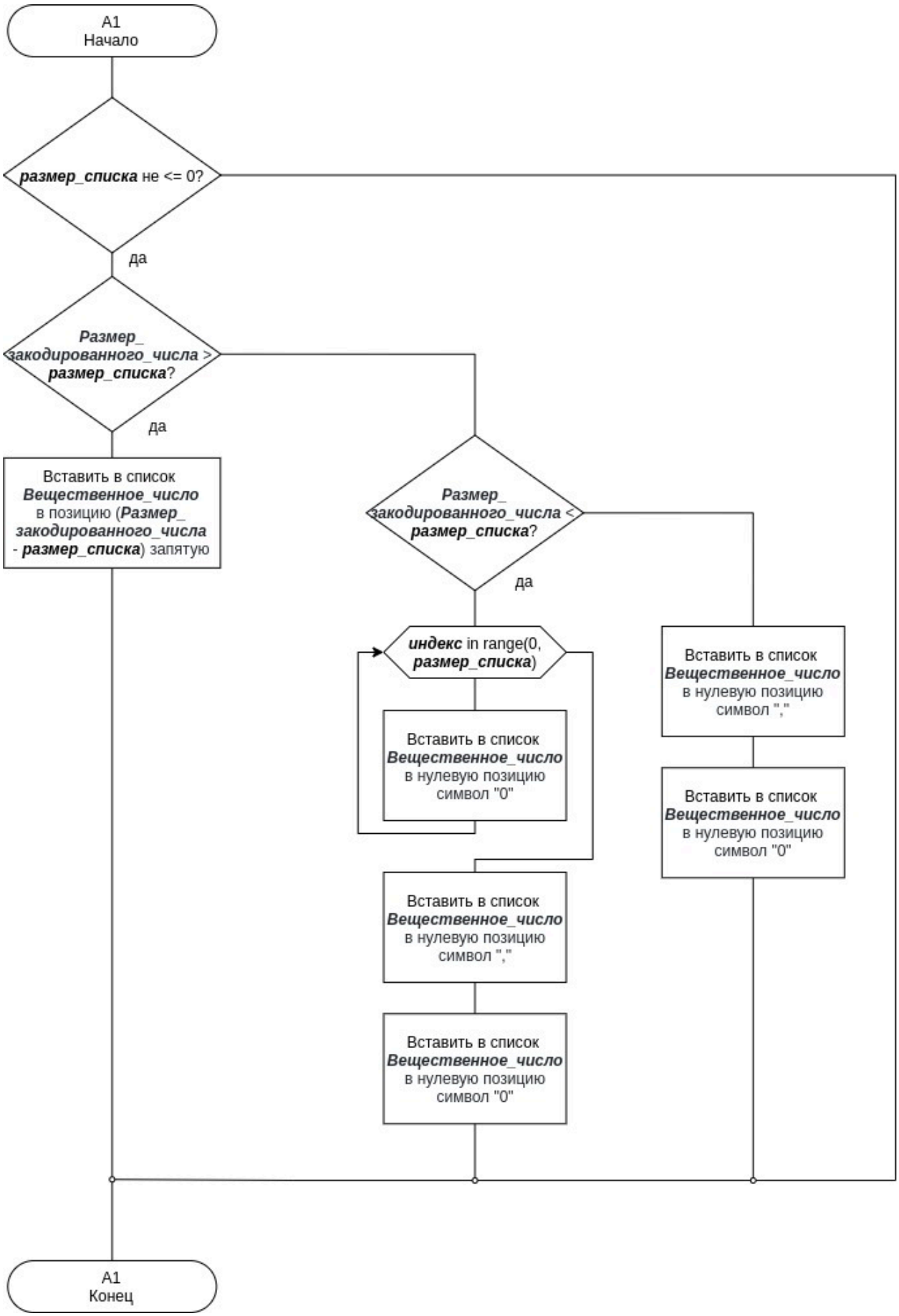
В ходе выполнения работы был разработан следующий алгоритм.

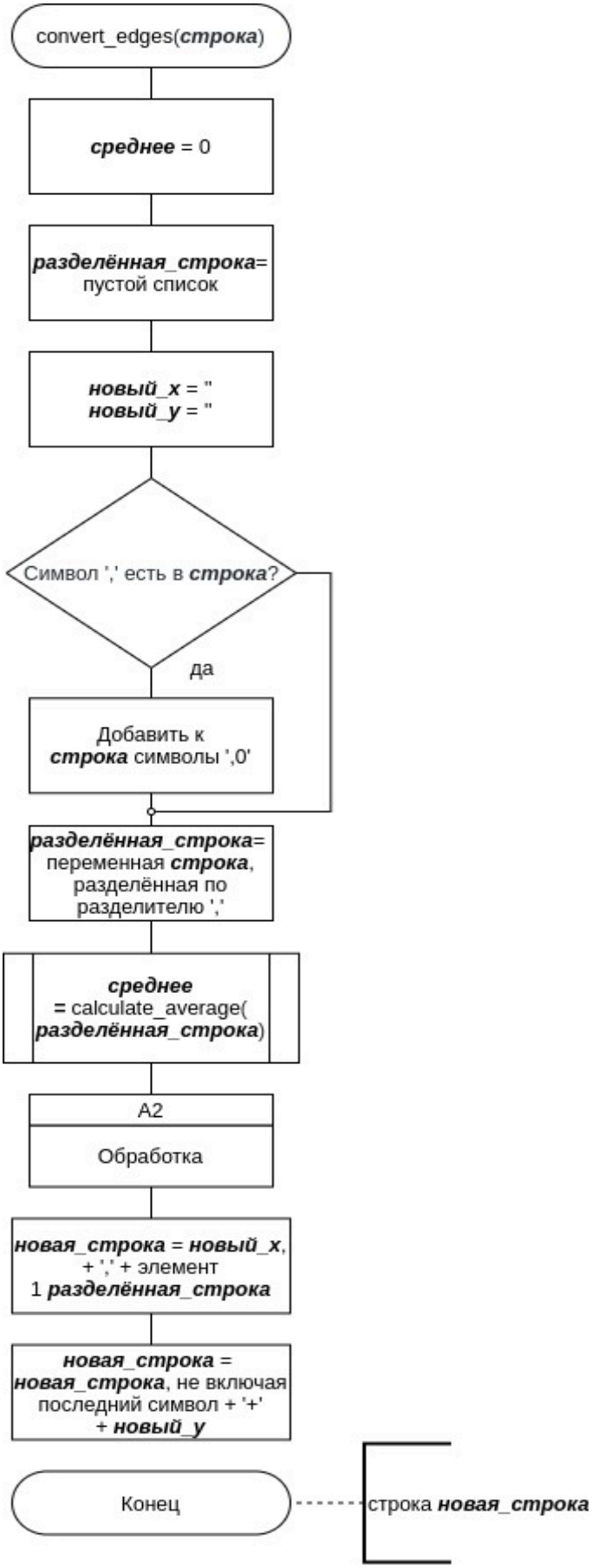

Начало
Ввод:
ключевое_слово
кодируемые_буквы =
letters_to_alphabet_num(
ключевое_слово)
кодируемые_буквы =
mul_to_incremented_next(кодируемые
буквы)
закодированное_число =
to_float(закодированное число)
закодированное_число =
convert_edges(
соединённое закодированное_число)
целая_часть = целая
часть от
закодированное_число
центральная_часть=
часть цисел между
целой часть и
последней цифрой
последняя_часть =
последняя цифра
закодированное_число
закодированное_число =
целая_часть + "," +
dec_to_three(центральная_часть) +
"+" + последняя_часть
закодированное_число =
move_right_part(
закодированное_число)
заменить ',' на 'C', '-' на
'#', '+' на '1' в
закодированное_число
Вывод:
закодированное_число
Конец

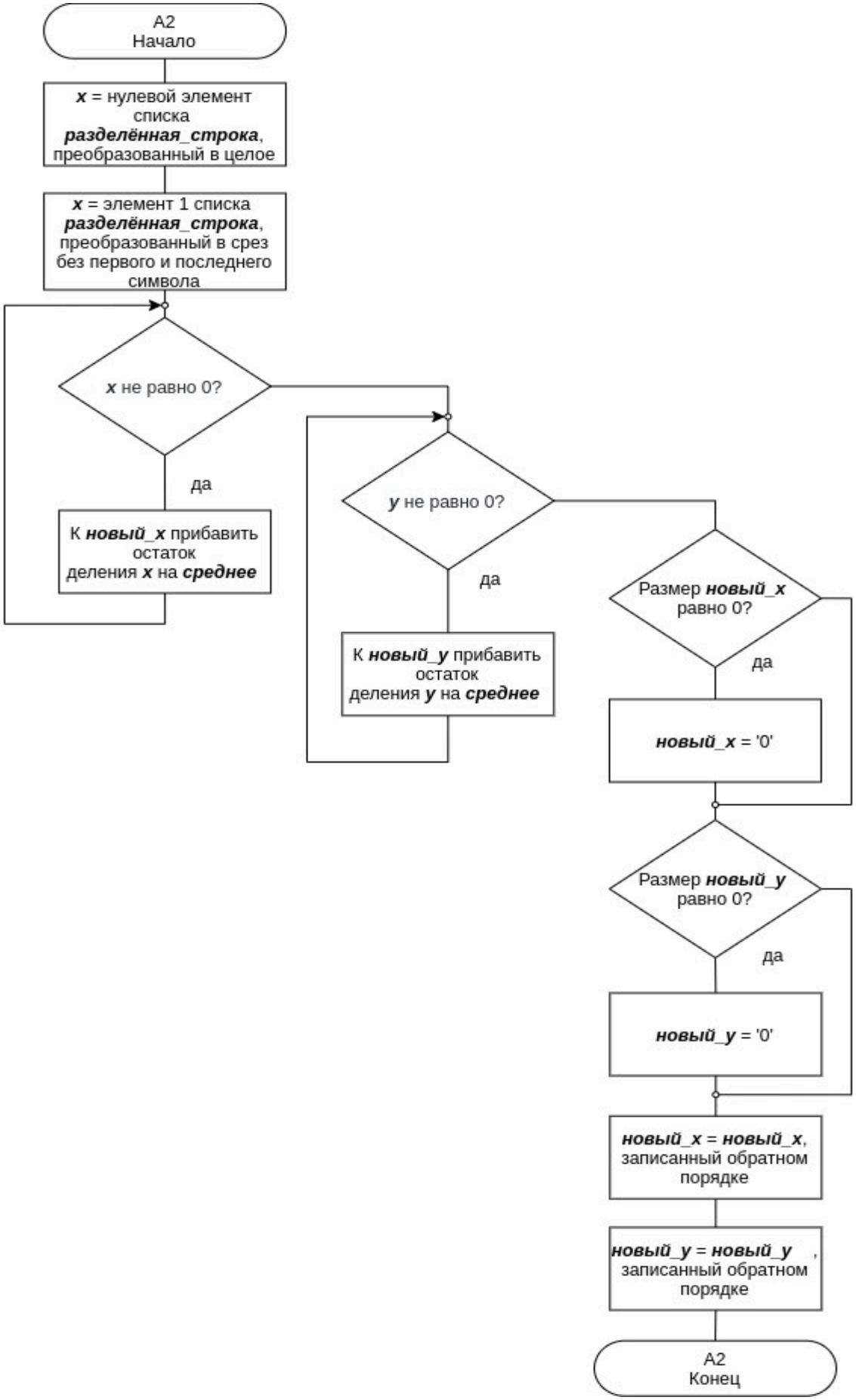

letters_to_alphabet_num(***ключевое_слово***)
code = пустой список
символ in ***ключевое_слово***
ascii_code = номер Unicode ***символ***`а привёденного в нижний регистр
97 <= ***ascii_code*** <= 122
да
добавить (***ascii_code*** - 96) в список ***code***
ascii_code находится между номерами 'а' и 'я'?
да
ascii_code > номера 'ё'?
да
добавить (***ascii_code*** - номер 'а' + 2) в список ***code***
добавить (ascii_code - номер 'а' + 2) в список ***code***
ascii_code это номер 'ё'?
да
добавить 7 в список ***code***
Вывод: "Можно вводить только русские или английские символы"
Конец
список ***code***

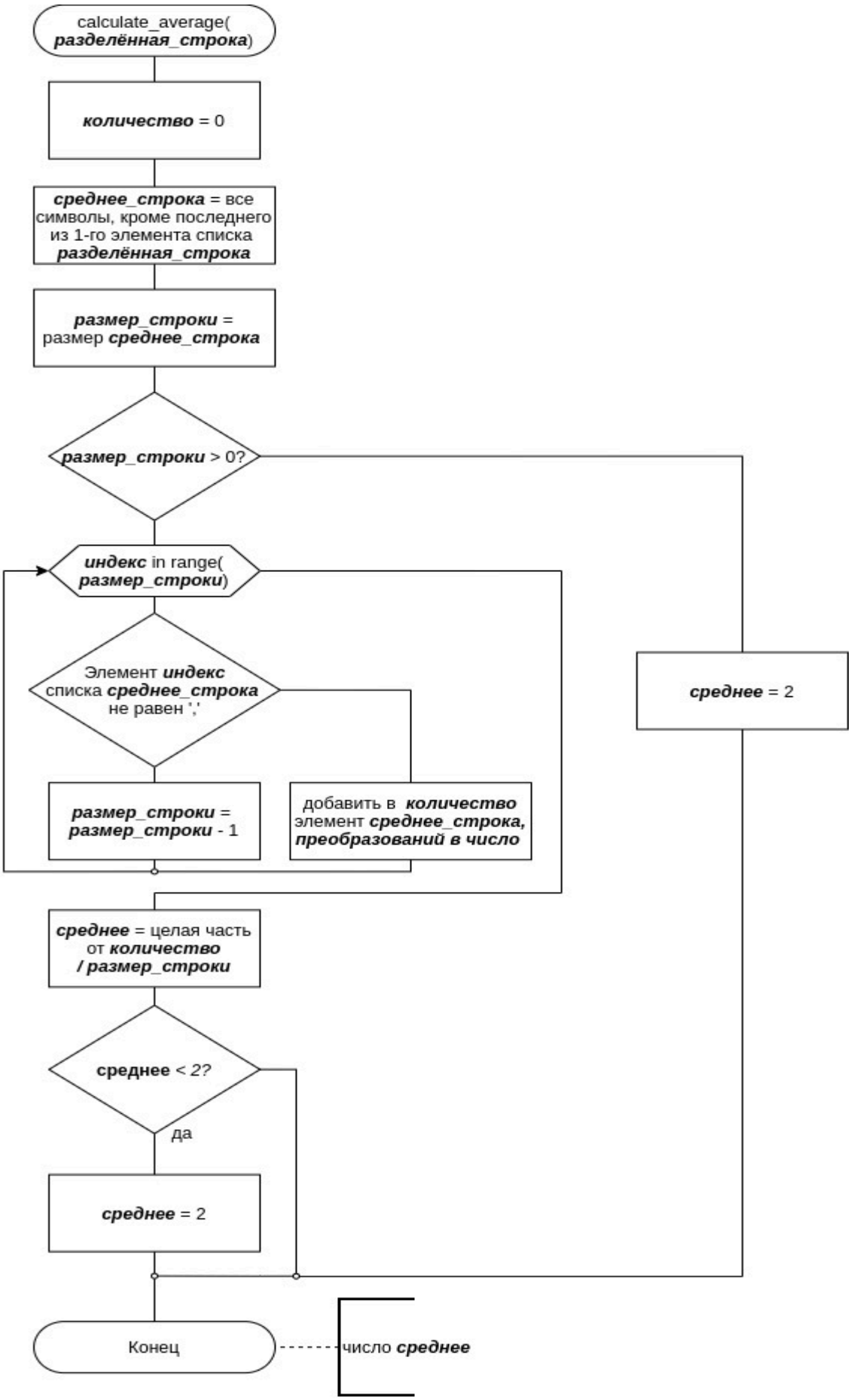

mul_to_incremented_next(
code)
размер_списка =
размер списка ***code***
Добавить нулевой
элемент списка
кодируемые_буквы
в кодируемые_буквы
индекс in range(0,
размер_списка)
элемент кодируемых
символов умножить на
(следующий + 1)
Конец
список ***code***

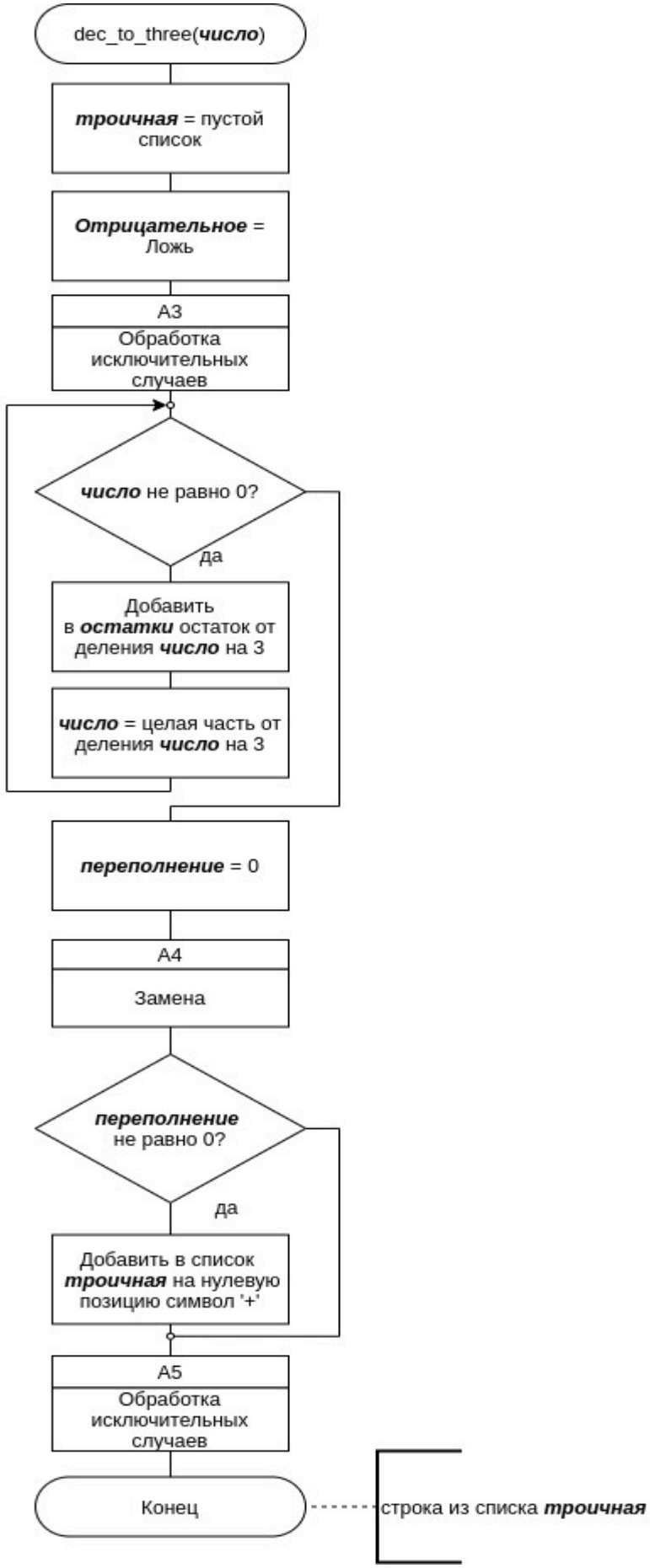

to_float(code)
размер_списка = размер списка code
Вещественное_число = 0
индекс in range(0, размер_списка)
элемент [индекс] списка code преобразовать в строку и добавить "0" * (размер_списка - индекс - 1)
К Вещественное_число прибавить преобразованный в целое элемент [индекс] списка code
Вещественное_число преобразовать в строчный список
Размер_закодированного_числа = размер списка Вещественное_число
A1
Обработка Вещественное_число
Конец
список Вещественное_число

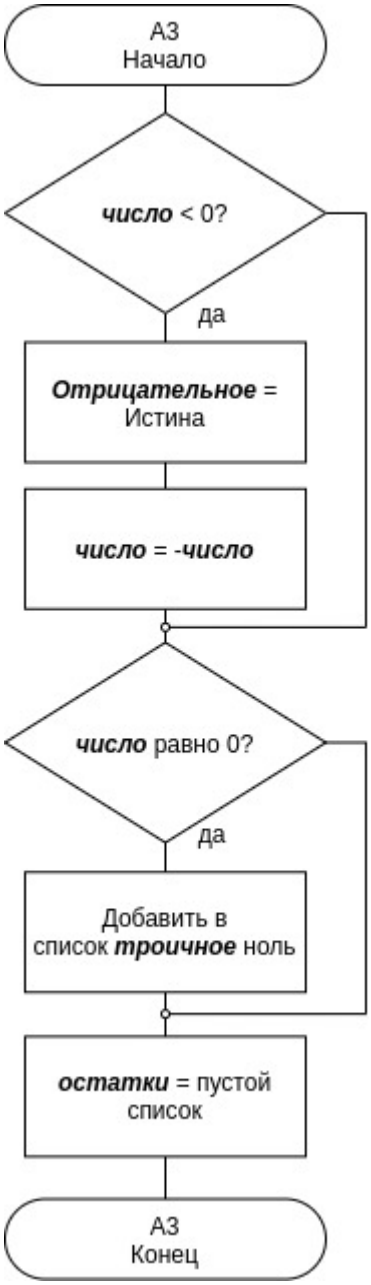

A1
Начало
размер_списка не <= 0?
да
Размер_ закодированного_числа > размер_списка?
да
Вставить в список Вещественное_число в позицию (Размер_ закодированного_числа - размер_списка) запятую
Размер_ закодированного_числа < размер_списка?
да
индекс in range(0, размер_списка)
Вставить в список Вещественное_число в нулевую позицию символ "0"
Вставить в список Вещественное_число в нулевую позицию символ ","
Вставить в список Вещественное_число в нулевую позицию символ ","
Вставить в список Вещественное_число в нулевую позицию символ "0"
Вставить в список Вещественное_число в нулевую позицию символ "0"
A1
Конец

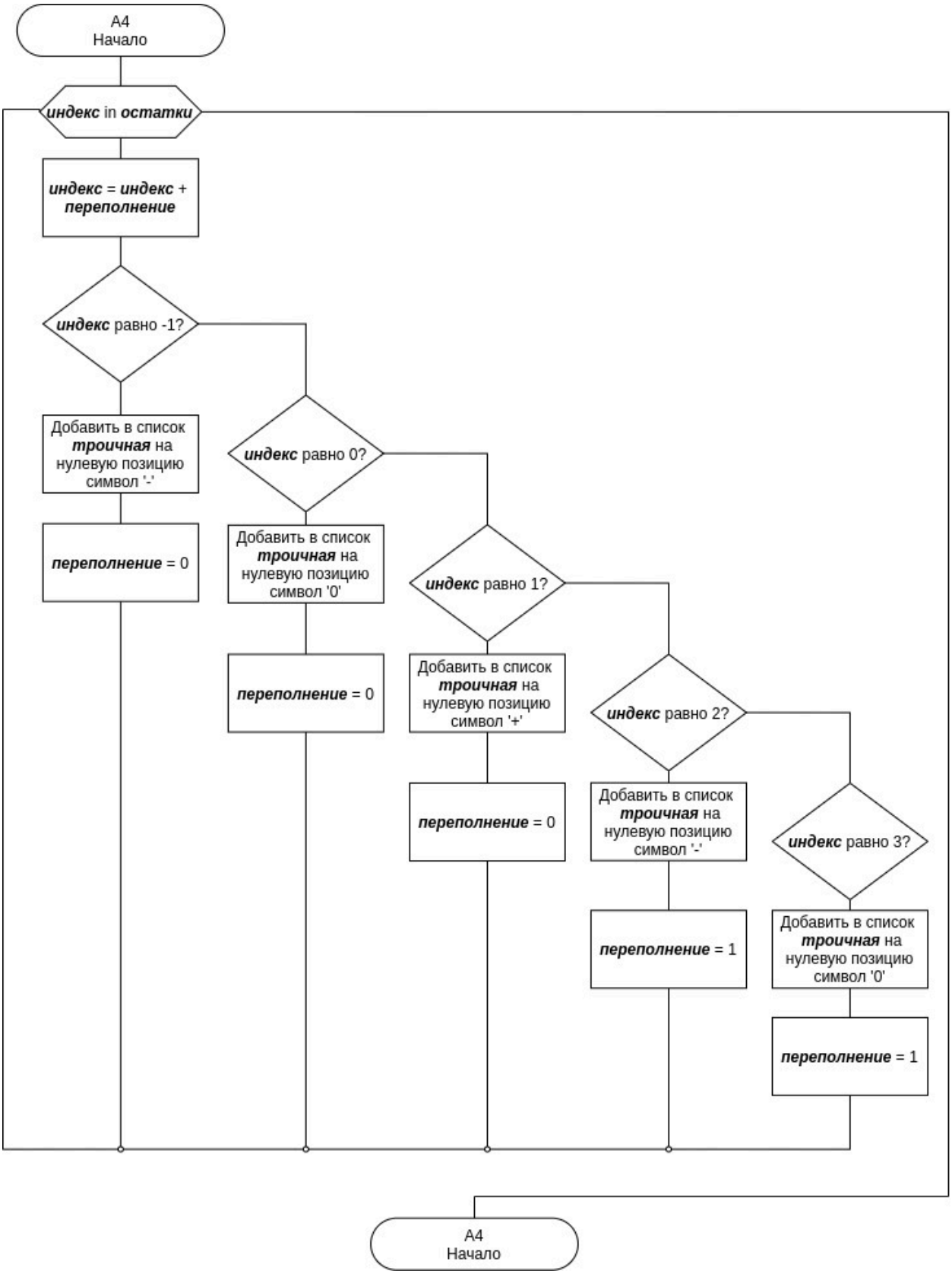

convert_edges(*строка*)
среднее = 0
разделённая_строка= пустой список
новый_x = ''
новый_y = ''
Символ ',' есть в *строка*?
да
Добавить к *строка* символы ',0'
разделённая_строка= переменная *строка*, разделённая по разделителю ','
среднее = calculate_average(*разделённая_строка*)
A2
Обработка
новая_строка = *новый_x*, + ',' + элемент 1 *разделённая_строка*
новая_строка = *новая_строка*, не включая последний символ + '+' + *новый_y*
Конец
строка *новая_строка*

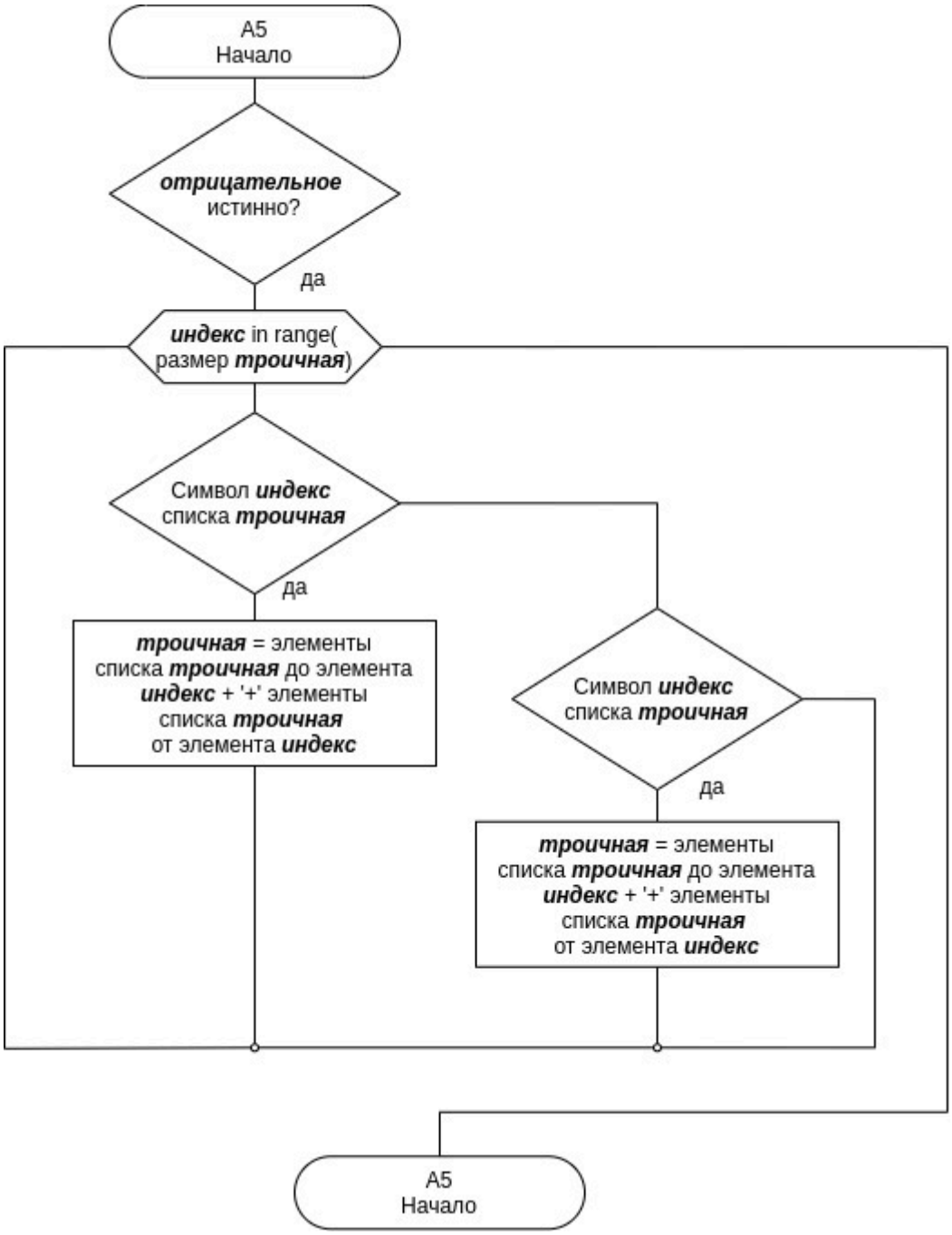

A2
Начало
x = нулевой элемент списка разделённая_строка, преобразованный в целое
x = элемент 1 списка разделённая_строка, преобразованный в срез без первого и последнего символа
x не равно 0?
да
К новый_x прибавить остаток деления x на среднее
y не равно 0?
да
К новый_y прибавить остаток деления y на среднее
Размер новый_x равно 0?
да
новый_x = '0'
Размер новый_y равно 0?
да
новый_y = '0'
новый_x = новый_x, записанный обратном порядке
новый_y = новый_y , записанный обратном порядке
A2
Конец

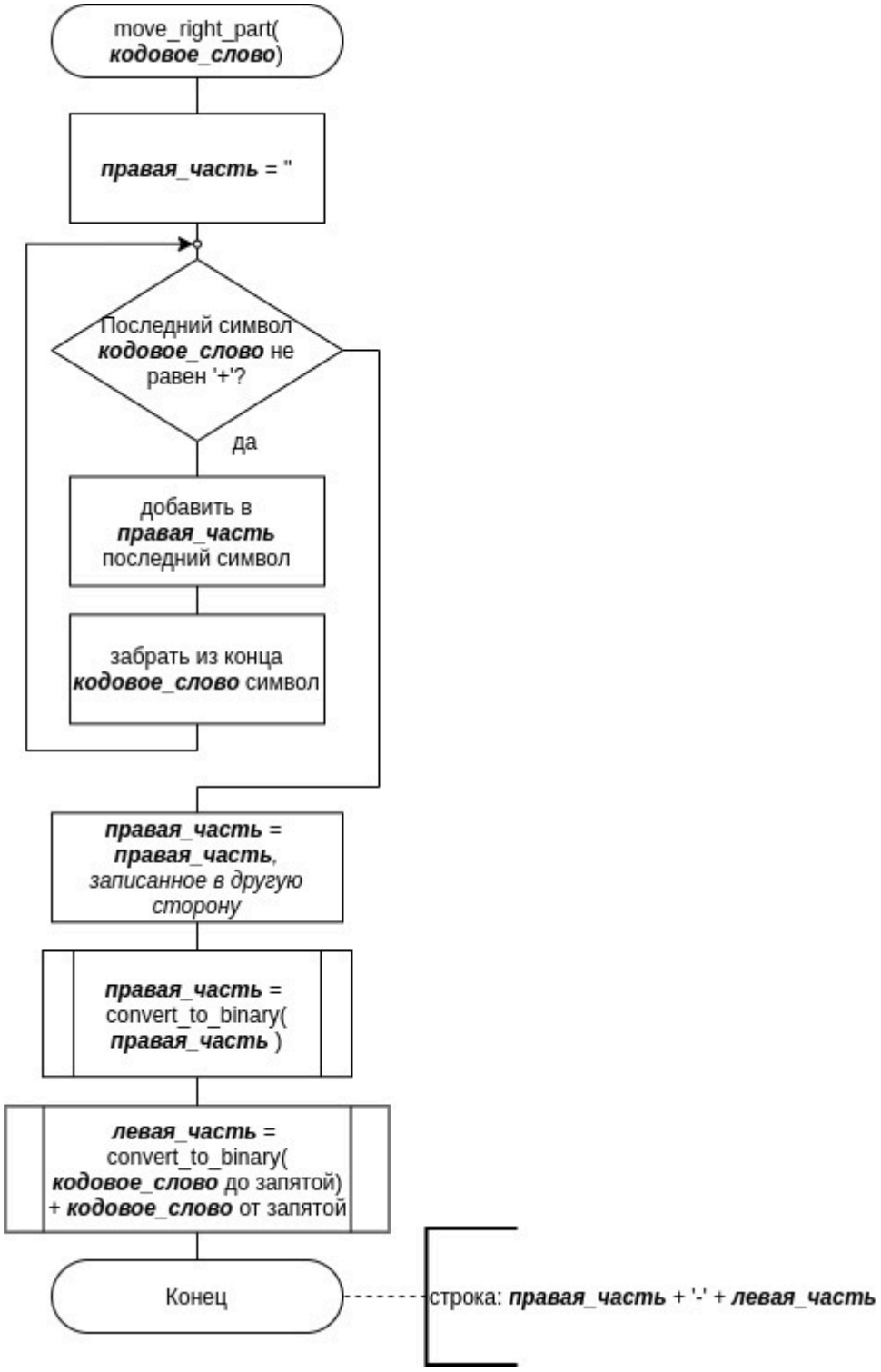

calculate_average(
разделённая_строка)
количество = 0
среднее_строка = все символы, кроме последнего из 1-го элемента списка разделённая_строка
размер_строки = размер среднее_строка
размер_строки > 0?
индекс in range(размер_строки)
Элемент индекс списка среднее_строка не равен ','
размер_строки = размер_строки - 1
добавить в количество элемент среднее_строка, преобразований в число
среднее = 2
среднее = целая часть от количество / размер_строки
среднее < 2?
да
среднее = 2
Конец
число среднее

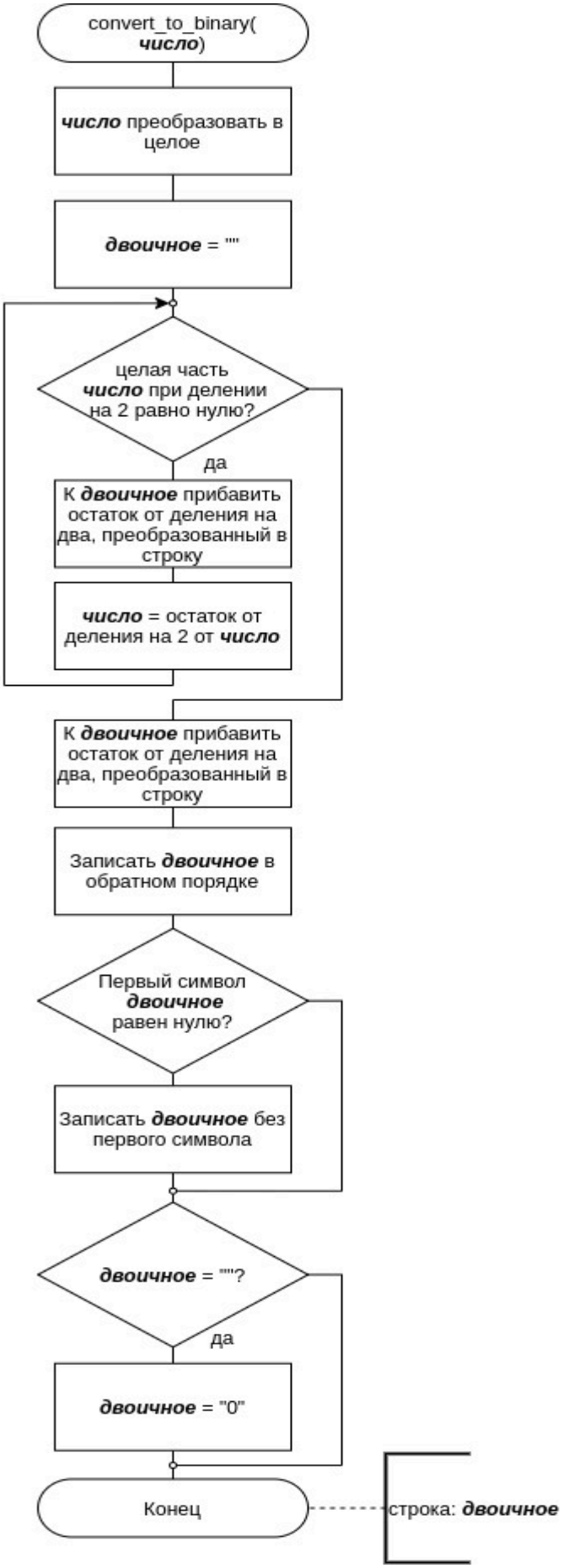

dec_to_three(*число*)
троичная = пустой список
Отрицательное = Ложь
A3
Обработка исключительных случаев
число не равно 0?
да
Добавить в *остатки* остаток от деления *число* на 3
число = целая часть от деления *число* на 3
переполнение = 0
A4
Замена
переполнение не равно 0?
да
Добавить в список *троичная* на нулевую позицию символ '+'
A5
Обработка исключительных случаев
Конец
строка из списка *троичная*


А3
Начало
число < 0?
да
Отрицательное =
Истина
число = -число
число равно 0?
да
Добавить в
список троичное ноль
остатки = пустой
список
А3
Конец


A4
Начало
индекс in остатки
индекс = индекс + переполнение
индекс равно -1?
Добавить в список троичная на нулевую позицию символ '-'
переполнение = 0
индекс равно 0?
Добавить в список троичная на нулевую позицию символ '0'
переполнение = 0
индекс равно 1?
Добавить в список троичная на нулевую позицию символ '+'
переполнение = 0
индекс равно 2?
Добавить в список троичная на нулевую позицию символ '-'
переполнение = 1
индекс равно 3?
Добавить в список троичная на нулевую позицию символ '0'
переполнение = 1
A4
Начало


A5
Начало
отрицательное
истинно?
да
индекс in range(
размер троичная)
Символ индекс
списка троичная
да
троичная = элементы
списка троичная до элемента
индекс + '+' элементы
списка троичная
от элемента индекс
Символ индекс
списка троичная
да
троичная = элементы
списка троичная до элемента
индекс + '+' элементы
списка троичная
от элемента индекс
A5
Начало


move_right_part(
кодовое_слово)
правая_часть = ''
Последний символ **кодовое_слово** не равен '+'?
да
добавить в **правая_часть** последний символ
забрать из конца **кодовое_слово** символ
правая_часть = **правая_часть**, *записанное в другую сторону*
правая_часть = convert_to_binary(**правая_часть**)
левая_часть = convert_to_binary(**кодовое_слово** до запятой) + **кодовое_слово** от запятой
Конец
строка: **правая_часть** + '-' + **левая_часть**


convert_to_binary(
число)
число преобразовать в целое
двоичное = ""
целая часть ***число*** при делении на 2 равно нулю?
да
К ***двоичное*** прибавить остаток от деления на два, преобразованный в строку
число = остаток от деления на 2 от ***число***
К ***двоичное*** прибавить остаток от деления на два, преобразованный в строку
Записать ***двоичное*** в обратном порядке
Первый символ ***двоичное*** равен нулю?
Записать ***двоичное*** без первого символа
двоичное = ""?
да
двоичное = "0"
Конец
строка: ***двоичное***

Примеры работы программы:

1) sUm => 0#11000101011101101C01

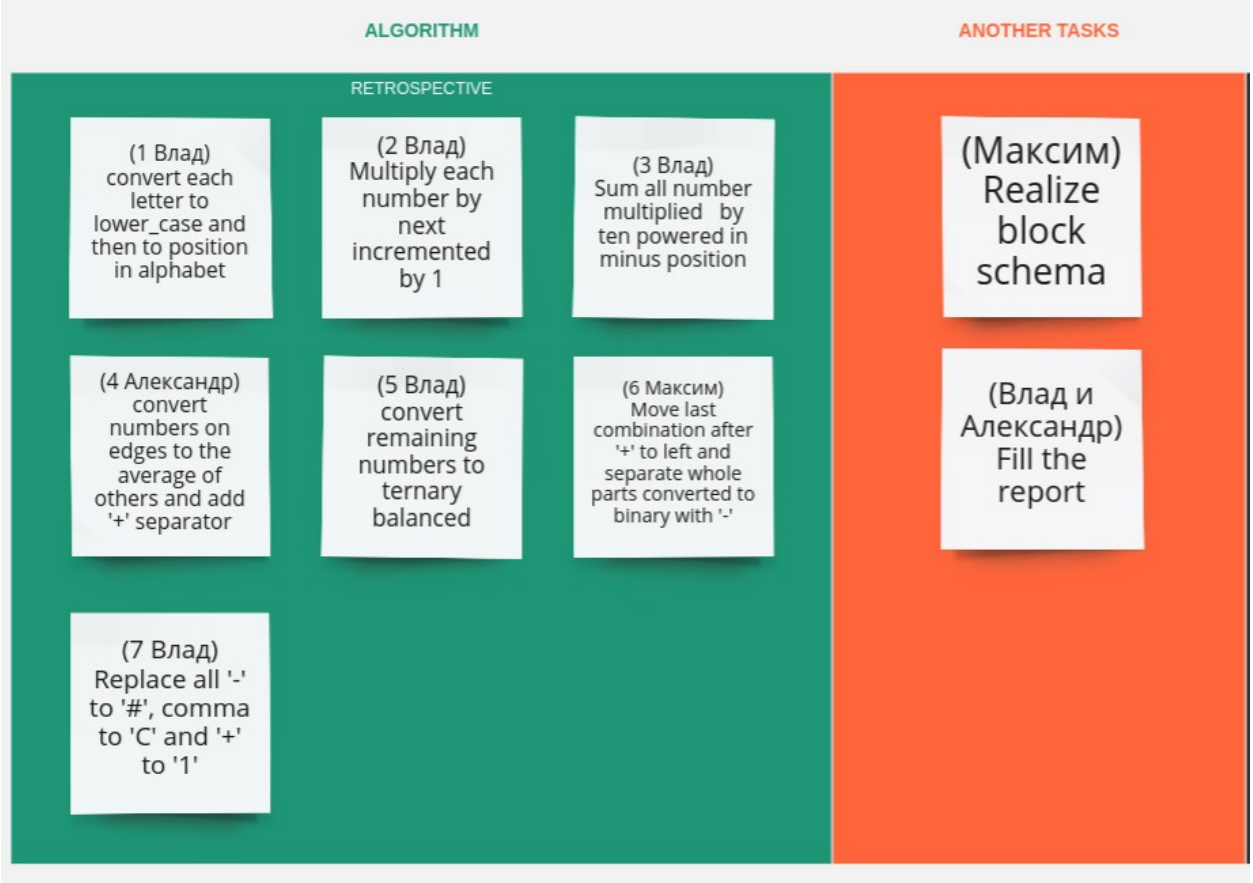
2) retroSpeCtIVe => 1100#1101110C10#101###11111#110##011##1

3) Мёд => 101#1110C10##01

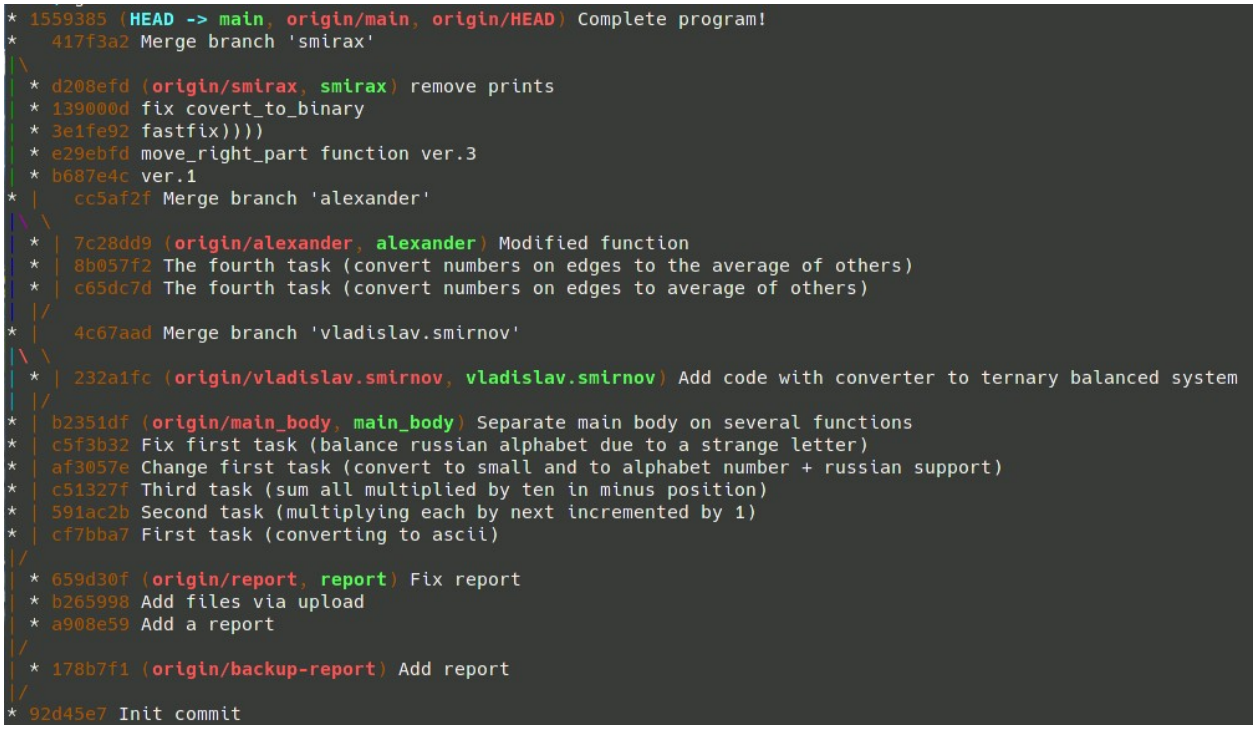
4) КалейДоскоП => 1#10C1##1111100###100#010001

Алгоритм реализован на Python 3, код представлен в приложении 1.

Скриншот данной kanban доски демонстрирует какие задачи были поставлены в ходе разработки алгоритма, а также подписаны исполнители.



Разработка велась при помощи системы контроля версий git, а исходный код лежит на github по следующему адресу: https://github.com/InformaticsLaboratory/Lab.



История разработки показана на данном скриншоте:

Заключение

Разработан алгоритм формирования кода для кодового замка, состоящего из n символов (цифры, латинские буквы и символ #), по слову ключу. Разработка алгоритма основана на теоретических положениях представления чисел в различных системах счисления. При разработке алгоритма было использовано два изученных алгоритма и один новый — перевод в троичную уравновешенную систему. Алгоритм оформлен графически (блоксхемы).

Были выполнены следующие задачи:

- Изучение учебных материалов, посвящённых системам счисления и хранению числовых данных в компьютере;

- Разработка алгоритма к заданию, оформление в графическом виде;
- Разработка программы на языке программирования Python, реализующей разработанный алгоритм;
- Разработка контрольных примеров и их решение с помощью программы и ручного расчёта, а также отладка программы;
- Подготовка отчёта.

В ходе работы были закреплены знания основных принципов представления чисел в различных системах счисления, которые необходимы для понимания того, как числовые данные хранятся и обрабатываются в компьютере.

Список использованных источников

Идея для перевода десятичного числа в троичную сбалансированную систему взята из данного источника: https://qna.habr.com/q/378857.

Приложение 1

```
def letters_to_alphabet_num(keyword):
    code = list()
    for symbol in keyword:
        ascii_code = ord(symbol.lower())
        if 97 <= ascii_code <= 122:
            code.append(ascii_code - 96)
        elif ord('a') <= ascii_code <= ord('я'):
            if ascii_code > ord('e'):
                code.append(ascii_code - ord('a') + 2)
            else:
                code.append(ascii_code - ord('a') + 1)
        elif ascii_code == ord('ё'):
            code.append(7)
```

```
        else:
            print("You can input only russian or english letters!")
            exit(0)

    return code


def mul_to_incremented_next(code):
    list_size = len(code)
    coded_letters.append(coded_letters[0])
    for index in range(0, list_size):
        coded_letters[index] = coded_letters[index] * (coded_letters[index + 1] + 1)

    return coded_letters[0:list_size]


def to_float(code):
    list_size = len(code)
    float_number = 0
    for index in range(0, list_size):
        code[index] = str(code[index]) + "0" * (list_size - index - 1)
        float_number += int(code[index])

    float_number = list(str(float_number))
    encoded_number_size = len(float_number)
    if not list_size <= 0:
        if encoded_number_size > list_size:
```

```
            float_number.insert(encoded_number_size - list_size, ",")
        elif encoded_number_size < list_size:
            for i in range(0, list_size - encoded_number_size):
                float_number.insert(0, "0")
            float_number.insert(0, ",")
            float_number.insert(0, "0")
        else:
            float_number.insert(0, ",")
            float_number.insert(0, "0")

    return float_number


def convert_edges(string):
    average = 0
    spl_str = []
    new_x = ''
    new_y = ''

    if not(',' in string):
        string += ',0'

    spl_str = string.split(',')

    average = calculate_average(spl_str)

    x = int(spl_str[0])
```

```
    y = int(spl_str[1][-1:-2:-1])

    while x != 0:
        new_x += str(x % average)
        x //= average

    while y != 0:
        new_y += str(y % average)
        y //= average

    if len(new_x) == 0:
        new_x = '0'
    if len(new_y) == 0:
        new_y = '0'

    new_x = new_x[::-1]
    new_y = new_y[::-1]
    new_string = new_x + ',' + spl_str[1]
    new_string = new_string[0:-1] + '+' + new_y

    return new_string


def calculate_average(spl_str):
    amount = 0
    average_str = spl_str[1][0:-1]
    len_str = len(average_str)
```

```
    if len_str > 0:
        for i in range(len_str):
            if average_str[i] != ',':
                amount += int(average_str[i])
            else:
                len_str -= 1
        average = int(amount // len_str)
        if average < 2:
            return 2
    else:
        return 2

    return average


def dec_to_three(n):
    orientated = list()
    is_negative = False
    if n < 0:
        is_negative = True
        n = -n
    if n == 0:
        orientated.append(0)

    remainders = list()
```

```
while n != 0:
    remainders.append(n % 3)
    n //= 3

overflow = 0
for i in remainders:
    i = i + overflow
    if i == -1:
        orientated.insert(0, '-')
        overflow = 0
    elif i == 0:
        orientated.insert(0, 0)
        overflow = 0
    elif i == 1:
        orientated.insert(0, '+')
        overflow = 0
    elif i == 2:
        orientated.insert(0, '-')
        overflow = 1
    elif i == 3:
        orientated.insert(0, 0)
        overflow = 1

if overflow != 0:
    orientated.insert(0, '+')

if is_negative:
```

```
        for i in range(0, len(orientated)):
            if orientated[i] == '-':
                orientated = orientated[0:i] + list('+') + orientated[i + 1:]
            elif orientated[i] == '+':
                orientated = orientated[0:i] + list('-') + orientated[i + 1:]

    return "".join(map(str, orientated))



def move_right_part(code_word):
    code_word = str(code_word)
    right_part = ""
    while code_word[-1] != "+":
        right_part += code_word[-1]
        code_word = code_word[:-1]
    right_part = right_part[::-1]

    right_part = convert_to_binary(right_part)

    left_part = convert_to_binary(code_word[:code_word.find(',')]) +
(code_word[code_word.find(','):])

    return right_part + '-' + left_part



def convert_to_binary(number):
    number = int(number)
```

```
    binary = ""
    while number // 2 != 0:
        binary += str(number % 2)
        number //= 2
    binary += str(number % 2)

    binary = binary[::-1]

    if binary[0] == '0':
        binary = binary[1:]
    if binary == '':
        binary = "0"
    return binary



print("Enter the keyword: ", end='')
user_keyword = input()

coded_letters = letters_to_alphabet_num(user_keyword)
coded_letters = mul_to_incremented_next(coded_letters)
encoded_number = to_float(coded_letters)
encoded_number = convert_edges(''.join(encoded_number))

whole_part = encoded_number.split(',')[0]
middle_part = encoded_number.split(',')[1].split('+')[0]
end_part = encoded_number.split(',')[1].split('+')[1]
```

```
encoded_number = whole_part + "," + dec_to_three(int(middle_part)) + "+" +
end_part


encoded_number = move_right_part(encoded_number)

encoded_number = encoded_number.replace(',', 'C').replace('-', '#').replace('+', '1')

print("The result code: " + encoded_number)
```