

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ
компьютерной безопасности и
криптографии

НЕЙРОННЫЕ СЕТИ
ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Яхина Шамиля Илдусовича

Преподаватель

Аспирант

подпись, дата

Б.А. Терebin

Саратов 2023

1. Создание ориентированного графа

На входе: текстовый файл с описанием графа в виде списка дуг:

$(a_1, b_1, n_1), (a_2, b_2, n_2), \dots, (a_k, b_k, n_k)$

где a_i - начальная вершина дуги i , b_i - конечная вершина дуги i , n_i - порядковый номер дуги в списке всех заходящих в вершину b_i дуг.

На выходе:

а) Ориентированный граф с именованными вершинами и линейно упорядоченными дугами (в соответствии с порядком из текстового файла).

б) Сообщение об ошибке в формате файла, если ошибка присутствует.

Способ проверки результата:

а) Сериализованная структура графа в формате XML или JSON.

Пример:

```
<graph>
  <vertex>v1</vertex>
  <vertex>v2</vertex>
  <vertex>v3</vertex>
  <arc>
    <from>v1</from>
    <to>v3</to>
    <order>1</order>
  </arc>
  <arc>
    <from>v2</from>
    <to>v3</to>
    <order>2</order>
  </arc>
</graph>
```

б) Сообщение об ошибке с указанием номера строки с ошибкой во входном файле.

Пример работы программы:

Ниже представлены примеры работы программы для входных файлов, соответствующих варианту 15.

На рисунках 1 – 2 показан первый пример работы программы nntask1.exe.

```

1 (1, 6, 1), (1, 1, 1), (1, 3, 1), (2, 2, 1), (1, 10, 1), (3, 4, 1), (1, 4, 1),
2 (1, 8, 1), (1, 9, 1), (2, 5, 1), (3, 7, 1)

```

Рисунок 1 – Содержимое файла 1.txt

```

D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>nntask1.exe input1=1.txt output1=1.xml
Ошибка в строке 1: дуга с номером 1 в вершину 4 уже существует
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>

```

Рисунок 2 – Результат работы программы для входного файла 1.txt

На рисунках 3 – 4 показан второй пример работы программы nntask1.exe.

```

1 (1, 6, 1), (1, 1, 1), (1, 3, 1), (2, 2, 1), (1, 10, 1), (3, 4, 1),
2 (5, 4, 1),
3 (1, 8, 1), (1, 9,
4 1), (2,
5 5, 1), (3, 7, 1)

```

Рисунок 3 – Содержимое файла 2.txt

```

D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>nntask1.exe input1=2.txt output1=2.xml
Ошибка в строке 2: дуга с номером 1 в вершину 4 уже существует
Ошибка в строке 3: неправильно задана компонента. Формат: (a, b, n)
Ошибка в строке 4: неправильно задана компонента. Формат: (a, b, n)
Ошибка в строке 4: неправильно задана компонента. Формат: (a, b, n)
Ошибка в строке 5: неправильно задана компонента. Формат: (a, b, n)

```

Рисунок 4 – Результат работы программы для входного файла 2.txt

На рисунках 5 – 6 показан третий пример работы программы nntask1.exe.

```

1 (1, 2, 1),
2 (2, 1, 1),
3 (2, 3, 1),
4 (3, 1, 1)

```

Рисунок 5 – Содержимое файла 6.txt

```

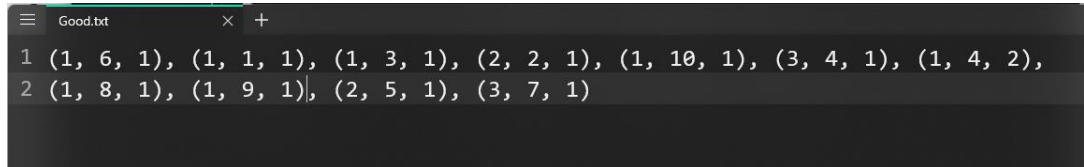
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>nntask1.exe input1=6.txt output1=6.xml
Ошибка в строке 4: дуга с номером 1 в вершину 1 уже существует

```

Рисунок 6 – Результат работы программы для входного файла 6.txt

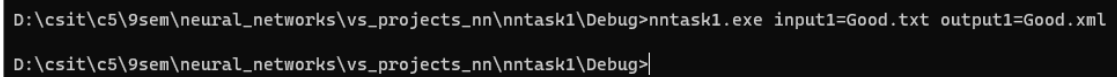
Так как все входные файлы, соответствующие варианту 15, выдают ошибки, проверим работу программы на одном файле с корректными данными.

На рисунках 7 – 9 показан четвертый пример работы программы nntask1.exe.



```
Good.txt
1 (1, 6, 1), (1, 1, 1), (1, 3, 1), (2, 2, 1), (1, 10, 1), (3, 4, 1), (1, 4, 2),
2 (1, 8, 1), (1, 9, 1), (2, 5, 1), (3, 7, 1)
```

Рисунок 7 – Содержимое файла Good.txt



```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>nntask1.exe input1=Good.txt output1=Good.xml
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask1\Debug>
```

Рисунок 8 – Результат работы программы для входного файла Good.txt

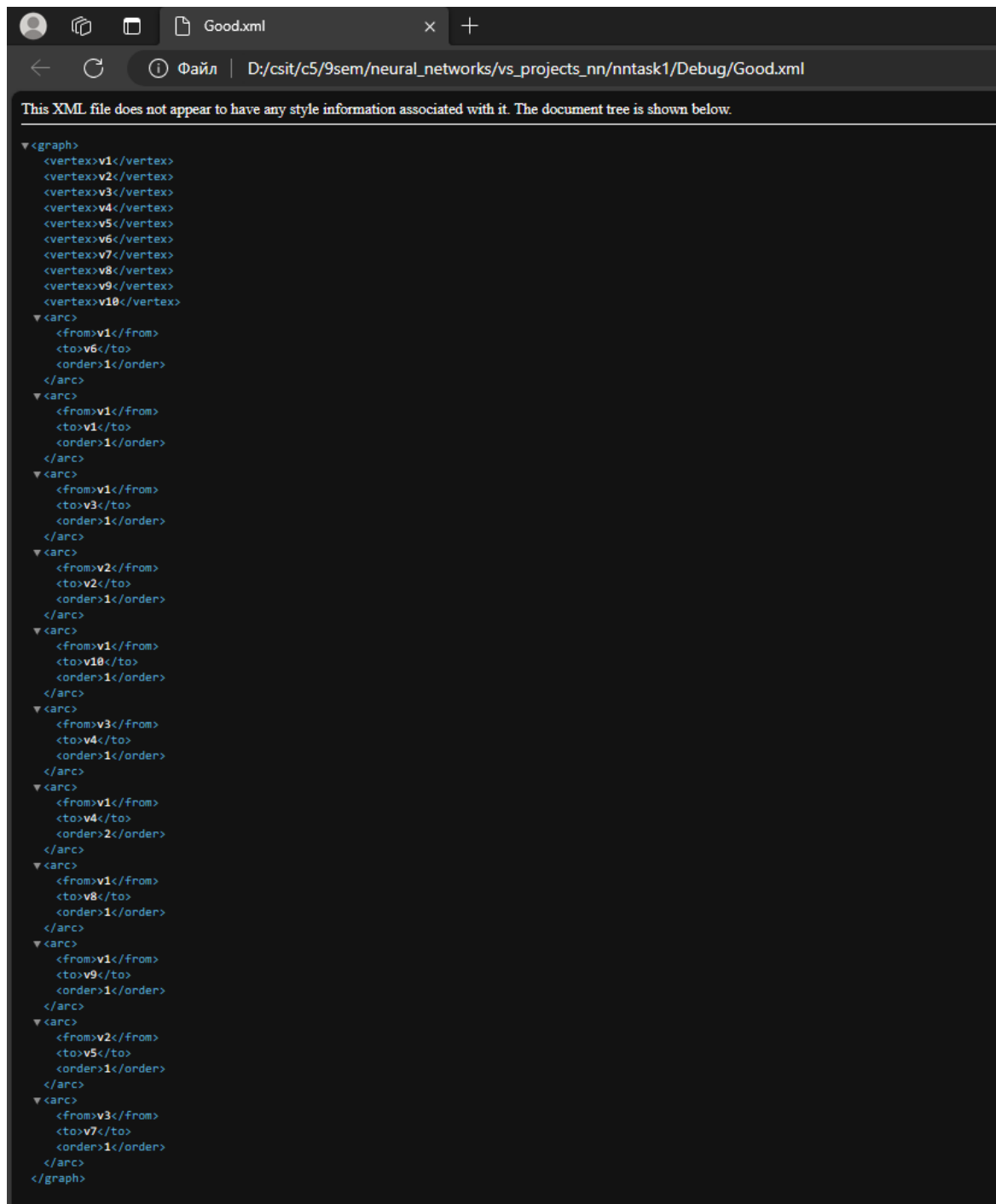


Рисунок 9 – Содержимое выходного файла Good.xml для входного файла Good.txt

2. Создание функции по графу

На входе: ориентированный граф с именованными вершинами как описано в задании 1.

На выходе: линейное представление функции, реализуемой графом в префиксной скобочной записи:

$A1(B1(C1(...),..., Cn(...)),..., Vn(...))$

Способ проверки результата:

а) выгрузка в текстовый файл результата преобразования графа в имя функции.

б) сообщение о наличии циклов в графе, если они присутствуют.

Пример работы программы:

Ниже представлены примеры работы программы для входных файлов, соответствующих варианту 15.

На рисунках 10 – 12 показан первый пример работы программы nntask2.exe.

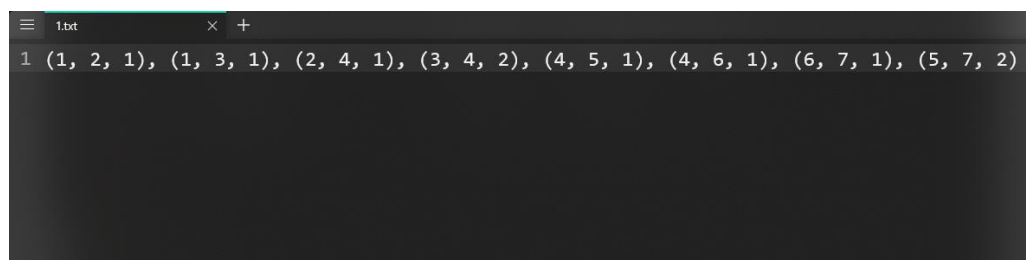


Рисунок 10 – Содержимое файла 1.txt

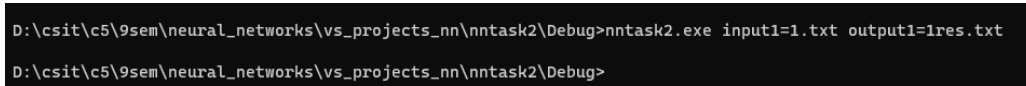


Рисунок 11 – Результат работы программы для входного файла 1.txt

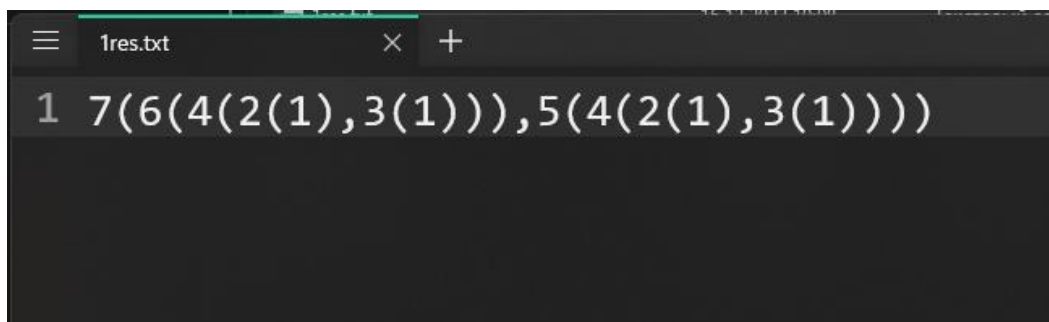
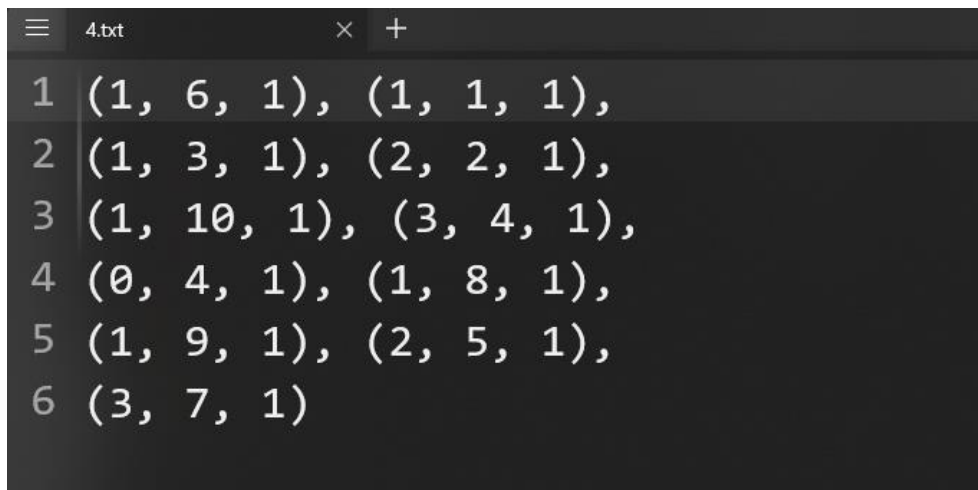


Рисунок 12 – Содержимое выходного файла 1res.txt для входного файла 1.txt

На рисунках 13 – 14 показан второй пример работы программы nntask2.exe.



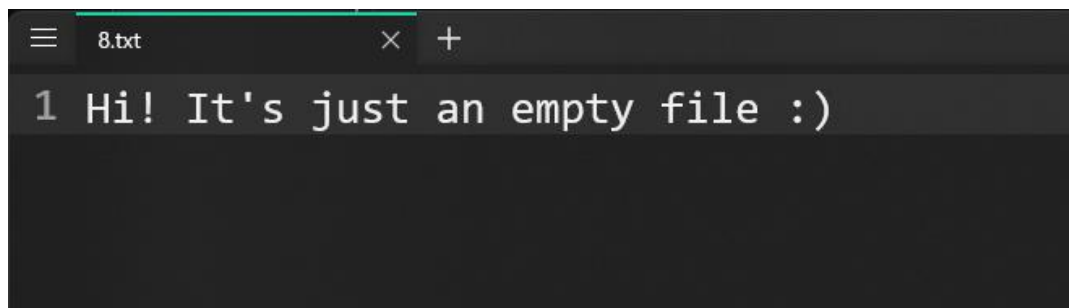
```
1 (1, 6, 1), (1, 1, 1),
2 (1, 3, 1), (2, 2, 1),
3 (1, 10, 1), (3, 4, 1),
4 (0, 4, 1), (1, 8, 1),
5 (1, 9, 1), (2, 5, 1),
6 (3, 7, 1)
```

Рисунок 13 – Содержимое файла 4.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask2\Debug>nntask2.exe input1=4.txt output1=4res.txt
Ошибка в строке 4: вершины 0 быть не может
Ошибка в строке 1: в графе существует цикл
Ошибка в строке 2: в графе существует цикл
```

Рисунок 14 – Результат работы программы для входного файла 4.txt

На рисунках 15 – 16 показан третий пример работы программы nntask2.exe.



```
1 Hi! It's just an empty file :)
```

Рисунок 15 – Содержимое файла 8.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask2\Debug>nntask2.exe input1=8.txt output1=8res.txt
Ошибка в строке 1: неправильно задана компонента. Формат: (a, b, n)
```

Рисунок 16 – Результат работы программы для входного файла 8.txt

3. Вычисление значения функции на графе

На входе:

а) Текстовый файл с описанием графа в виде списка дуг (смотри задание 1).

б) Текстовый файл соответствий арифметических операций именам вершин:

a_1 : операция_1

a_2 : операция_2

...

a_n : операция_n

где a_i - имя i-й вершины, операция_i - символ операции, соответствующий вершине a_i.

Допустимы следующие символы операций:

+ – сумма значений,

* – произведение значений,

exp – экспонирование входного значения,

число – любая числовая константа.

На выходе: значение функции, построенной по графу а) и файлу б).

Способ проверки результата: результат вычисления, выведенный в файл.

Пример работы программы:

Ниже представлены примеры работы программы для входных файлов, соответствующих варианту 15.

На рисунках 17 – 19 показан первый пример работы программы nntask3.exe.

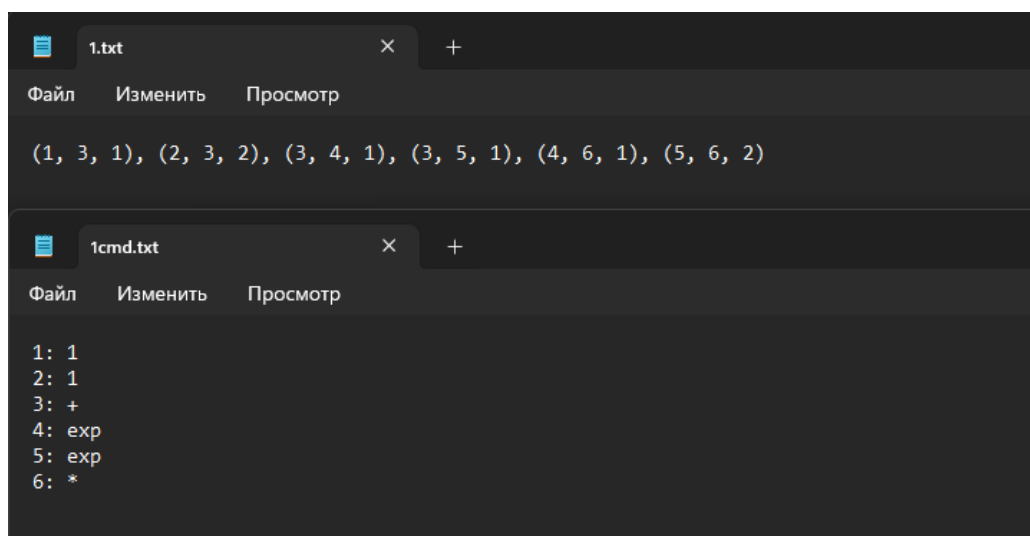


Рисунок 17 – Содержимое файлов 1.txt и 1cmd.txt


```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask3\Debug>ntask3.exe input1=1.txt input2=1cmd.txt output1=1prefix.txt output2=1znach.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask3\Debug>
```

Рисунок 18 – Результат работы программы для входных файлов 1.txt и 1cmd.txt

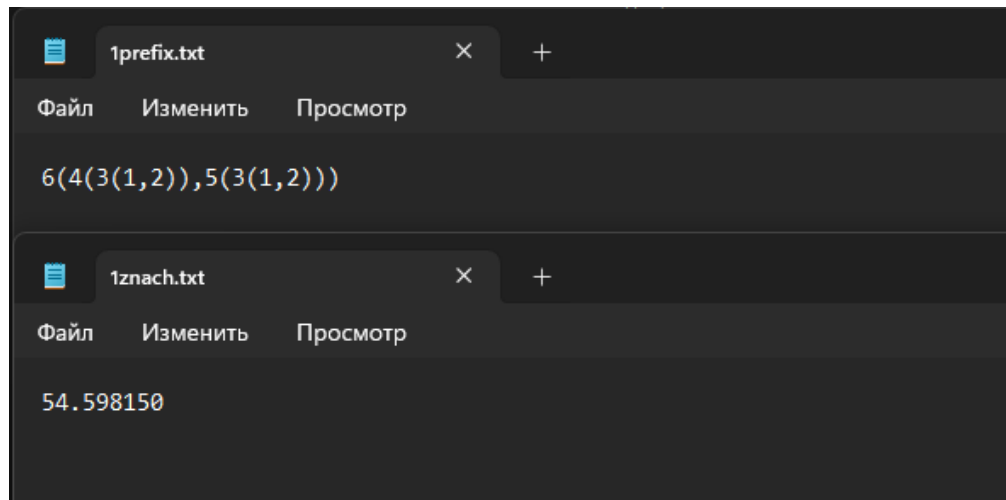


Рисунок 19 – Содержимое выходных файлов 1prefix.txt и 1znach.txt для входных файлов 1.txt и 1cmd.txt

На рисунках 20 – 22 показан второй пример работы программы ntask3.exe.

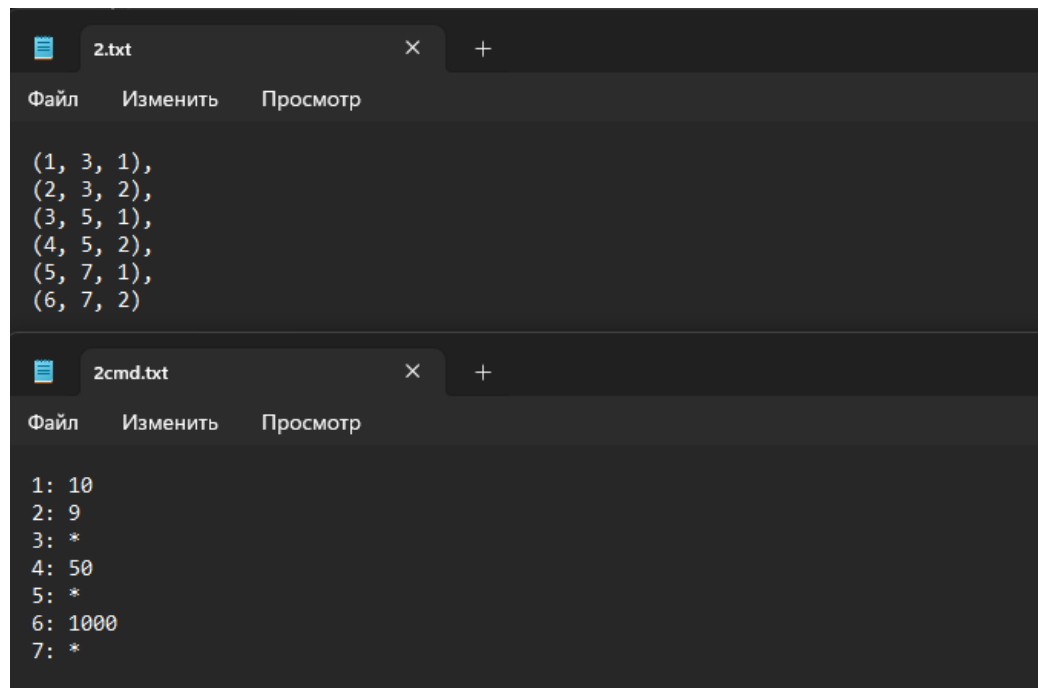


Рисунок 20 – Содержимое файлов 2.txt и 2cmd.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask3\Debug>ntask3.exe input1=2.txt input2=2cmd.txt output1=2prefix.txt output2=2znach.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask3\Debug>
```

Рисунок 21 – Результат работы программы для входных файлов 2.txt и 2cmd.txt

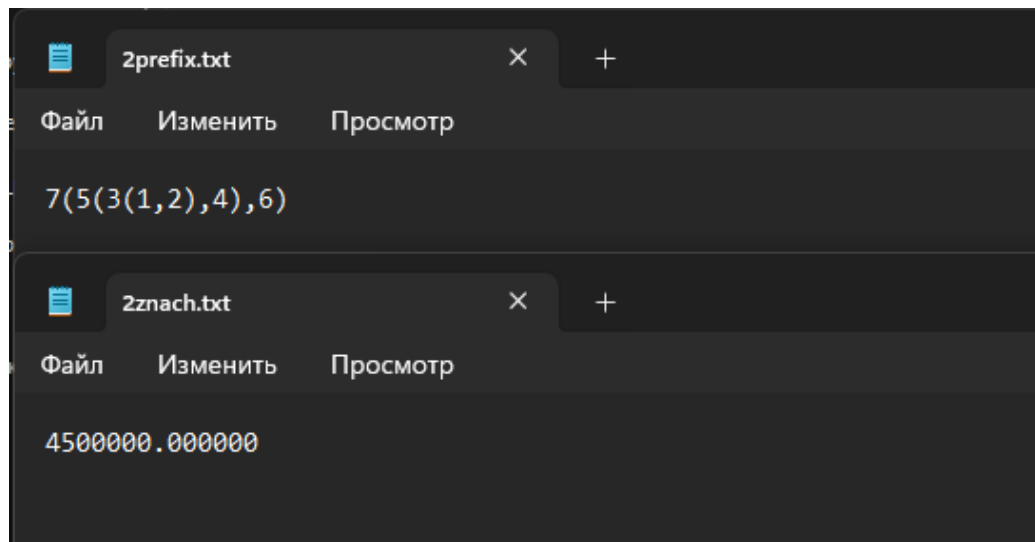


Рисунок 22 – Содержимое выходных файлов 2prefix.txt и 2znach.txt для входных файлов 2.txt и 2cmd.txt

На рисунках 23 – 25 показан третий пример работы программы nntask3.exe.

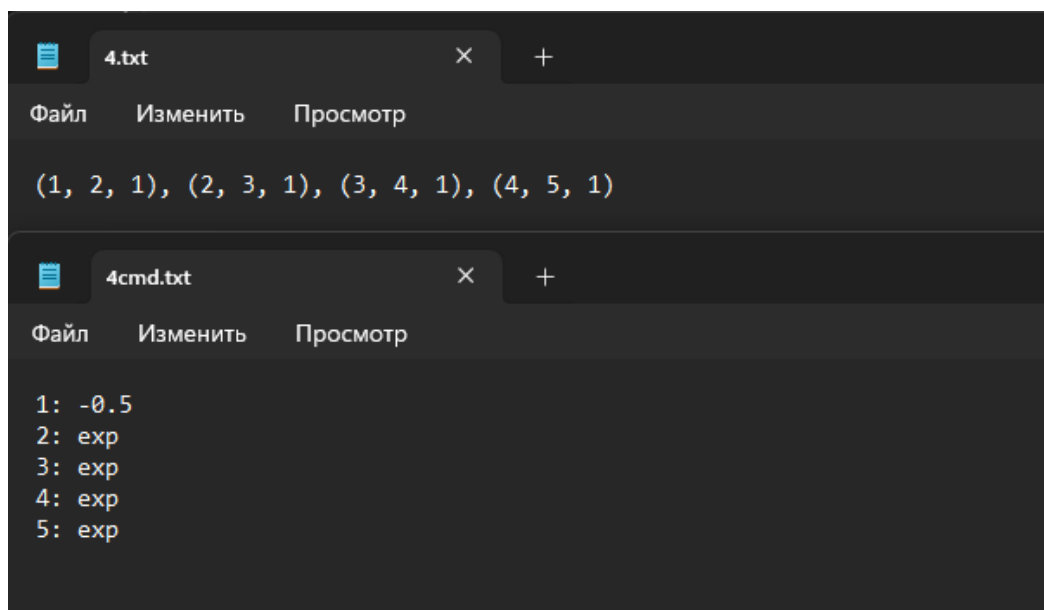


Рисунок 23 – Содержимое файлов 4.txt и 4cmd.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask3\Debug>nntask3.exe input1=4.txt input2=4cmd.txt output1=4prefix.txt output2=4znach.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask3\Debug>
```

Рисунок 24 – Результат работы программы для входных файлов 4.txt и 4cmd.txt

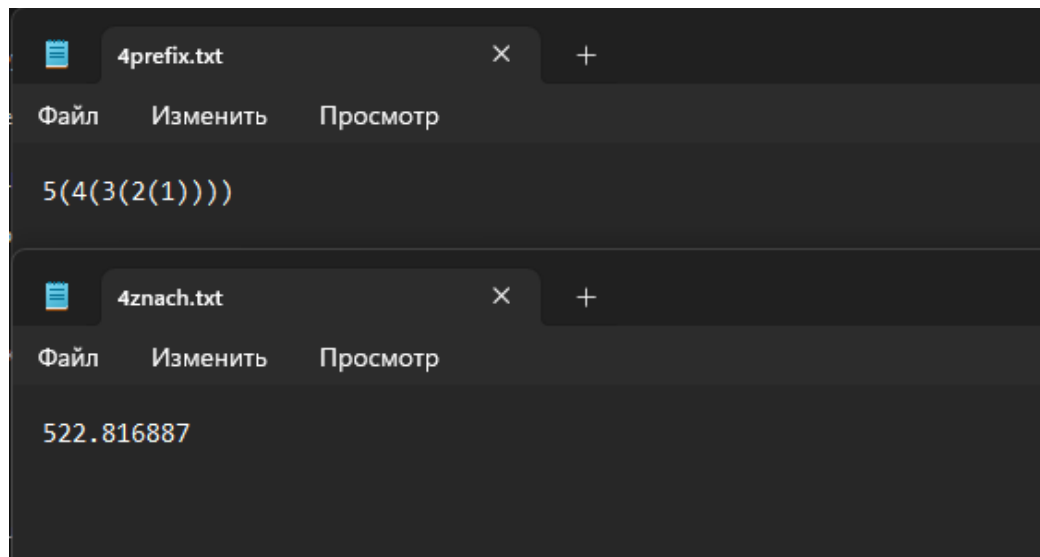


Рисунок 25 – Содержимое выходных файлов 4prefix.txt и 4znach.txt для входных файлов 4.txt и 4cmd.txt

4. Построение многослойной нейронной сети

На входе:

- а) Текстовый файл с набором матриц весов межнейронных связей:

$M1 : [M1[1,1], M1[1,2], \dots, M1[1,n]], \dots, [M1[m,1], M1[m,2], \dots, M1[m,n]]$

$M2 : [M2[1,1], M2[1,2], \dots, M2[1,n]], \dots, [M2[m,1], M2[m,2], \dots, M2[m,n]]$

...

$Mp : [Mp[1,1], Mp[1,2], \dots, Mp[1,n]], \dots, [Mp[m,1], Mp[m,2], \dots, Mp[m,n]]$

- б) Текстовый файл с входным вектором в формате:

x_1, x_2, \dots, x_n .

На выходе:

- а) Сериализованная многослойная нейронная сеть (в формате XML или JSON) с полносвязной межслойной структурой.

Файл с выходным вектором – результатом вычислений НС в формате:

y_1, y_2, \dots, y_n .

- в) Сообщение об ошибке, если в формате входного вектора или файла описания НС допущена ошибка.

Пример работы программы:

Ниже представлены примеры работы программы для входных файлов, соответствующих варианту 15.

На рисунках 26 – 29 показан первый пример работы программы nntask4.exe.

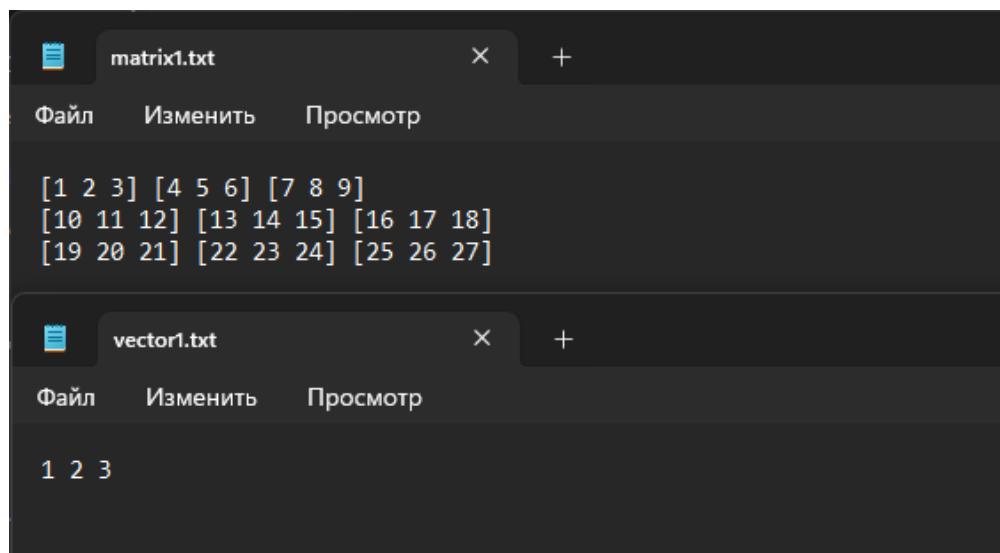


Рисунок 26 – Содержимое файлов matrix1.txt и vector1.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>nntask4.exe input1=matrix1.txt input2=vector1
.txt output1=matrix1.json output2=res1.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>
```

Рисунок 27 – Результат работы программы для входных файлов matrix1.txt и vector1.txt

```
matrix1.json X
D: > csit > c5 > 9sem > neural_networks > vs_projects_nn > nntask4 > Debug > matrix1.json > [ ] 1 > [ ] 0

1  [
2    [
3      [
4        [
5          1,
6          2,
7          3
8        ],
9        [
10         4,
11         5,
12         6
13       ],
14       [
15         7,
16         8,
17         9
18       ]
19     ],
20   ],
21   [
22     [
23       [
24         10,
25         11,
26         12
27       ],
28       [
29         13,
30         14,
31         15
32       ],
33       [
34         16,
35         17,
36         18
37       ]
38     ],
39   ],
40   [
41     [
42       [
43         19,
44         20,
45         21
46       ],
47       [
48         22,
49         23,
50         24
51       ],
52       [
53         25,
54         26,
55         27
56       ]
57     ],
58   ]
59 ]
60
```

Рисунок 28 – Содержимое выходного файла matrix1.json для входных файлов matrix1.txt и vector1.txt

```
res1.txt
1 0.9831982861694673 0.9853573859225755 0.9870247650305428
```

Рисунок 29 – Содержимое выходного файла res1.txt для входных файлов matrix1.txt и vector1.txt

На рисунках 30 – 33 показан второй пример работы программы nntask4.exe.

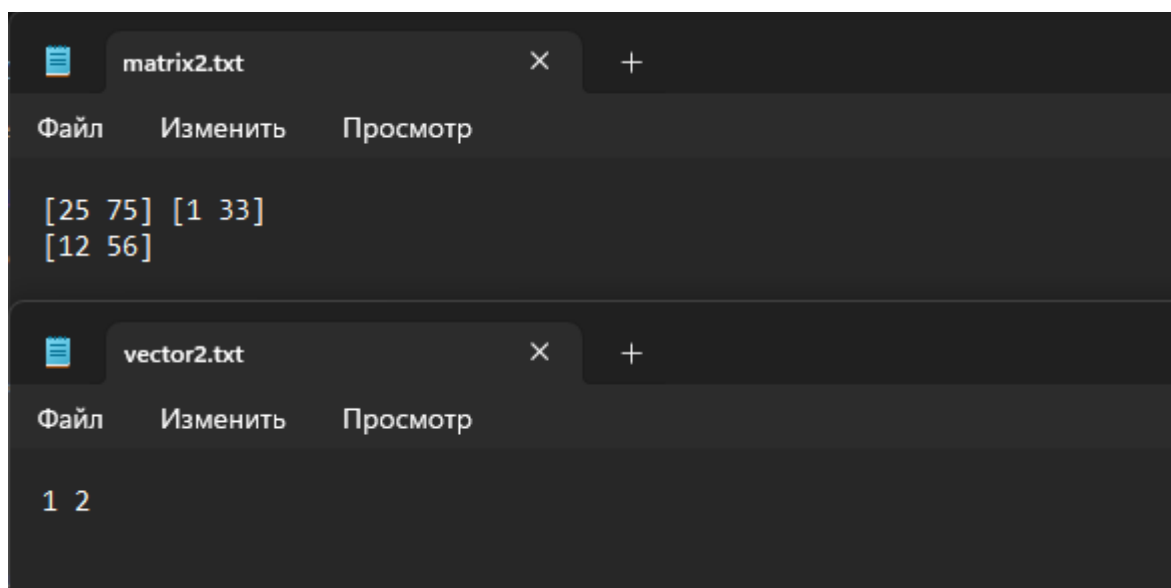


Рисунок 30 – Содержимое файлов matrix2.txt и vector2.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>nntask4.exe input1=matrix2.txt input2=vector2
.txt output1=matrix2.json output2=res2.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>
```

Рисунок 31 – Результат работы программы для входных файлов matrix2.txt и vector2.txt

```
1  [
2
3  [
4      [
5          25,
6          75
7      ],
8      [
9          1,
10         33
11     ]
12 ]
13 ],
14 [
15     [
16         [
17             12,
18             56
19         ]
20     ]
21 ]
22 ]
23
```

Рисунок 32 – Содержимое выходного файла matrix2.json для входных файлов matrix2.txt и vector2.txt

```
1 0.9853174992639121
```

Рисунок 33 – Содержимое выходного файла res2.txt для входных файлов matrix2.txt и vector2.txt

На рисунках 34 – 38 показан третий пример работы программы nntask4.exe.

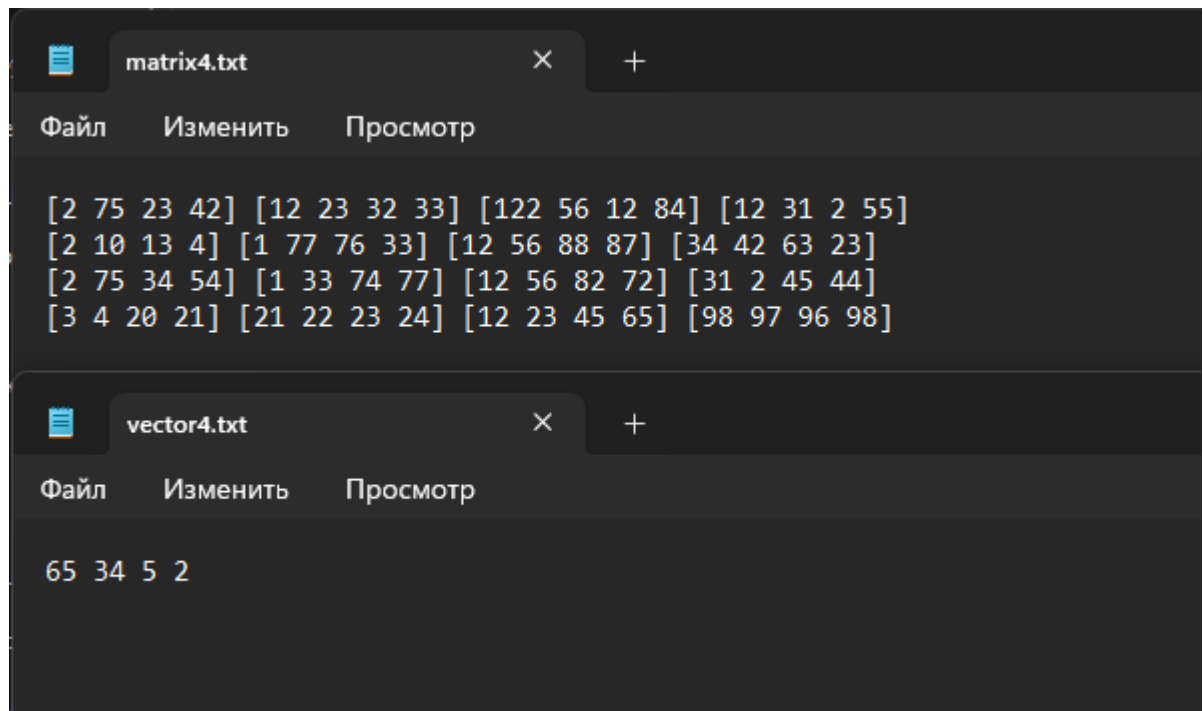


Рисунок 34 – Содержимое файлов matrix4.txt и vector4.txt

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>nntask4.exe input1=matrix4.txt input2=vector4
.txt output1=matrix4.json output2=res4.txt
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask4\Debug>|
```

Рисунок 35 – Результат работы программы для входных файлов matrix4.txt и vector4.txt


```
matrix4.json x
D: > csit > c5 > 9sem > neural_networks > vs_projects_nn > nntask4 > Debug > matrix4.json > [ ] 2
1  [
2  [
3  [
4  [
5      2,
6      75,
7      23,
8      42
9  ],
10 [
11     12,
12     23,
13     32,
14     33
15 ],
16 [
17     122,
18     56,
19     12,
20     84
21 ],
22 [
23     12,
24     31,
25     2,
26     55
27 ]
28 ],
29 [
30 [
31 [
32 [
33     2,
34     10,
35     13,
36     4
37 ],
38 [
39     1,
40     77,
41     76,
42     33
43 ],
44 [
45     12,
46     56,
47     88,
48     87
49 ],
50 [
51     34,
52     42,
53     63,
54     23
55 ]
56 ]
57 ],
58 [
59 [
```

Рисунок 36 – Первая часть содержимого выходного файла matrix4.json для входных файлов matrix4.txt и vector4.txt

```
58 [
59   [
60     2,
61     75,
62     34,
63     54
64   ],
65   [
66     1,
67     33,
68     74,
69     77
70   ],
71   [
72     12,
73     56,
74     82,
75     72
76   ],
77   [
78     31,
79     2,
80     45,
81     44
82   ]
83 ],
84 [
85   [
86     3,
87     4,
88     20,
89     21
90   ],
91   [
92     21,
93     22,
94     23,
95     24
96   ],
97   [
98     12,
99     23,
100    45,
101    65
102   ],
103   [
104     98,
105     97,
106     96,
107     98
108   ]
109 ],
110 ],
111 ],
112 ],
113 ],
114 ],
115 ]
```

Рисунок 37 – Вторая часть содержимого выходного файла matrix4.json для входных файлов matrix4.txt и vector4.txt

```
1 0.9794648894110536 0.9889444750533095 0.9931065324537568 0.9974203006324479
```

Рисунок 38 – Содержимое выходного файла res4.txt для входных файлов matrix4.txt и vector4.txt

5. Реализация метода обратного распространения ошибки для многослойной НС

На входе:

а) Текстовый файл с описанием НС (формат см. в задании 4).

б) Текстовый файл с обучающей выборкой:

$[x_{11}, x_{12}, \dots, x_{1n}] \rightarrow [y_{11}, y_{12}, \dots, y_{1m}]$

...

$[x_{k1}, x_{k2}, \dots, x_{kn}] \rightarrow [y_{k1}, y_{k2}, \dots, y_{km}]$

Формат описания входного вектора x и выходного вектора y соответствует формату из задания 4.

в) Число итераций обучения (в строке параметров).

На выходе:

Текстовый файл с историей N итераций обучения методом обратного распространения ошибки:

1 : Ошибка1

2 : Ошибка2

...

N : Ошибка N

Пример работы программы:

Ниже представлены примеры работы программы для входных файлов, соответствующих варианту 15.

На рисунках 39 – 42 показан первый пример работы программы nntask5.exe.

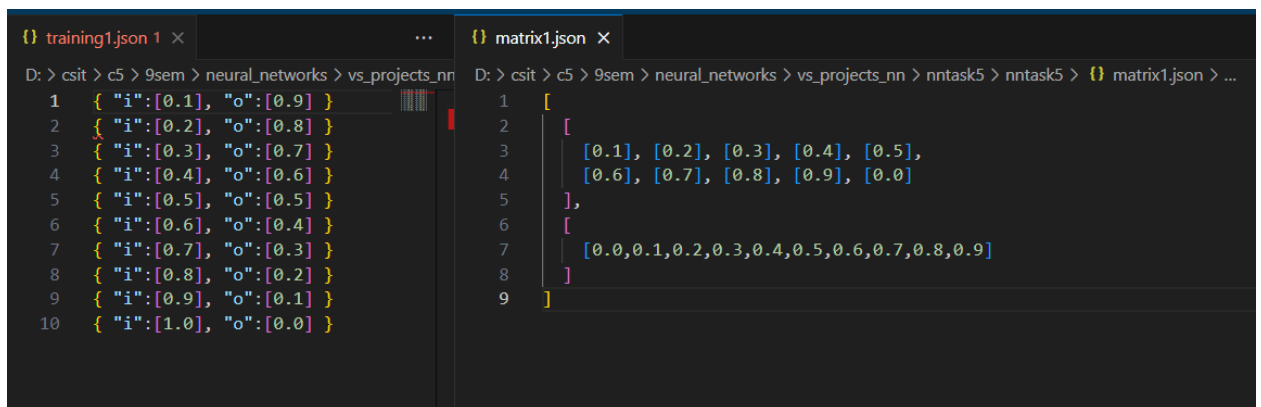


Рисунок 39 – Содержимое файлов matrix1.json и training1.json

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>nntask5.exe input1=matrix1.json input2=traini
ng1.json output1=output1.txt iterats=100
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>
```

Рисунок 40 – Результат работы программы для входных файлов matrix1.json и training1.json

```
output1.txt
1 1 : 0.0 0.04601906620676518 0.0902596596880356 0.13279836302196324 0.17370788746129157
0.21305730011355678 0.2509122359062895 0.287335095482364 0.322385230075034 0.4309041283336664
2 2 : 0.00001139301881463638 0.023139583658110988 0.04453778027217453 0.06434892087355124
0.08270219631042547 0.09971457241386049 0.11549212134943922 0.1301311888373507 0.1437194198031962
0.2250038987242403
3 3 : 0.000011720946353361596 0.011497443106873978 0.021734764714270858 0.030872360640948858
0.03903845437888449 0.04634403291351679 0.052885493555268615 0.058746833868962754 0.06400147206255062
0.11599094791970145
4 4 : 0.000006968182315059297 0.004834650678569379 0.008984029079194197 0.012558732722306397
0.015644235067440747 0.018311464109986336 0.020619610966132677 0.022618332620635943 0.024349492743838987
0.05057410730795944
5 5 : 0.000001184903996045845 6.95764348047973 0.0012719892265711642 0.0017516305804972632
0.0021518977400495215 0.002486516424456771 0.002766523940811527 0.003000862740025325 0.003196828341625257
0.007544060626435457
6 6 : -0.0000036149830151432134 -0.0020345796675078484 -0.0036623525108527653 -0.004972750325780218 -0.006
030914706811199 -0.006886897903314106 -0.007579582012029399 -0.008139473142562553 -0.008590722670430866 -
0.022861430792118956
7 7 : -0.000006443990745808007 -0.003919718542250888 -0.006952459667401737 -0.00931571236922578 -0.0111636
0308117094 -0.012610461886498971 -0.013742284234128464 -0.014624561269235262 -0.015307727929154549 -0.045
63886038906138
8 8 : -0.000006791619858113513 -0.005268149398762694 -0.009214229893914514 -0.01219311116058819 -0.0144497
91120376702 -0.016160066544671184 -0.01745223015552964 -0.01842131603024507 -0.019138661305576853 -0.0635
6430352755536
9 9 : -0.000004366073732276837 -0.006259978011516268 -0.010804293290387125 -0.014130000453987535 -0.016571
88427925056 -0.018362730659598214 -0.019667667564975993 -0.020605890779776315 -0.02126476924563139 -0.078
28660565091337
10 10 : 0.0000010262256086895102 -0.007005976315530074 -0.011940074100996633 -0.015443064458058209 -0.01793662629
626783 -0.01970535646891027 -0.02094581003884543 -0.021796467302815294 -0.022356597517453603 -0.090769696
0914384
11 11 : -0.00004884932675336205 0.045960060719337816 0.09020224163938556 0.1327504037354823
0.17367463256097834 0.21304218584896525 0.2509174364024936 0.28736189686937363 0.32243429049586797
0.4310092804922415
12 12 : -0.000013615793558247577 0.023119238757473175 0.04452748058877112 0.0643515537938257
0.08271912583547082 0.09974623101651073 0.11553837724081313 0.13019157667506567 0.1437932866661864
0.22517428752859794
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask5\Debug\output1.txt Стр 1, столб 1 100% Windows (CRLF) UTF-8
```

Рисунок 41 – Начало содержимого выходного файла output1.txt для входных файлов matrix1.json и training1.json

```
output1.txt
-----
8271791
90 90 : 0.00007064694483407013 -0.006960077902495981 -0.011925030492708438 -0.01545469296889541 -0.01796844287223
126 -0.01975145982857281 -0.021001579085290475 -0.02185852522742656 -0.02242257474581175 -0.0898129497006
4274
91 91 : -0.000440129956440464 0.04548544286278293 0.08973811811119478 0.13236002671912936
0.17340031990078428 0.212911737039939 0.2509484806036391 0.28756488788480067 0.3228145990793827
0.43185823036638127
92 92 : -0.00021462958205596574 0.02295319490890728 0.044439696427926445 0.06436547798398651
0.08284598133815174 0.09998971774140807 0.11589761546945064 0.13066296727201768 0.14437169321120868
0.22591656269770446
93 93 : -0.00010289637005457856 0.01145103000442081 0.021774970650899533 0.03100485631916627
0.039262036982402104 0.046653960336884646 0.053275381879477744 0.059209729112785034 0.06453044264989591
0.11677305136078903
94 94 : -0.00004233108524525054 0.004850475531838208 0.009066790134390011 0.012704967765030464
0.0158482504358354 0.018566888554378002 0.02092016570129444 0.022958187153279307 0.024723403127911835
0.05122910322408245
95 95 : -0.000006388401625623069 7.400322869926108 0.0013609400637643364 0.0018785890560778199
0.002310948764150762 0.0026725501243632928 0.002975183298636369 0.003228456441073151
0.0034402372962809543 0.008092016129535346
96 96 : 0.00001727148049266229 -0.001978116936480026 -0.0035818613149627494 -0.004874923337137293 -0.005919910772
522284 -0.00676550477182165 -0.007449811130781618 -0.008002870504758286 -0.008448535276514154 -0.02238425
237141811
97 97 : 0.000034900130065105124 -0.003859438272965136 -0.00688627017012828 -0.009248547639100648 -0.0110969934752
28191 -0.01254460702212156 -0.013676951127071131 -0.01455945331903361 -0.015242618246924236 -0.0451723315
82542566
98 98 : 0.00004991662714325082 -0.00520857452033024 -0.0091643604398373 -0.012155129095758908 -0.0144222796131433
31 -0.016140715373949305 -0.01743884136050734 -0.018412111232728134 -0.019132314947842906 -0.063016843409
74856
99 99 : 0.00006432666184657508 -0.0062036068338407364 -0.01077103084013361 -0.014118734920152523 -0.0165782151850
99606 -0.01838205650465308 -0.019696143083535025 -0.020640589071857802 -0.021303575613425847 -0.077528118
66517911
100 100 : 0.00007938889043405972 -0.006954174837075901 -0.011922900392352484 -0.015455842187840916 -0.0179720786076
897 -0.01975685556184945 -0.02100816478591899 -0.021865884379574235 -0.022430415447186054 -0.089697544996
31007
D:\csit\c5\9sem\neural_networks\vs_projects_nn\ntask5\Debug\output1.txt
Стр 1, столб 1 100% Windows (CRLF) UTF-8
```

Рисунок 42 – Конец содержимого выходного файла output1.txt для входных файлов matrix1.json и training1.json

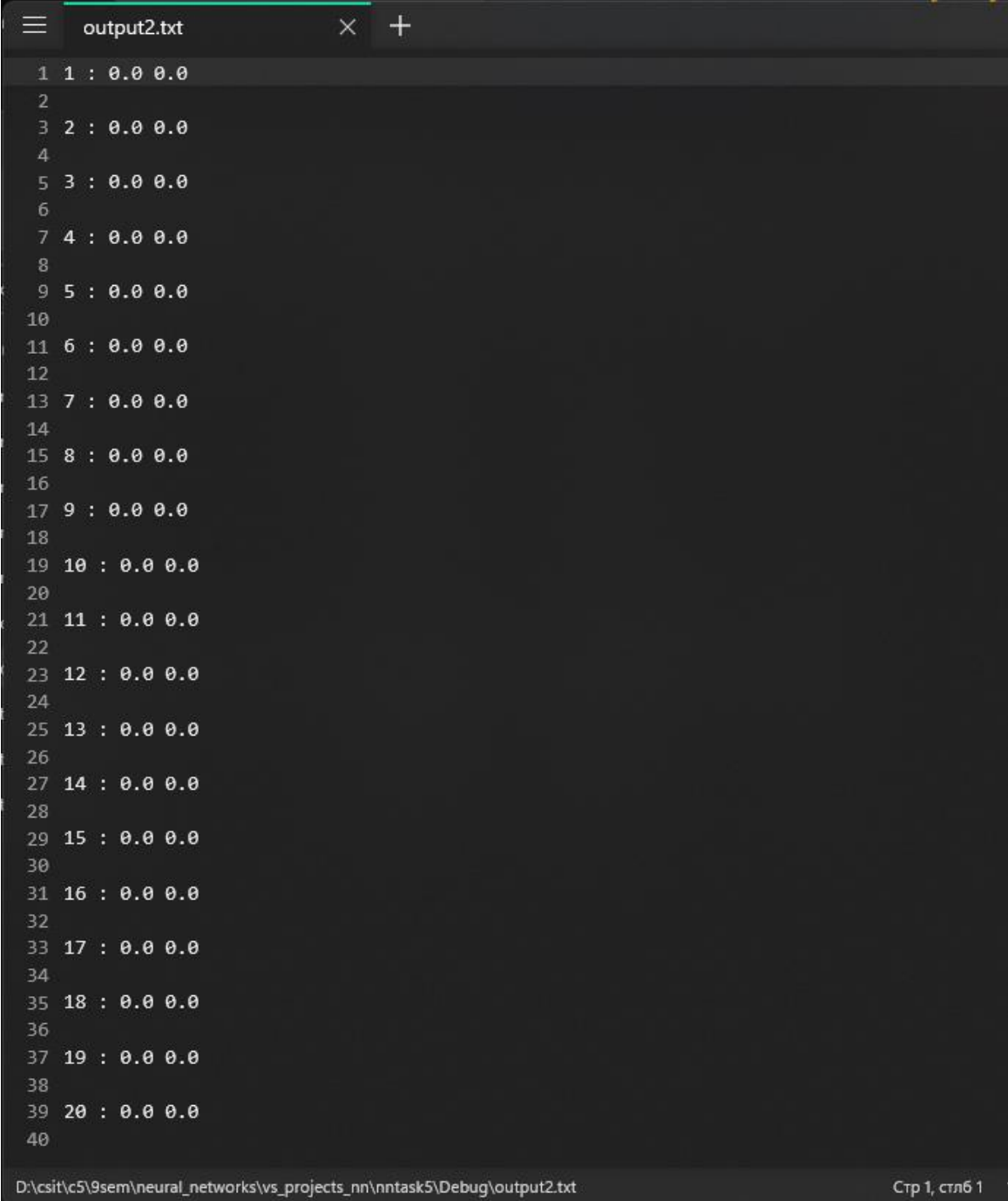
На рисунках 43 – 46 показан второй пример работы программы nntask5.exe.

```
training2.json  matrix2.json
D: > csit > c5 > 9sem > neural_networks > vs_projects_nn > nntask5 > Debug  D: > csit > c5 > 9sem > neural_networks > vs_projects_nn > nntask5 > Debug > {} matrix2.json > ...
1  { "i": [1, 2], "o": [0.9853174992639121] }
1  [
2    [
3      [25, 75],
4      [1, 33]
5    ],
6    [
7      [12, 56]
8    ]
9  ]
```

Рисунок 43 – Содержимое файлов matrix2.json и training2.json

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>nntask5.exe input1=matrix2.json input2=training2.json output1=output2.txt iterats=100  
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>
```

Рисунок 44 – Результат работы программы для входных файлов matrix2.json и training2.json



```
1 1 : 0.0 0.0  
2  
3 2 : 0.0 0.0  
4  
5 3 : 0.0 0.0  
6  
7 4 : 0.0 0.0  
8  
9 5 : 0.0 0.0  
10  
11 6 : 0.0 0.0  
12  
13 7 : 0.0 0.0  
14  
15 8 : 0.0 0.0  
16  
17 9 : 0.0 0.0  
18  
19 10 : 0.0 0.0  
20  
21 11 : 0.0 0.0  
22  
23 12 : 0.0 0.0  
24  
25 13 : 0.0 0.0  
26  
27 14 : 0.0 0.0  
28  
29 15 : 0.0 0.0  
30  
31 16 : 0.0 0.0  
32  
33 17 : 0.0 0.0  
34  
35 18 : 0.0 0.0  
36  
37 19 : 0.0 0.0  
38  
39 20 : 0.0 0.0  
40
```

D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug\output2.txt Стр 1, стлб 1

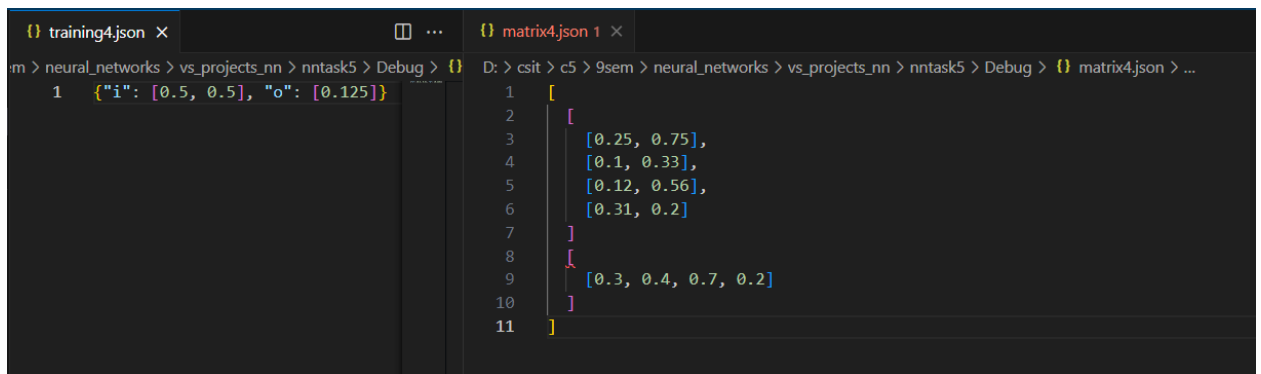
Рисунок 45 – Начало содержимого выходного файла output2.txt для входных файлов matrix2.json и training2.json

```
162  
163 82 : 0.0 0.0  
164  
165 83 : 0.0 0.0  
166  
167 84 : 0.0 0.0  
168  
169 85 : 0.0 0.0  
170  
171 86 : 0.0 0.0  
172  
173 87 : 0.0 0.0  
174  
175 88 : 0.0 0.0  
176  
177 89 : 0.0 0.0  
178  
179 90 : 0.0 0.0  
180  
181 91 : 0.0 0.0  
182  
183 92 : 0.0 0.0  
184  
185 93 : 0.0 0.0  
186  
187 94 : 0.0 0.0  
188  
189 95 : 0.0 0.0  
190  
191 96 : 0.0 0.0  
192  
193 97 : 0.0 0.0  
194  
195 98 : 0.0 0.0  
196  
197 99 : 0.0 0.0  
198  
199 100 : 0.0 0.0  
200  
201
```

D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug\output2.txt Стр 1, столб 1

Рисунок 46 – Конец содержимого выходного файла output2.txt для входных файлов matrix2.json и training2.json

На рисунках 47 – 48 показан третий пример работы программы nntask5.exe.



The image shows a code editor with two tabs: 'training4.json' and 'matrix4.json 1'. The 'training4.json' tab is active, showing a JSON object with keys 'i' and 'o'. The 'matrix4.json' tab is also visible, showing a JSON array of arrays. The file paths in the background are 'D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug'.

```
1 {"i": [0.5, 0.5], "o": [0.125]}
```

```
1 [  
2   [  
3     [0.25, 0.75],  
4     [0.1, 0.33],  
5     [0.12, 0.56],  
6     [0.31, 0.2]  
7   ]  
8   [  
9     [0.3, 0.4, 0.7, 0.2]  
10  ]  
11 ]
```

Рисунок 47 – Содержимое файлов matrix4.json и training4.json

```
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>nntask5.exe input1=matrix4.json input2=traini  
ng4.json output1=output4.txt iterats=100  
Файл matrix4.json имеет некорректное содержимое  
  
D:\csit\c5\9sem\neural_networks\vs_projects_nn\nntask5\Debug>
```

Рисунок 48 – Результат работы программы для входных файлов matrix4.json и training4.json

ПРИЛОЖЕНИЕ А

Листинг программы первого задания nntask1.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cctype>
#include <set>
#include <map>
#include <algorithm>

using namespace std;

struct Vert {
    int a, b, n, l;
};

class Graph {
private:
    vector<Vert> edges;
    set<pair<int, int>> uniqueEdgesAB;
    set<pair<int, int>> uniqueEdgesBN;
    int lineNum;
public:
    int addEdge(const Vert& edge) {
        if (uniqueEdgesAB.count({ edge.a, edge.b }) > 0) {
            return -1;
        }
        if (!uniqueEdgesBN.insert({ edge.b, edge.n }).second) {
            return -2;
        }
        uniqueEdgesAB.insert({ edge.a, edge.b });
        edges.push_back(edge);
        lineNum = edge.l;
        return 1;
    }

    const vector<Vert>& getEdges() const {
        return edges;
    }
    const int getLineNumber() const {
        return lineNum;
    }
};

bool fileToStruct(const string& fileName, Graph& graph) {
    bool isGoodResult = true;
    ifstream file(fileName);
    if (!file.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return false;
    }

    string line;
    size_t lineNumber = 0;

    while (getline(file, line)) {
        lineNumber++;
    }
}
```

```

    istream iss(line);

    char nextChar;
    bool findOpen = true;
    while (iss >> nextChar) {
        if (nextChar != '(' && findOpen) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
            isGoodResult = false;
            while (nextChar != '(') {
                iss >> nextChar;
            }
        }
        findOpen = false;
        Vert edge;
        string a, b, n;
        if (nextChar == '(') {

            while (iss >> nextChar && isdigit(nextChar)) {
                a += nextChar;
            }

            if (a == "0") {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  вершины 0
быть не может\n";
                isGoodResult = false;
                continue;
            }
            if (nextChar == '\n') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                continue;
            }
            if (isalpha(nextChar)) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  некорректные
данные (введены буквы вместо цифр)\n";
                isGoodResult = false;
                continue;
            }
            if (nextChar != ',') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                continue;
            }
            iss >> ws >> nextChar;
            while (iss && isdigit(nextChar)) {
                b += nextChar;
                iss >> nextChar;
            }

            if (b == "0") {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  вершины 0
быть не может\n";
                isGoodResult = false;
                continue;
            }
            if (nextChar == '\n') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                continue;
            }
        }
    }

```

```

        if (isalpha(nextChar)) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
            isGoodResult = false;
            continue;
        }

        if (nextChar != ',') {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
            isGoodResult = false;
            continue;
        }

        iss >> ws >> nextChar;
        while (iss && nextChar != ')' && isdigit(nextChar)) {
            n += nextChar;
            iss >> nextChar;
        }

        if (isalpha(nextChar)) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
            isGoodResult = false;
            continue;
        }

        if (nextChar == '\n') {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
            isGoodResult = false;
            continue;
        }

        if (nextChar != ')') {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
            isGoodResult = false;
            continue;
        }

        if (!a.empty() && !b.empty() && !n.empty()) {
            edge.a = stoi(a);
            edge.b = stoi(b);
            edge.n = stoi(n);
            edge.l = lineNumber;
            int errorCode = graph.addEdge(edge);
            if (errorCode == -1) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":
повторяющаяся дуга " + a + " -> " + b << "\n";
                isGoodResult = false;
            }
            if (errorCode == -2) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ": дуга с
номером " + n + " в вершину " + b + " уже существует\n";
                isGoodResult = false;
            }
        }
    }
    else {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
    }

    iss >> ws >> nextChar;

```

```

        if (nextChar != ',' && nextChar != '\n' && !iss.eof()) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
            isGoodResult = false;
        }
        if (iss.eof())
            break;
    }
}
return isGoodResult;
}

bool checkNumbers(const Graph& graph, set<int>& uniqueA, set<int>& uniqueB) {
    const vector<Vert>& edges = graph.getEdges();
    map<int, set<int>> incomingEdges;

    for (const auto& edge : edges) {
        uniqueB.insert(edge.b);
        uniqueA.insert(edge.a);
        incomingEdges[edge.b].insert(edge.n);
    }
    for (const auto& vertex : uniqueA) {
        if (vertex > *max_element(uniqueB.begin(), uniqueB.end())) {
            cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
Неправильная нумерация вершин. Номер вершины а больше количества вершин.\n";
            return false;
        }
    }

    for (const auto& entry : incomingEdges) {
        const auto& edgesSet = entry.second;
        for (int i = 1; i <= *max_element(edgesSet.begin(), edgesSet.end()); ++i) {
            if (edgesSet.find(i) == edgesSet.end()) {
                cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
неправильно заданы номера дуг\n";
                return false;
            }
        }
    }
    return true;
}

void writeVertices(const set<int>& elems, ofstream& outFile) {
    for (const auto& elem : elems) {
        outFile << "    <vertex>v" << elem << "</vertex>\n";
    }
}

void writeEdges(const vector<Vert>& edges, ofstream& outFile) {
    for (const auto& edge : edges) {
        outFile << "    <arc>\n";
        outFile << "        <from>v" << edge.a << "</from>\n";
        outFile << "        <to>v" << edge.b << "</to>\n";
        outFile << "        <order>" << edge.n << "</order>\n";
        outFile << "    </arc>\n";
    }
}

void graphToXML(const string& fileName, Graph& graph, set<int>& elems) {
    ofstream outFile(fileName, ios::out | ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }
}

```

```

    }

    outFile << "<graph>\n";

    writeVertices(elems, outFile);

    const auto& edges = graph.getEdges();
    writeEdges(edges, outFile);

    outFile << "</graph>\n";

    outFile.close();
}

string findInStr(string const& str, int n) {
    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    Graph graph;
    set<int> uniqueA;
    set<int> uniqueB;
    string input1, input2, output1, output2;
    for (int i = 0; argv[i]; i++)
    {
        string checkStr = string(argv[i]);
        if (checkStr.length() > 4) {
            string ifStr1 = findInStr(checkStr, 7);
            string ifStr2 = findInStr(checkStr, 8);
            string subStr = checkStr.substr(7, checkStr.length());
            string subStr2 = checkStr.substr(8, checkStr.length());
            if (ifStr1 == "input1=") {
                input1 = subStr;
            }
            if (ifStr1 == "input2=") {
                input2 = subStr;
            }
            if (ifStr2 == "output1=") {
                output1 = subStr2;
            }
            if (ifStr2 == "output2=") {
                output2 = subStr2;
            }
        }
    }
    bool isGoodResult = fileToStruct(input1, graph);
    if (!checkNumbers(graph, uniqueA, uniqueB)) {
        isGoodResult = false;
    }
    if (isGoodResult) {
        uniqueA.insert(uniqueB.begin(), uniqueB.end());
        graphToXML(output1, graph, uniqueA);
    }
    return 0;
}

```

ПРИЛОЖЕНИЕ Б

Листинг программы второго задания nntask2.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cctype>
#include <set>
#include <map>
#include <algorithm>

using namespace std;

struct Vert {
    int a, b, n, l;
};

class Graph {
private:
    vector<Vert> edges;
    set<pair<int, int>> uniqueEdgesAB;
    set<pair<int, int>> uniqueEdgesBN;
    int lineNum;
public:
    int addEdge(const Vert& edge) {
        if (uniqueEdgesAB.count({ edge.a, edge.b }) > 0) {
            return -1;
        }
        if (!uniqueEdgesBN.insert({ edge.b, edge.n }).second) {
            return -2;
        }
        uniqueEdgesAB.insert({ edge.a, edge.b });
        edges.push_back(edge);
        lineNum = edge.l;
        return 1;
    }

    const vector<Vert>& getEdges() const {
        return edges;
    }
    const int getLineNumber() const {
        return lineNum;
    }

    int findSink() const {
        set<int> allVertices;
        set<int> outVertices;

        for (const auto& edge : edges) {
            allVertices.insert(edge.a);
            allVertices.insert(edge.b);
            outVertices.insert(edge.a);
        }

        for (int vertex : allVertices) {
            if (outVertices.find(vertex) == outVertices.end()) {
                return vertex;
            }
        }
        return -1;
    }
};
```

```

    }

    int edgesCount() const {
        int maxElement = 1;
        for (const auto& edge : edges) {
            if (edge.a > maxElement)
                maxElement = edge.a;
            if (edge.b > maxElement)
                maxElement = edge.b;
        }
        return maxElement;
    }
};

void buildParents(const Graph& graph, vector <vector <int>>& parents, const vector<Vert>&
edges) {
    for (const auto& edge : edges)
        parents[edge.b - 1].push_back(edge.a);
    return;
}

void printPrefix(vector<vector<int>>& parents, int startVertex, ofstream &outFile) {
    outFile << startVertex;
    vector<int> checkedVertex = parents[startVertex - 1];
    if (!checkedVertex.empty()) {
        outFile << "(";
        for (int i = 0; i < checkedVertex.size(); ++i) {
            printPrefix(parents, checkedVertex[i], outFile);
            if (i < checkedVertex.size() - 1)
                outFile << ",";
        }
        outFile << ")";
    }
}

void findPrefix(const Graph& graph, const string& fileName) {
    const vector<Vert>& edges = graph.getEdges();
    set<int> visitedVertices;

    int startVertex = graph.findSink();

    int maxEl = graph.edgesCount();
    vector<int> dop;
    vector <vector <int>> parents(maxEl, dop);
    buildParents(graph, parents, edges);

    ofstream outFile(fileName, ios::out | ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    printPrefix(parents, startVertex, outFile);

    outFile.close();
}

bool fileToStruct(const string& fileName, Graph& graph) {
    bool isGoodResult = true;
    ifstream file(fileName);
    if (!file.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return false;
    }
}

```

```

string line;
size_t lineNumber = 0;

while (getline(file, line)) {
    lineNumber++;
    istringstream iss(line);

    char nextChar;
    bool findOpen = true;
    bool errorPrint1 = true;
    while (iss >> nextChar) {
        if (nextChar != '(' && findOpen) {
            if (errorPrint1) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
            }
            while (nextChar != '(') {
                if (!(iss >> nextChar))
                    return false;
            }
        }
        errorPrint1 = false;
        findOpen = false;
        Vert edge;
        string a, b, n;
        if (nextChar == '(') {

            while (iss >> nextChar && isdigit(nextChar)) {
                a += nextChar;
            }

            if (a == "0") {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  вершины 0
быть не может\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            if (nextChar == '\n') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                continue;
            }
            if (isalpha(nextChar)) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  некорректные
данные (введены буквы вместо цифр)\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            if (nextChar != ',') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            iss >> ws >> nextChar;
            while (iss && isdigit(nextChar)) {
                b += nextChar;
                iss >> nextChar;
            }

```



```

    }

    if (b == "0") {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": вершины 0
быть не может\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }
    if (nextChar == '\n') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        continue;
    }
    if (isalpha(nextChar)) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (nextChar != ',') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    iss >> ws >> nextChar;
    while (iss && nextChar != ')' && isdigit(nextChar)) {
        n += nextChar;
        iss >> nextChar;
    }

    if (isalpha(nextChar)) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }
    if (nextChar == '\n') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        continue;
    }
    if (nextChar != ')') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (!a.empty() && !b.empty() && !n.empty()) {
        edge.a = stoi(a);
        edge.b = stoi(b);
        edge.n = stoi(n);
        edge.l = lineNumber;
        int errorCode = graph.addEdge(edge);
    }

```

```

        if (errorCode == -1) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ":
повторяющаяся дуга " + a + " -> " + b << "\n";
            isGoodResult = false;
        }
        if (errorCode == -2) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": дуга с
номером " + n + " в вершину " + b + " уже существует\n";
            isGoodResult = false;
        }
    }
    else {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
    }

    iss >> ws >> nextChar;
    if (nextChar != ',' && nextChar != '\n' && !iss.eof()) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
    }
    if (iss.eof())
        break;
}
}
return isGoodResult;
}

```

```

bool checkNumbers(const Graph& graph, set<int>& uniqueA, set<int>& uniqueB) {
    const vector<Vert>& edges = graph.getEdges();
    map<int, set<int>> incomingEdges;

    for (const auto& edge : edges) {
        uniqueB.insert(edge.b);
        uniqueA.insert(edge.a);
        incomingEdges[edge.b].insert(edge.n);
    }
    for (const auto& vertex : uniqueA) {
        if (vertex > *max_element(uniqueB.begin(), uniqueB.end())) {
            cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
Неправильная нумерация вершин. Номер вершины а больше количества вершин.\n";
            return false;
        }
    }

    for (const auto& entry : incomingEdges) {
        const auto& edgesSet = entry.second;
        for (int i = 1; i <= *max_element(edgesSet.begin(), edgesSet.end()); ++i) {
            if (edgesSet.find(i) == edgesSet.end()) {
                cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
неправильно заданы номера дуг\n";
                return false;
            }
        }
    }
    return true;
}

```

```

string findInStr(string const& str, int n) {
    if (str.length() < n) {

```

```

        return str;
    }
    return str.substr(0, n);
}

bool isCyclic(int vertex, const vector<Vert>& edges, set<int>& visited, set<int>&
recStack) {
    visited.insert(vertex);
    recStack.insert(vertex);

    for (const auto& edge : edges) {
        if (edge.a == vertex) {
            int neighbor = edge.b;

            if (visited.find(neighbor) == visited.end()) {
                if (isCyclic(neighbor, edges, visited, recStack)) {
                    return true;
                }
            }
            else if (recStack.find(neighbor) != recStack.end()) {
                return true;
            }
        }
    }

    recStack.erase(vertex);
    return false;
}

bool checkCycle(const Graph& graph, vector<int> &cycleLines) {
    const auto& edges = graph.getEdges();
    set<int> visited;
    set<int> recStack;
    bool haveCycle = false;
    for (const auto& edge : edges) {
        int vertex = edge.a;
        int lineNum = edge.l;
        if (visited.find(vertex) == visited.end()) {
            if (isCyclic(vertex, edges, visited, recStack)) {
                cycleLines.push_back(lineNum);
                haveCycle = true;
            }
        }
    }
    if (haveCycle)
        return true;
    return false;
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    Graph graph;
    set<int> uniqueA;
    set<int> uniqueB;
    string input1, input2, output1, output2;
    for (int i = 0; argv[i]; i++)
    {
        string checkStr = string(argv[i]);
        if (checkStr.length() > 4) {
            string ifStr1 = findInStr(checkStr, 7);
            string ifStr2 = findInStr(checkStr, 8);
            string subStr = checkStr.substr(7, checkStr.length());
            string subStr2 = checkStr.substr(8, checkStr.length());

```

```

        if (ifStr1 == "input1=") {
            input1 = subStr;
        }
        if (ifStr1 == "input2=") {
            input2 = subStr;
        }
        if (ifStr2 == "output1=") {
            output1 = subStr2;
        }
        if (ifStr2 == "output2=") {
            output2 = subStr2;
        }
    }
}
bool isGoodResult = fileToStruct(input1, graph);
if (!checkNumbers(graph, uniqueA, uniqueB)) {
    isGoodResult = false;
}
vector<int> cycleLines;
if (checkCycle(graph, cycleLines)) {
    for (auto i: cycleLines)
        cout << "Ошибка в строке " << i << ": в графе существует цикл\n";
    return 0;
}
if (isGoodResult) {
    findPrefix(graph, output1);
}
return 0;
}
}

```

ПРИЛОЖЕНИЕ В

Листинг программы третьего задания nntask3.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cctype>
#include <set>
#include <map>
#include <stack>
#include <algorithm>

using namespace std;

struct Vert {
    int a, b, n, l;
};

class Graph {
private:
    vector<Vert> edges;
    set<pair<int, int>> uniqueEdgesAB;
    set<pair<int, int>> uniqueEdgesBN;
    int lineNum;
public:
    int addEdge(const Vert& edge) {
        if (uniqueEdgesAB.count({ edge.a, edge.b }) > 0) {
            return -1;
        }
        if (!uniqueEdgesBN.insert({ edge.b, edge.n }).second) {
            return -2;
        }
        uniqueEdgesAB.insert({ edge.a, edge.b });
        edges.push_back(edge);
        lineNum = edge.l;
        return 1;
    }

    const vector<Vert>& getEdges() const {
        return edges;
    }
    const int getLineNumber() const {
        return lineNum;
    }

    int findSink() const {
        set<int> allVertices;
        set<int> outVertices;

        for (const auto& edge : edges) {
            allVertices.insert(edge.a);
            allVertices.insert(edge.b);
            outVertices.insert(edge.a);
        }

        for (int vertex : allVertices) {
            if (outVertices.find(vertex) == outVertices.end()) {
                return vertex;
            }
        }
    }
}
```

```

        return -1;
    }

    int edgesCount() const {
        int maxElement = 1;
        for (const auto& edge : edges) {
            if (edge.a > maxElement)
                maxElement = edge.a;
            if (edge.b > maxElement)
                maxElement = edge.b;
        }
        return maxElement;
    }
};

void buildParents(const Graph& graph, vector <vector <int>>& parents, const vector<Vert>&
edges) {
    for (const auto& edge : edges)
        parents[edge.b - 1].push_back(edge.a);
    return;
}

void printPrefix(vector<vector<int>>& parents, int startVertex, ofstream& outFile) {
    outFile << startVertex;
    vector<int> checkedVertex = parents[startVertex - 1];
    if (!checkedVertex.empty()) {
        outFile << "(";
        for (int i = 0; i < checkedVertex.size(); ++i) {
            printPrefix(parents, checkedVertex[i], outFile);
            if (i < checkedVertex.size() - 1)
                outFile << ",";
        }
        outFile << ")";
    }
}

void findPrefix(const Graph& graph, const string& fileName) {
    const vector<Vert>& edges = graph.getEdges();
    set<int> visitedVertices;

    int startVertex = graph.findSink();

    int maxEl = graph.edgesCount();
    vector<int> dop;
    vector <vector <int>> parents(maxEl, dop);
    buildParents(graph, parents, edges);

    ofstream outFile(fileName, ios::out | ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    printPrefix(parents, startVertex, outFile);

    outFile.close();
}

bool fileToStruct(const string& fileName, Graph& graph) {
    bool isGoodResult = true;
    ifstream file(fileName);
    if (!file.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return false;
    }
}

```

```

}

string line;
size_t lineNumber = 0;

while (getline(file, line)) {
    lineNumber++;
    istringstream iss(line);

    char nextChar;
    bool findOpen = true;
    bool errorPrint1 = true;
    while (iss >> nextChar) {
        if (nextChar != '(' && findOpen) {
            if (errorPrint1) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
            }
            while (nextChar != '(') {
                if (!(iss >> nextChar))
                    return false;
            }
        }
        errorPrint1 = false;
        findOpen = false;
        Vert edge;
        string a, b, n;
        if (nextChar == '(') {

            while (iss >> nextChar && isdigit(nextChar)) {
                a += nextChar;
            }

            if (a == "0") {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  вершины 0
быть не может\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            if (nextChar == '\n') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                continue;
            }
            if (isalpha(nextChar)) {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  некорректные
данные (введены буквы вместо цифр)\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            if (nextChar != ',') {
                cout << "Ошибка в строке " + to_string(lineNumber) + ":  неправильно
задана компонента. Формат: (a, b, n)\n";
                isGoodResult = false;
                findOpen = true;
                continue;
            }
            iss >> ws >> nextChar;
            while (iss && isdigit(nextChar)) {
                b += nextChar;
            }
        }
    }
}

```

```

        iss >> nextChar;
    }

    if (b == "0") {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": вершины 0
быть не может\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (nextChar == '\n') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        continue;
    }

    if (isalpha(nextChar)) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (nextChar != ',') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    iss >> ws >> nextChar;
    while (iss && nextChar != ')' && isdigit(nextChar)) {
        n += nextChar;
        iss >> nextChar;
    }

    if (isalpha(nextChar)) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": некорректные
данные (введены буквы вместо цифр)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (nextChar == '\n') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        continue;
    }

    if (nextChar != ')') {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
        findOpen = true;
        continue;
    }

    if (!a.empty() && !b.empty() && !n.empty()) {
        edge.a = stoi(a);
        edge.b = stoi(b);
        edge.n = stoi(n);
        edge.l = lineNumber;
    }

```



```

        int errorCode = graph.addEdge(edge);
        if (errorCode == -1) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ":
повторяющаяся дуга " + a + " -> " + b << "\n";
            isGoodResult = false;
        }
        if (errorCode == -2) {
            cout << "Ошибка в строке " + to_string(lineNumber) + ": дуга с
номером " + n + " в вершину " + b + " уже существует\n";
            isGoodResult = false;
        }
    }
}
else {
    cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
    isGoodResult = false;
}

    iss >> ws >> nextChar;
    if (nextChar != ',' && nextChar != '\n' && !iss.eof()) {
        cout << "Ошибка в строке " + to_string(lineNumber) + ": неправильно
задана компонента. Формат: (a, b, n)\n";
        isGoodResult = false;
    }
    if (iss.eof())
        break;
}
}
return isGoodResult;
}

```

```

bool checkNumbers(const Graph& graph, set<int>& uniqueA, set<int>& uniqueB) {
    const vector<Vert>& edges = graph.getEdges();
    map<int, set<int>> incomingEdges;

    for (const auto& edge : edges) {
        uniqueB.insert(edge.b);
        uniqueA.insert(edge.a);
        incomingEdges[edge.b].insert(edge.n);
    }
    for (const auto& vertex : uniqueA) {
        if (vertex > *max_element(uniqueB.begin(), uniqueB.end())) {
            cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
Неправильная нумерация вершин. Номер вершины а больше количества вершин.\n";
            return false;
        }
    }

    for (const auto& entry : incomingEdges) {
        const auto& edgesSet = entry.second;
        for (int i = 1; i <= *max_element(edgesSet.begin(), edgesSet.end()); ++i) {
            if (edgesSet.find(i) == edgesSet.end()) {
                cout << "Ошибка в строке " + to_string(graph.getLineNumber()) + ":
неправильно заданы номера дуг\n";
                return false;
            }
        }
    }
    return true;
}

string findInStr(string const& str, int n) {

```

```

    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

bool isCyclic(int vertex, const vector<Vert>& edges, set<int>& visited, set<int>&
recStack) {
    visited.insert(vertex);
    recStack.insert(vertex);

    for (const auto& edge : edges) {
        if (edge.a == vertex) {
            int neighbor = edge.b;

            if (visited.find(neighbor) == visited.end()) {
                if (isCyclic(neighbor, edges, visited, recStack)) {
                    return true;
                }
            }
            else if (recStack.find(neighbor) != recStack.end()) {
                return true;
            }
        }
    }

    recStack.erase(vertex);
    return false;
}

bool checkCycle(const Graph& graph, vector<int>& cycleLines) {
    const auto& edges = graph.getEdges();
    set<int> visited;
    set<int> recStack;
    bool haveCycle = false;
    for (const auto& edge : edges) {
        int vertex = edge.a;
        int lineNum = edge.l;
        if (visited.find(vertex) == visited.end()) {
            if (isCyclic(vertex, edges, visited, recStack)) {
                cycleLines.push_back(lineNum);
                haveCycle = true;
            }
        }
    }
    if (haveCycle)
        return true;
    return false;
}

void cmdFileToVec(vector<string>& graphCommands, const string& fileName) {
    ifstream inputFile(fileName);

    if (!inputFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    string line;
    while (getline(inputFile, line)) {
        istringstream ss(line);
        string command;

        ss.ignore(numeric_limits<streamsize>::max(), ':');

```

```

        ss >> ws;
        getline(ss, command);

        graphCommands.push_back(command);
    }

    inputFile.close();
    return;
}

string prefixFileToStr(const string& fileName) {
    ifstream inputFile(fileName);

    if (!inputFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return "";
    }
    string resStr;
    getline(inputFile, resStr);

    inputFile.close();
    return resStr;
}

void replaceNumbers(string& prefixStr, const vector<string>& cmdCommands) {
    istringstream ss(prefixStr);
    char elem;
    while (ss >> elem) {
        if (isdigit(elem)) {
            int index = elem - '0';
            prefixStr.replace(prefixStr.find(elem), 1, cmdCommands[index - 1]);
        }
    }
}

bool isOperator(char elem) {
    return elem == '+' || elem == '*' || elem == 'e';
}

double PrefixResult(const string& prefixCode, const vector<string>& cmdCommands) {
    locale::global(locale("C"));
    stack<double> stack;

    for (int i = prefixCode.size() - 1; i >= 0; --i) {
        char elem = prefixCode[i];
        if (isdigit(elem) || (elem == '-' && i > 0 && (isdigit(prefixCode[i - 1]) ||
prefixCode[i - 1] == '.'))) {
            string number;
            while (i >= 0 && (isdigit(prefixCode[i]) || prefixCode[i] == '.')) {
                number = prefixCode[i] + number;
                --i;
            }
            int sign = 1;
            if (prefixCode[i] == '-') {
                sign = -1;
            }
            stack.push(sign * stod(number));
        }
        else if (isOperator(elem)) {
            if (elem == 'e') {
                double operand1 = stack.top();
                stack.pop();
                stack.push(exp(operand1));
            }
        }
    }
}

```

```

        else {
            double dig1 = stack.top();
            stack.pop();
            double dig2 = stack.top();
            stack.pop();
            if (elem == '+')
                stack.push(dig1 + dig2);
            else if (elem == '*')
                stack.push(dig1 * dig2);
        }
    }
}
return stack.top();
}

void resToFile(double result, const string& fileName) {
    ofstream outFile(fileName, ios::out | ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    outFile << fixed << result;

    outFile.close();
}

void findPrefixResult(const string& input2, const string& output1, const string& output2)
{
    vector<string> cmdCommands;
    cmdFileToVec(cmdCommands, input2);

    string prefixStr = prefixFileToStr(output1);
    replaceNumbers(prefixStr, cmdCommands);

    double result = PrefixResult(prefixStr, cmdCommands);
    resToFile(result, output2);
    return;
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    Graph graph;
    set<int> uniqueA;
    set<int> uniqueB;
    string input1, input2, output1, output2;
    for (int i = 0; argv[i]; i++)
    {
        string checkStr = string(argv[i]);
        if (checkStr.length() > 4) {
            string ifStr1 = findInStr(checkStr, 7);
            string ifStr2 = findInStr(checkStr, 8);
            string subStr = checkStr.substr(7, checkStr.length());
            string subStr2 = checkStr.substr(8, checkStr.length());
            if (ifStr1 == "input1=") {
                input1 = subStr;
            }
            if (ifStr1 == "input2=") {
                input2 = subStr;
            }
            if (ifStr2 == "output1=") {
                output1 = subStr2;
            }
            if (ifStr2 == "output2=") {

```

```

        output2 = subStr2;
    }
}
bool isGoodResult = fileToStruct(input1, graph);
if (!checkNumbers(graph, uniqueA, uniqueB)) {
    isGoodResult = false;
}
vector<int> cycleLines;
if (checkCycle(graph, cycleLines)) {
    for (auto i : cycleLines)
        cout << "Ошибка в строке " << i << ": в графе существует цикл\n";
    return 0;
}
if (isGoodResult) {
    findPrefix(graph, output1);
    findPrefixResult(input2, output1, output2);
}
return 0;
}

```

ПРИЛОЖЕНИЕ Г

Листинг программы четвертого задания nntask4.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cctype>
#include <iomanip>
#include <algorithm>

using namespace std;

class Matrix {
private:
    vector< vector <vector<int>>> matrix;
    vector<int> vector;

public:
    Matrix(const std::vector< std::vector<std::vector<int>>>& matrixData, const
std::vector<int>& vectorData) : matrix(matrixData), vector(vectorData) {}

    std::vector< std::vector <std::vector<int>>> getMatrix() {
        return matrix;
    }
    std::vector<int> getVector() {
        return vector;
    }
};

Matrix readFromFile(const string& matrixFileName, const string& vectorFileName) {
    vector <vector< vector<int>>> matrixData;
    vector<int> vectorData;

    ifstream vectorFile(vectorFileName);
    if (!vectorFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        exit(EXIT_FAILURE);
    }

    int rowSize;
    string line;
    getline(vectorFile, line);
    stringstream ss(line);
    int value;
    while (ss >> value) {
        vectorData.push_back(value);
    }
    vectorFile.close();
    rowSize = vectorData.size();

    ifstream matrixFile(matrixFileName);
    if (!matrixFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        exit(EXIT_FAILURE);
    }

    int strNumber = 1;

    while (getline(matrixFile, line)) {
```

```

vector<vector<int>>> strM;
vector<int> row;
stringstream ss(line);
char ch;
while (ss >> ch) {
    if (isdigit(ch)) {
        ss.putback(ch);
        int value;
        ss >> value;
        row.push_back(value);
    }
    if (ch == ']') {
        strM.push_back(row);
        if (rowSize != row.size()) {
            cout << "Ошибка в строке " << strNumber << ": некорректное число
компонент нейронов\n";
            exit(EXIT_FAILURE);
        }
        row.clear();
    }
    if (ch != '[' && ch != '[' && ch != ',' && ch != ' ' && !isdigit(ch)) {
        cout << "Ошибка в строке " << strNumber << ": неправильно задан вес
матрицы межнейронной связи\n";
        exit(EXIT_FAILURE);
    }
}
matrixData.push_back(strM);
strNumber++;
}
matrixFile.close();

Matrix myMatrix = Matrix(matrixData, vectorData);
return myMatrix;
}

void matrixToJSON(Matrix matrix, const string& fileName) {
    vector< vector<vector<int>>>> matrixVec = matrix.getMatrix();
    ofstream jsonFile(fileName);
    if (!jsonFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        exit(EXIT_FAILURE);
    }
    jsonFile << "[\n";

    for (int i = 0; i < matrixVec.size(); ++i) {
        jsonFile << "\t\n";
        jsonFile << "\t\t\n";
        for (int j = 0; j < matrixVec[i].size(); ++j) {
            jsonFile << "\t\t\t\n";
            for (int k = 0; k < matrixVec[i][j].size(); ++k) {
                jsonFile << "\t\t\t\t" << matrixVec[i][j][k];
                if (k < matrixVec[i][j].size() - 1) {
                    jsonFile << ",";
                }
            }
            jsonFile << "\n";
        }
        jsonFile << "\t\t\t";
        if (j + 1 < matrixVec[i].size()) {
            jsonFile << ",";
        }
        jsonFile << "\n";
    }
    jsonFile << "\t\t]";
    jsonFile << "\n";
}

```

```

        jsonFile << "\t]";
        if (i + 1 < matrixVec.size()) {
            jsonFile << ",";
        }
        jsonFile << "\n";
    }

    jsonFile << "]\n";

    jsonFile.close();

    return;
}

vector<double> resFunc(Matrix matrix) {
    cout << setprecision(16);
    vector< vector< vector<int>>> matrixVec = matrix.getMatrix();
    vector<int> vectorVec = matrix.getVector();
    vector<double> vecDouble;
    for (int i = 0; i < vectorVec.size(); i++) {
        vecDouble.push_back(static_cast<double>(vectorVec[i]));
    }
    vector<vector<double>> newM;

    for (const auto& layer : matrixVec) {
        vector<double> vecFor;
        for (const auto& n : layer) {
            double res = 0;
            for (int i = 0; i < vecDouble.size(); ++i) {
                res += static_cast<double>(n[i]) * vecDouble[i];
            }
            res /= (1 + abs(res));
            vecFor.push_back(res);
        }
        newM.push_back(vecFor);
        vecDouble = vecFor;
    }

    return newM[newM.size() - 1];
}

string findInStr(string const& str, int n) {
    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

void resToFile(vector<double> resultMatrix, const string& fileName) {
    ofstream outFile(fileName, ios::out | ios::trunc);
    outFile << setprecision(16);
    if (!outFile.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    for (int i = 0; i < resultMatrix.size(); i++)
        outFile << resultMatrix[i] << " ";

    outFile.close();
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");

```



```

string input1, input2, output1, output2;
for (int i = 0; argv[i]; i++)
{
    string checkStr = string(argv[i]);
    if (checkStr.length() > 4) {
        string ifStr1 = findInStr(checkStr, 7);
        string ifStr2 = findInStr(checkStr, 8);
        string subStr = checkStr.substr(7, checkStr.length());
        string subStr2 = checkStr.substr(8, checkStr.length());
        if (ifStr1 == "input1=") {
            input1 = subStr;
        }
        if (ifStr1 == "input2=") {
            input2 = subStr;
        }
        if (ifStr2 == "output1=") {
            output1 = subStr2;
        }
        if (ifStr2 == "output2=") {
            output2 = subStr2;
        }
    }
}

Matrix myMatrix = readFromFile(input1, input2);

matrixToJSON(myMatrix, output1);

vector<double> resultMatrix = resFunc(myMatrix);

resToFile(resultMatrix, output2);

return 0;
}

```

ПРИЛОЖЕНИЕ Д

Листинг программы пятого задания nntask5.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <sstream>
#include <cctype>
#include <iomanip>
#include <algorithm>

using namespace std;

vector< vector< vector<double>>> jsonToVecs(const string& jsonString, const string&
filename) {
    vector< vector< vector<double>>> result;

    int pos = 1;

    while (pos < jsonString.length()) {
        pos = jsonString.find('[', pos);
        if (pos == string::npos) {
            break;
        }
        vector< vector<double>> svv;
        pos++;

        while (true) {
            pos = jsonString.find('[', pos);
            if (pos == string::npos) {
                break;
            }
            vector<double> sv;
            pos++;

            while (true) {
                double value;
                stringstream ss;

                pos = jsonString.find_first_of("0123456789.-]", pos);
                if (pos == string::npos || jsonString[pos] == ']') {
                    break;
                }

                ss << jsonString[pos];
                pos++;
                while (isdigit(jsonString[pos]) || jsonString[pos] == '.') {
                    ss << jsonString[pos];
                    pos++;
                }

                ss >> value;
                sv.push_back(value);
                if (jsonString[pos] == '[') {
                    cerr << "Файл " << filename << " имеет некорректное содержимое\n";
                    exit(EXIT_FAILURE);
                }
                pos = jsonString.find_first_of(",]", pos);
            }

            if (pos == string::npos || jsonString[pos] == ']') {
```

```

        break;
    }

    pos++;
}

svv.push_back(sv);
pos++;
while (jsonString[pos] == ' ' || jsonString[pos] == '\n')
    pos++;
if (jsonString[pos] == ']') {
    break;
}
if (jsonString[pos] == '[') {
    cerr << "Файл " << filename << " имеет некорректное содержимое\n";
    exit(EXIT_FAILURE);
}
pos = jsonString.find(',', pos);
pos++;
}

result.push_back(svv);
pos++;
if (pos == string::npos) {
    break;
}
while (jsonString[pos] == ' ' || jsonString[pos] == '\n') {
    pos++;
}
if (jsonString[pos] == ']') {
    break;
}
if (jsonString[pos] == '[') {
    cerr << "Файл " << filename << " имеет некорректное содержимое\n";
    exit(EXIT_FAILURE);
}

pos = jsonString.find(',', pos);
pos++;
}
return result;
}

vector<double> parseVector(string line, string key) {
    vector<double> result;
    int startPos = line.find(key) + key.length();
    int endPos = line.find("]", startPos);
    string values = line.substr(startPos + 1, endPos - startPos);
    istringstream iss(values);
    double value;
    while (iss >> value) {
        result.push_back(value);
        if (iss.peek() == ',') {
            iss.ignore();
        }
    }
    return result;
}

class NNClass {
private:
    vector<int> nCount;
    vector<vector<vector<double>>> nnWeights;
public:

```

```

        NNClass(const vector<vector<vector<double>>>& vec);
        void nnTraining(const string& trainingFile, int n, const string& outputFile);
};

vector<vector<double>> matrTranspose(const vector<vector<double>> m) {
    int rows = m.size();
    int cols = m[0].size();
    vector<vector<double>> result(cols, vector<double>(rows, 0.0));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[j][i] = m[i][j];
        }
    }
    return result;
}

vector< vector<double>> matrMultiplication(const vector< vector<double>>& m1, const
vector< vector<double>>& m2) {
    int rows1 = m1.size();
    int cols1 = m1[0].size();
    int rows2 = m2.size();
    int cols2 = m2[0].size();
    vector< vector<double>> result(rows1, vector<double>(cols2, 0.0));
    for (int i = 0; i < rows1; ++i) {
        for (int j = 0; j < cols2; ++j) {
            for (int k = 0; k < cols1; ++k) {
                result[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
    return result;
}

NNClass::NNClass(const vector< vector< vector<double>>>& vec) {
    this->nCount = vector<int>();
    this->nnWeights = vector< vector< vector<double>>>();

    for (const auto& layer : vec) {
        vector< vector<double>> weights;
        for (const auto& neuron : layer) {
            weights.push_back(neuron);
        }

        if (this->nCount.empty()) {
            this->nCount.push_back(weights[0].size());
        }
        else if (this->nCount.back() != weights[0].size()) {
            cerr << "Некорректное количество компонентов нейрона в слое\n";
            exit(EXIT_FAILURE);
        }

        this->nCount.push_back(weights.size());
        this->nnWeights.push_back(matrTranspose(weights));
    }
}

void NNClass::nnTraining(const string& trainingFile, int n, const string& outputFile) {
    ifstream fs(trainingFile);
    if (!fs.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return;
    }

    pair <vector<double>, vector<double>> trData;

```

```

vector<pair <vector<double>, vector<double>>> nnTrainingVec;
string line;
while (getline(fs, line)) {
    vector<double> file_i = parseVector(line, "\\i\\:");
    vector<double> file_o = parseVector(line, "\\o\\:");

    if (file_i.empty() || file_o.empty()) {
        cerr << "Некорректное содержимое файла: " << trainingFile << endl;
        exit(EXIT_FAILURE);
    }

    nnTrainingVec.push_back(make_pair(file_i, file_o));
}

fs.close();

if (nnTrainingVec.size() == 0) {
    cerr << "Некорректное содержимое файла: " << trainingFile << endl;
    return;
}

int trainingDataIndex = 0;
double k = 0.01;

ofstream outFile(outputFile, ios::out, ios::trunc);
if (!outFile.is_open()) {
    cerr << "Ошибка открытия файла\n";
    exit(EXIT_FAILURE);
}
outFile << setprecision(16);

for (int iter = 1; iter <= n; iter++) {
    auto funcAct = [](double x) -> double {
        return x / (1 + abs(x));
    };
    auto funcActProizv = [](double x) -> double {
        double absX = abs(x);
        return 1.0 / ((1 + absX) * (1 + absX));
    };
    pair <double, double> nValue;
    vector< vector<pair <double, double>>> nValues;
    pair <vector<double>, vector<double>> thisTrainingData =
nnTrainingVec[trainingDataIndex];
    vector<double> input = thisTrainingData.first;

    vector<pair <double, double>> layerNValues;
    for (double inp1 : input) {
        layerNValues.push_back(make_pair(inp1, inp1));
    }
    nValues.push_back(layerNValues);

    vector< vector<double>> resMult;
    resMult.push_back(input);

    for (int layerIndex = 0; layerIndex < nnWeights.size(); layerIndex++) {
        layerNValues.clear();
        resMult = matrMultiplication(resMult, nnWeights[layerIndex]);

        for (int i = 0; i < resMult[0].size(); i++) {
            pair <double, double> pairNVal;
            pairNVal.first = resMult[0][i];
            pairNVal.second = funcAct(pairNVal.first);
            resMult[0][i] = pairNVal.second;
            layerNValues.push_back(pairNVal);
        }
    }
}

```

```

    }
    nValues.push_back(layerNValues);
}

vector< vector< vector<double>>> weightUpdates;
vector<double> nLayerErr;

int i = nValues.size() - 1;
for (int m = 0; m < nValues[i].size(); m++)
    nLayerErr.push_back((thisTrainingData.second[m] - nValues[i][m].second) *
funcActProizv(nValues[i][m].first));

vector< vector<double>> wu2;
vector< vector<double>> wi = nnWeights[i - 1];

for (int m = 0; m < wi[0].size(); m++) {
    vector<double> deltaWij;
    for (int z = 0; z < wi.size(); z++) {
        deltaWij.push_back(k * nLayerErr[m] * nValues[i - 1][z].second);
    }
    wu2.push_back(deltaWij);
}
weightUpdates.push_back(matrTranspose(wu2));

for (int layerIndex = i - 1; layerIndex >= 1; layerIndex--) {
    wi = nnWeights[layerIndex];
    vector<double> nLayerErrIn;

    for (int m = 0; m < wi.size(); m++) {
        double nLayerErrIJ = 0;
        for (int z = 0; z < wi[m].size(); z++) {
            nLayerErrIJ += nLayerErr[z] * wi[m][z];
        }
        nLayerErrIn.push_back(nLayerErrIJ);
    }

    nLayerErr.clear();

    for (int m = 0; m < nValues[layerIndex].size(); m++) {
        nLayerErr.push_back(nLayerErrIn[m] *
funcActProizv(nValues[layerIndex][m].first));
    }

    wi = nnWeights[layerIndex - 1];
    wu2.clear();

    for (int i1 = 0; i1 < wi.size(); i1++) {
        vector<double> wuIJ;
        for (int i2 = 0; i2 < wi[i1].size(); i2++) {
            wuIJ.push_back(k * nLayerErr[i2] * nValues[layerIndex -
1][i1].second);
        }
        wu2.push_back(wuIJ);
    }

    weightUpdates.push_back(wu2);
}

for (int i1 = 0; i1 < nnWeights.size(); i1++) {
    wi = nnWeights[i1];
    vector< vector<double>> uwVec = weightUpdates[nnWeights.size() - i1 - 1];

    for (int i2 = 0; i2 < wi.size(); i2++) {
        vector<double> uwIJ = wi[i2];

```

```

        const vector<double>& uwIJvec = uwVec[i2];
        for (int i3 = 0; i3 < uwIJ.size(); i3++) {
            uwIJ[i3] += uwIJvec[i3];
        }
    }
}

outFile << "" << iter << " : ";
for (int m = 0; m < nLayerErr.size(); m++) {
    if (nLayerErr[m] == 0)
        outFile << "0.0 ";
    else
        outFile << nLayerErr[m] << " ";
}
outFile << "\n\n";

trainingDataIndex = (trainingDataIndex + 1) % nnTrainingVec.size();
}
outFile.close();
}

string findInStr(string const& str, int n) {
    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

vector< vector< vector<double>>> jsonMatrixToVec(const string &filename) {
    vector< vector< vector<double>>> res;
    ifstream file(filename);
    if (!file.is_open()) {
        cerr << "Ошибка открытия файла\n";
        return res;
    }
    string jsonString((istreambuf_iterator<char>(file)), istreambuf_iterator<char>());
    file.close();

    res = jsonToVecs(jsonString, filename);
    return res;
}

int main(int argc, char* argv[]) {
    cout << setprecision(16);
    setlocale(LC_ALL, "rus");
    string input1, input2, output1, output2;
    int n_count;
    for (int i = 0; argv[i]; i++)
    {
        string checkStr = string(argv[i]);
        if (checkStr.length() > 4) {
            string ifStr1 = findInStr(checkStr, 7);
            string ifStr2 = findInStr(checkStr, 8);
            string subStr = checkStr.substr(7, checkStr.length());
            string subStr2 = checkStr.substr(8, checkStr.length());
            if (ifStr1 == "input1=") {
                input1 = subStr;
            }
            if (ifStr1 == "input2=") {
                input2 = subStr;
            }
            if (ifStr2 == "output1=") {
                output1 = subStr2;
            }
        }
    }
}

```

```

        if (ifStr2 == "iterats=") {
            n_count = stoi(subStr2);
        }
    }

    vector< vector< vector<double>>> vec = jsonMatrixToVec(input1);
    NNClass n(vec);
    n.nnTraining(input2, n_count, output1);
    return 0;
}

```