

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
По практической работе №1
Дисциплины «Алгоритмизация»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и
вычислительная техника (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических
наук, доцент, доцент кафедры
инфокоммуникаций

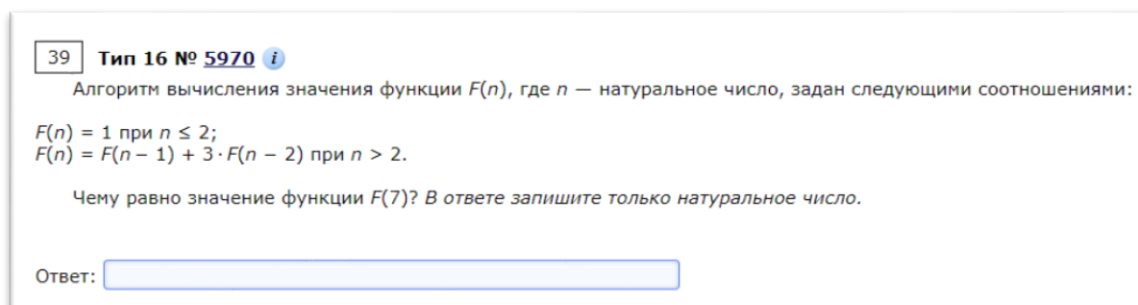
(подпись)

Ставрополь, 2023 г.

Ход работы:

Задание 16.

Необходимо найти значения функции $F(n)$ где n – натуральное число (рис. 1).



39 Тип 16 № 5970 *i*

Алгоритм вычисления значения функции $F(n)$, где n – натуральное число, задан следующими соотношениями:

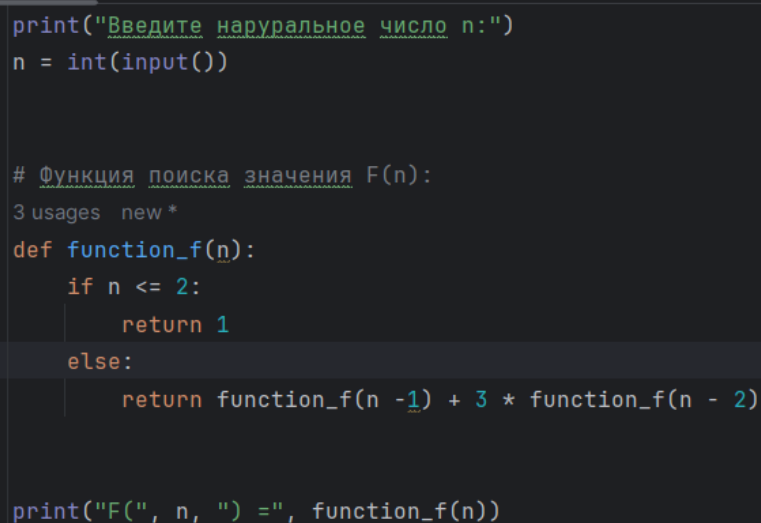
$$F(n) = 1 \text{ при } n \leq 2;$$
$$F(n) = F(n-1) + 3 \cdot F(n-2) \text{ при } n > 2.$$

Чему равно значение функции $F(7)$? В ответе запишите только натуральное число.

Ответ:

Рисунок 1 – Условия задачи № 16 (5970).

Код программы на языке Python данной задачи (рис. 2).



```
print("Введите натуральное число n:")
n = int(input())

# Функция поиска значения F(n):
3 usages new *
def function_f(n):
    if n <= 2:
        return 1
    else:
        return function_f(n - 1) + 3 * function_f(n - 2)

print("F(", n, ") =", function_f(n))
```

Рисунок 2 – Код программы решения задачи №16.

Результаты работы программы (ввод значения «7» по условию) (рис. 3).

```

C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\
Введите натуральное число n:
7
F( 7 ) = 97

Process finished with exit code 0

```

Рисунок 3 – Результат работы программы.

Блок-схема алгоритма работы программы и алгоритма функции (рис. 4).

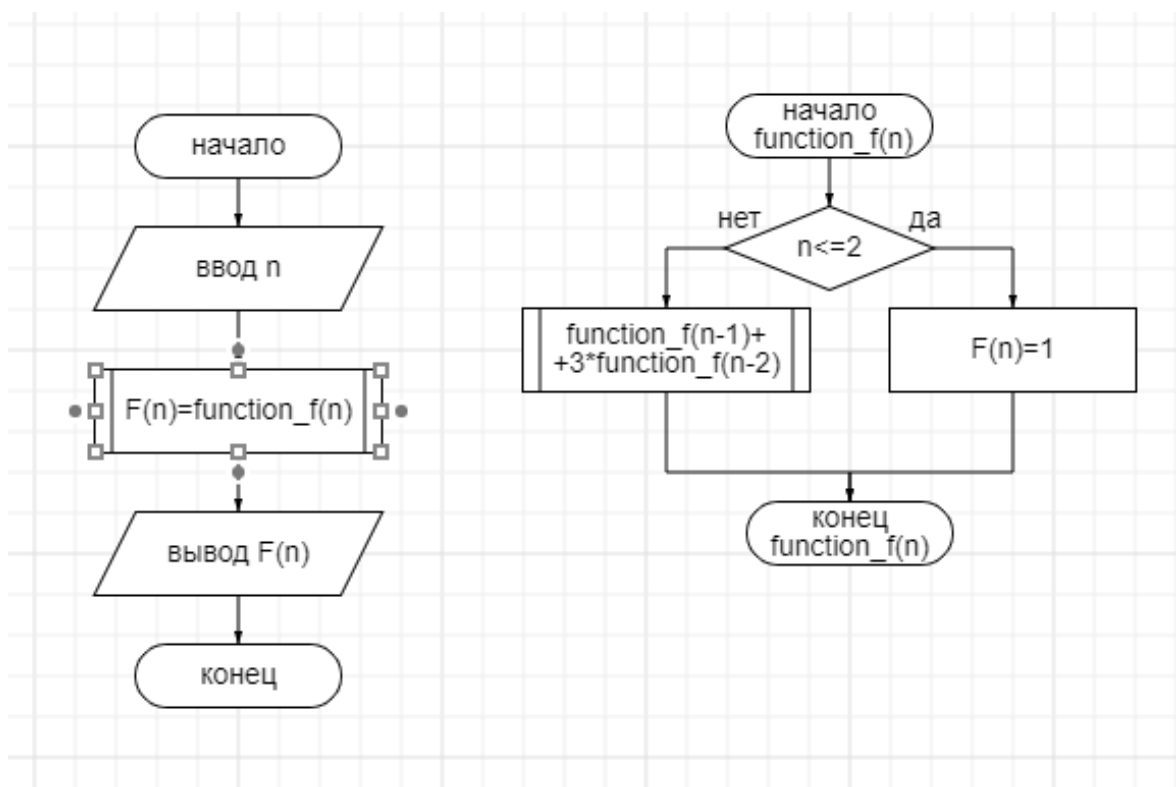


Рисунок 4 – Блок-схема алгоритма.

Задание 17.

Необходимо найти количество троек элементов последовательности в текстовом файле, удовлетворяющих некоторому условию и найти максимальную из сумм таких троек (рис. 5).

Тип 17 № 60259

В файле содержится последовательность натуральных чисел, каждое из которых не превышает 100 000. Определите количество троек элементов последовательности, в которых ровно два из трёх элементов являются трёхзначными числами, а сумма элементов тройки не больше максимального элемента последовательности, оканчивающегося на 13. Гарантируется, что в последовательности есть хотя бы одно число, оканчивающееся на 13. В ответе запишите количество найденных троек чисел, затем максимальную из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Рисунок 5 – Условия задачи № 17 (60259).

Код программы данной задачи (рис. 6).

```
count = 0
max_element = 0
max_sum = 0

# Открытие файла
f = open('17_2024.txt')

# Чтение файла
s = f.readlines()

# Разбор по строкам:
for i in range(len(s)):
    s[i] = int(s[i])

# Определение максимального элемента, оканчивающегося на 13:
for i in range(len(s)):
    if s[i] % 100 == 13:
        max_element = max(max_element, s[i])

# Определение кол-ва троек чисел и максимальной из сумм:
for i in range(2, len(s)):
    bit_depth_1 = 0
    bit_depth_2 = 0
    bit_depth_3 = 0
    number_module_1 = s[i - 2]
    number_module_2 = s[i - 1]
    number_module_3 = s[i]
    while number_module_1 > 0:
        bit_depth_1 = bit_depth_1 + 1
        number_module_1 = number_module_1 // 10
    while number_module_2 > 0:
        bit_depth_2 = bit_depth_2 + 1
        number_module_2 = number_module_2 // 10
    while number_module_3 > 0:
        bit_depth_3 = bit_depth_3 + 1
        number_module_3 = number_module_3 // 10
    if ((bit_depth_1 == 3 and bit_depth_2 == 3 and bit_depth_3 != 3)
        or (bit_depth_1 == 3 and bit_depth_2 != 3 and bit_depth_3 ==
3)
        or (bit_depth_1 != 3 and bit_depth_2 == 3 and bit_depth_3 ==
3)):
        if s[i - 2] + s[i - 1] + s[i] <= max_element:
            count = count + 1
            max_sum = max(max_sum, s[i - 2] + s[i - 1] + s[i])

# Закрытие файла
f.close()
```


из первого элемента двумерного списка в последний. Робот может переходить либо из верхней клетки в нижнюю или вправо (робот не может проходить через внутренние стены) (рис. 9).

39

Тип 18 № 45252

Квадрат разбит на $N \times N$ клеток ($1 < N < 30$). Исполнитель Робот может перемещаться по клеткам, выполняя за одно перемещение одну из двух команд: вправо или вниз. По команде вправо Робот перемещается в соседнюю правую клетку, по команде **вниз** — в соседнюю нижнюю. Квадрат ограничен внешними стенами. Между соседними клетками квадрата также могут быть внутренние стены. Сквозь стену Робот пройти не может.

Перед каждым запуском Робота в каждой клетке квадрата лежит монета достоинством от 1 до 100. Посетив клетку, Робот забирает монету с собой; это также относится к начальной и конечной клеткам маршрута Робота.

Определите максимальную и минимальную денежные суммы, которые может собрать Робот, пройдя из левой верхней клетки в правую нижнюю. В ответе укажите два числа — сначала максимальную сумму, затем минимальную.

[18.xlsx](#)

Исходные данные представляют собой электронную таблицу размером $N \times N$, каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщенными линиями.

Пример входных данных:

Рисунок 9 – Условия задачи №18.

Таблица с номиналами монет и стенками для робота-сборщика монет (рис. 10).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	3	36	74	75	85	44	68	39	45	42	35	39	75	46	52	39	46	37	55	25
2	35	41	35	55	46	25	85	72	88	65	48	69	57	86	86	54	37	68	62	78
3	85	67	27	51	44	85	62	80	27	68	40	35	34	84	57	67	79	74	52	72
4	83	33	38	90	79	80	77	49	84	35	57	68	26	44	25	53	75	84	70	80
5	36	67	33	72	27	60	63	81	37	69	80	25	36	50	48	80	72	32	31	46
6	37	55	64	66	73	26	75	85	74	38	58	57	61	25	66	59	84	39	47	63
7	54	84	38	80	52	32	57	83	68	62	51	68	57	90	27	55	38	27	52	61
8	81	55	74	75	26	78	83	34	46	90	74	67	54	88	33	70	76	35	56	62
9	59	63	90	26	31	83	63	46	70	47	58	72	58	41	34	66	28	60	50	51
10	75	59	84	57	31	36	66	87	70	70	37	77	80	65	53	78	61	39	25	76
11	61	60	76	36	56	73	71	69	50	50	44	55	48	58	90	59	71	37	70	67
12	71	36	72	50	67	64	83	42	62	50	62	27	29	77	75	39	36	31	31	84
13	87	63	88	36	54	41	35	67	44	69	64	32	42	38	62	83	71	31	63	39
14	86	33	38	50	57	66	67	73	69	60	27	38	32	33	53	73	42	36	86	57
15	42	42	70	60	79	65	81	45	33	88	62	71	61	89	54	51	34	44	64	64
16	85	89	66	84	80	73	47	50	67	86	59	75	83	51	75	83	68	39	73	75
17	52	46	30	79	62	53	82	79	40	34	79	61	50	71	66	87	59	47	58	58
18	41	37	25	54	52	77	76	30	39	41	57	79	88	81	59	52	73	41	85	40
19	55	48	49	55	89	68	75	90	76	72	34	72	78	78	78	39	47	63	76	42
20	73	78	25	90	49	42	78	45	71	29	71	36	76	60	49	84	34	88	36	51

Рисунок 10 – Таблица к задаче №18.

Код программы задачи №18 (рис. 11).

```
import copy
# Задание списка чисел по условию:
original_list = [[3, 36, 74, 75, 85, 44, 68, 39, 45, 42,
35, 39, 75, 46, 52, 39, 46, 37, 55, 25],
[35, 41, 35, 55, 46, 25, 85, 72, 88, 65,
48, 69, 57, 86, 86, 54, 37, 68, 62, 78],
[85, 67, 27, 51, 44, 85, 62, 80, 27, 68,
40, 35, 34, 84, 57, 67, 79, 74, 52, 72],
[83, 33, 38, 90, 79, 80, 77, 49, 84, 35,
57, 68, 26, 44, 25, 53, 75, 84, 70, 80],
[36, 67, 33, 72, 27, 60, 63, 81, 37, 69,
80, 25, 36, 50, 48, 80, 72, 32, 31, 46],
[37, 55, 64, 66, 73, 26, 75, 85, 74, 38,
58, 57, 61, 25, 66, 59, 84, 39, 47, 63],
[54, 84, 38, 80, 52, 32, 57, 83, 68, 62,
51, 68, 57, 90, 27, 55, 38, 27, 52, 61],
```

```

[81, 55, 74, 75, 26, 78, 83, 34, 46, 90,
 74, 67, 54, 88, 33, 70, 76, 35, 56, 62],
[59, 63, 90, 26, 31, 83, 63, 46, 70, 47,
 58, 72, 58, 41, 34, 66, 28, 60, 50, 51],
[75, 59, 84, 57, 31, 36, 66, 87, 70, 70,
 37, 77, 80, 65, 53, 78, 61, 39, 25, 76],
[61, 60, 76, 36, 56, 73, 71, 69, 50, 50,
 44, 55, 48, 58, 90, 59, 71, 37, 70, 67],
[71, 36, 72, 50, 67, 64, 83, 42, 62, 50,
 62, 27, 29, 77, 75, 39, 36, 31, 31, 84],
[87, 63, 88, 36, 54, 41, 35, 67, 44, 69,
 64, 32, 42, 38, 62, 83, 71, 31, 63, 39],
[86, 33, 38, 50, 57, 66, 67, 73, 69, 60,
 27, 38, 32, 33, 53, 73, 42, 36, 86, 57],
[42, 42, 70, 60, 79, 65, 81, 45, 33, 88,
 62, 71, 61, 89, 54, 51, 34, 44, 64, 64],
[85, 89, 66, 84, 80, 73, 47, 50, 67, 86,
 59, 75, 83, 51, 75, 83, 68, 39, 73, 75],
[52, 46, 30, 79, 62, 53, 82, 79, 40, 34,
 79, 61, 50, 71, 66, 87, 59, 47, 58, 58],
[41, 37, 25, 54, 52, 77, 76, 30, 39, 41,
 57, 79, 88, 81, 59, 52, 73, 41, 85, 40],
[55, 48, 49, 55, 89, 68, 75, 90, 76, 72,
 34, 72, 78, 78, 78, 39, 47, 63, 76, 42],
[73, 78, 25, 90, 49, 42, 78, 45, 71, 29,
 71, 36, 76, 60, 49, 84, 34, 88, 36, 51]]
max_list = copy.deepcopy(original_list) # Копирование списка в список макс.
путей
min_list = copy.deepcopy(original_list) # Копирование списка в список мин.
путей
for i in range(1, 20):
    # Определение мин. и макс. путей 1-ой строки и 1-го столбца таблицы
    max_list[0][i] = max_list[0][i - 1] + max_list[0][i]
    max_list[i][0] = max_list[i - 1][0] + max_list[i][0]
    min_list[0][i] = max_list[0][i - 1] + max_list[0][i]
    min_list[i][0] = max_list[i - 1][0] + max_list[i][0]
for i in range(1, 20):
    for j in range(1, 20):
        # Определение мин. и макс. путей оставшейся части таблицы
        if (i > 2 and i < 15 and j == 6):
            # Учет боковых стенок для робота-сборщика
            max_list[i][j] = max_list[i - 1][j] + original_list[i][j]
            min_list[i][j] = min_list[i - 1][j] + original_list[i][j]
        elif (i == 17 and j > 10 and j < 17):
            # Учет верхних стенок для робота-сборщика
            max_list[i][j] = max_list[i][j - 1] + original_list[i][j]
            min_list[i][j] = min_list[i][j - 1] + original_list[i][j]
        else:
            # Вычисление основных ячеек таблицы:
            max_list[i][j] = original_list[i][j] + max(max_list[i][j - 1],
max_list[i - 1][j])
            min_list[i][j] = original_list[i][j] + min(min_list[i][j - 1],
min_list[i - 1][j])
print("Максимальная денежная сумма =", max_list[19][19])
print("Минимальная денежная сумма =", min_list[19][19])

```

Рисунок 11 – Код задания №18.

Результаты работы программы задания №18 (рис. 12).

```

Максимальная денежная сумма = 2656
Минимальная денежная сумма = 1668

Process finished with exit code 0

```

Рисунок 12 – Результат работы программы.

Блок-схема алгоритма работы программы задания №18 (рис. 13).

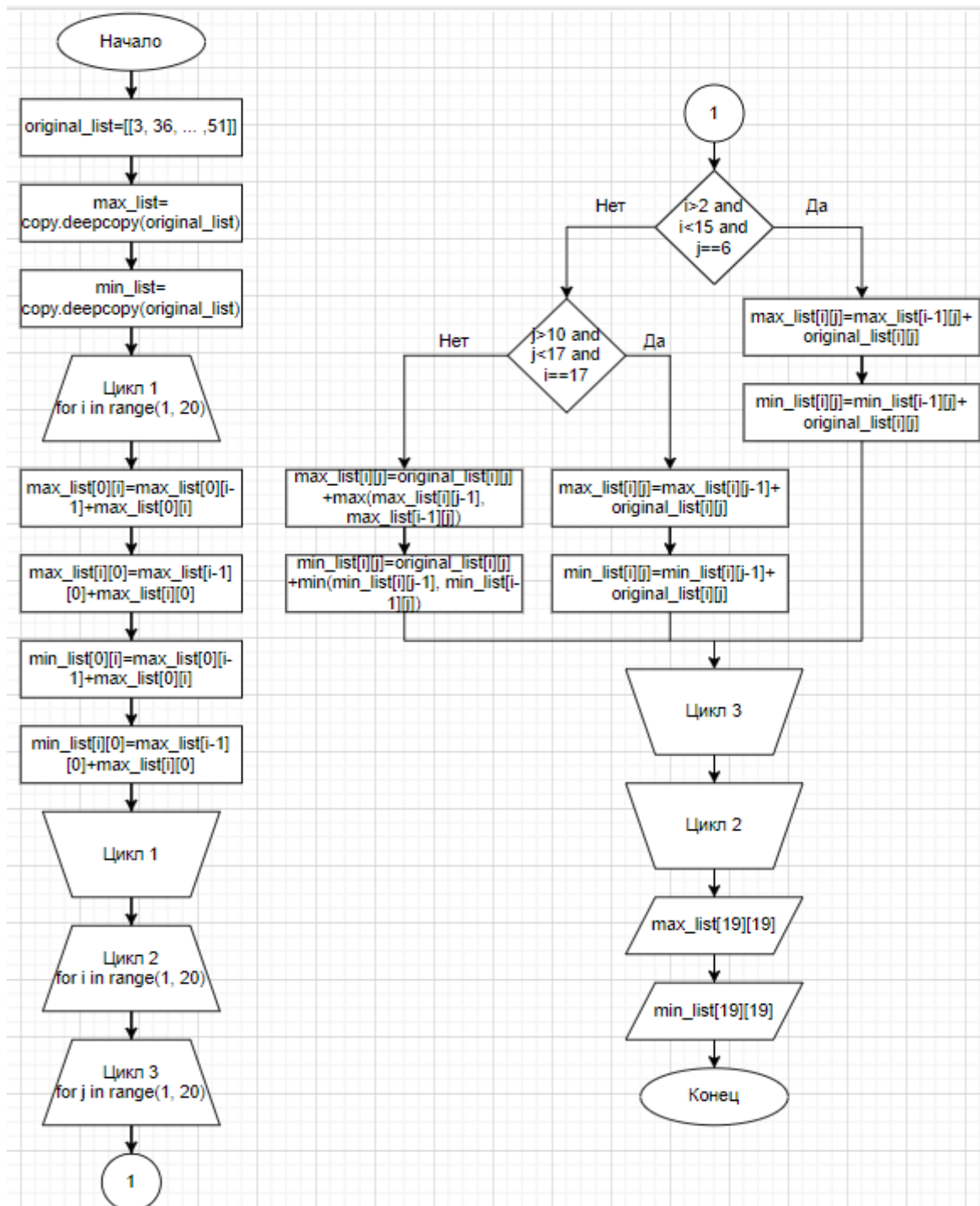


Рисунок 13 – Блок-схема алгоритма задачи №18.

Вывод: в ходе выполнения лабораторной работы были прорешены задачи ЕГЭ по информатике (тип 16, 17 и 18) на создание рекурсивного алгоритма, обработки строк и поиска максимального и минимального путей в таблице. Были созданы блок-схемы к задачам данного типа по ГОСТ-у 19.701-90. Были изучены особенности создания блок-схем по данному стандарту.