МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | **Факультет прикладной математики и информатики**

Кафедра теоретической и прикладной информатики

Практическое задание № 2

по дисциплине «Компьютерные технологии моделирования и анализа данных»

АЛГОРИТМЫ НУМЕРАЦИИ БАЗИСНЫХ ФУНКЦИЙ В МКЭ

Вариант 1 ПММ-52 КУСАКИН АЛЕКСАНДР

ПММ-52 ЦИРКОВА АЛИНА

ПММ-52 БОРИСОВ ДМИТРИЙ

Преподаватели КОШКИНА ЮЛИЯ ИГОРЕВНА

Новосибирск, 2025

Цель работы

Реализовать алгоритмы нумерации глобальных базисных функций в конечноэлементной сетке.

Задание

Пронумеровать ребра в конечноэлементной сетке.

Реализовать подпрограмму, выдающую по номеру конечного элемента номер его ребер.

Реализовать подпрограмму, выдающую номер ребра по набору из двух узлов.

Реализовать подпрограмму, выдающую по номеру ребра номера соответствующих узлов и конечных элементов, которым принадлежит данное ребро.

Алгоритм

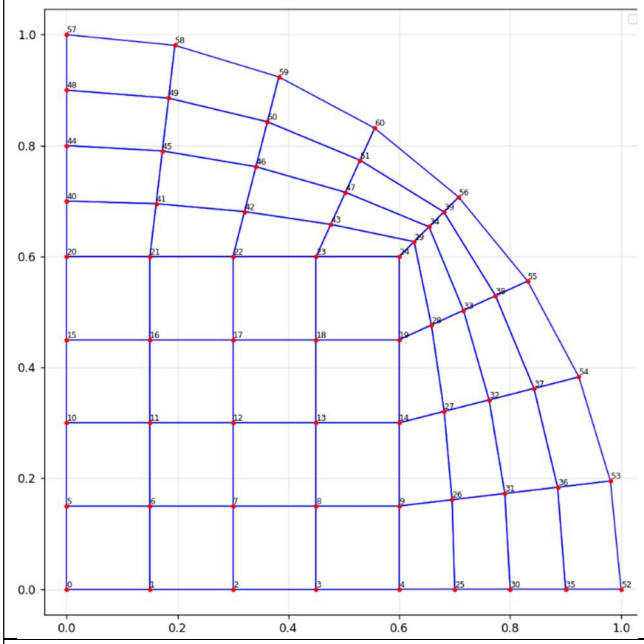
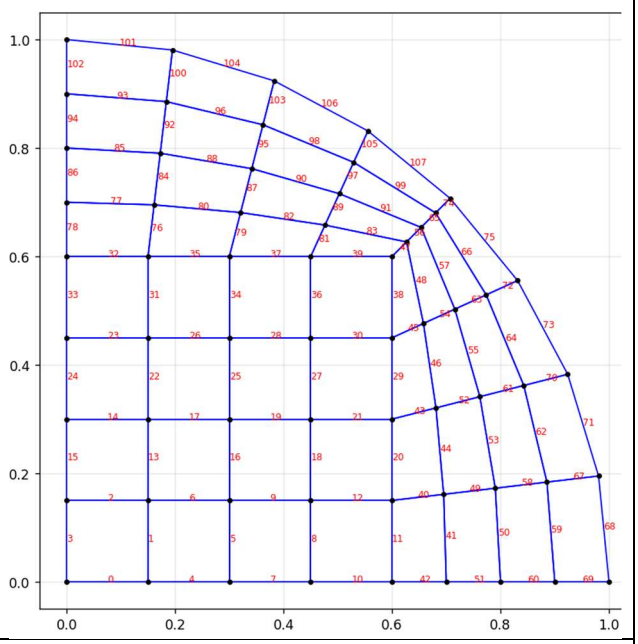
1. Для каждого элемента выполнить:
    a. выписать его ребра
    b. отсортировать пару узлов внутри ребра
    c. если ребра нет в словаре – добавить новое
    d. связать ребро с элементом
2. Для каждого элемента сохранить список его рёбер.
3. Построить:
    - edge_list[edge_id] = (node1, node2)
    - elem_edges[elem_id] = [edge_id, edge_id, edge_id, edge_id]
    - edge_to_elems[edge_id] = [elem_id1, elem_id2]

Нумерация рёбер ведётся в порядке первого появления ребра при обходе элементов.

Результат работы программы

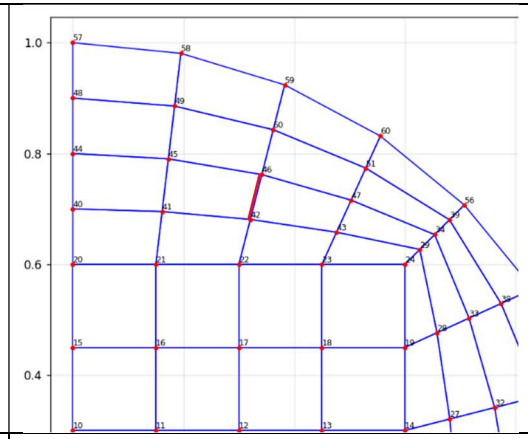
Входные данные для построения сетки:

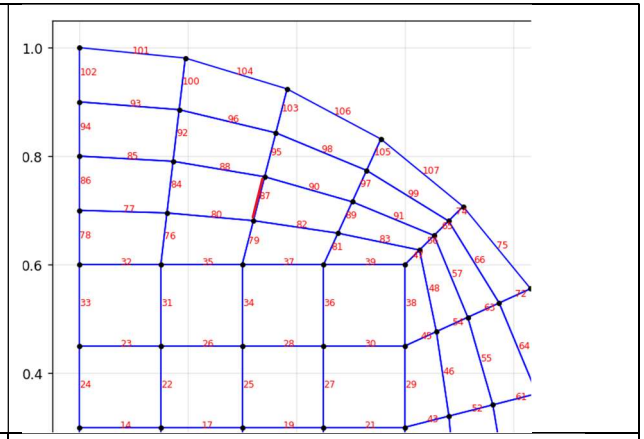
```
radius 1.0 square_size 0.6
5 5
0.0 0.0   0.15 0.0   0.3 0.0   0.45 0.0   0.6 0.0
0.0 0.15  0.15 0.15  0.3 0.15  0.45 0.15  0.6 0.15
0.0 0.3   0.15 0.3   0.3 0.3   0.45 0.3   0.6 0.3
0.0 0.45  0.15 0.45  0.3 0.45  0.45 0.45  0.6 0.45
0.0 0.6   0.15 0.6   0.3 0.6   0.45 0.6   0.6 0.6
1
1 1 4 1 4
```

Построенная сетка	Нумерация ребер
	
Ребро: узел1 узел2 : [элементы]	*Элемент: [ребра]*
0 : 0 1 : [0] 1 : 1 6 : [0, 1] 2 : 5 6 : [0, 4] 3 : 0 5 : [0] 4 : 1 2 : [1] 5 : 2 7 : [1, 2] 6 : 6 7 : [1, 5] 7 : 2 3 : [2] 8 : 3 8 : [2, 3] 9 : 7 8 : [2, 6] 10 : 3 4 : [3] … 100 : 49 58 : [44, 45] 101 : 57 58 : [44] 102 : 48 57 : [44] 103 : 50 59 : [45, 46] 104 : 58 59 : [45] 105 : 51 60 : [46, 47] 106 : 59 60 : [46] 107 : 56 60 : [47]	0: [0, 1, 2, 3] 1: [4, 5, 6, 1] 2: [7, 8, 9, 5] 3: [10, 11, 12, 8] 4: [2, 13, 14, 15] 5: [6, 16, 17, 13] 6: [9, 18, 19, 16] 7: [12, 20, 21, 18] 8: [14, 22, 23, 24] 9: [17, 25, 26, 22] 10: [19, 27, 28, 25] … 40: [85, 92, 93, 94] 41: [88, 95, 96, 92] 42: [90, 97, 98, 95] 43: [91, 65, 99, 97] 44: [93, 100, 101, 102] 45: [96, 103, 104, 100] 46: [98, 105, 106, 103] 47: [99, 74, 107, 105]

Тестирование подпрограмм

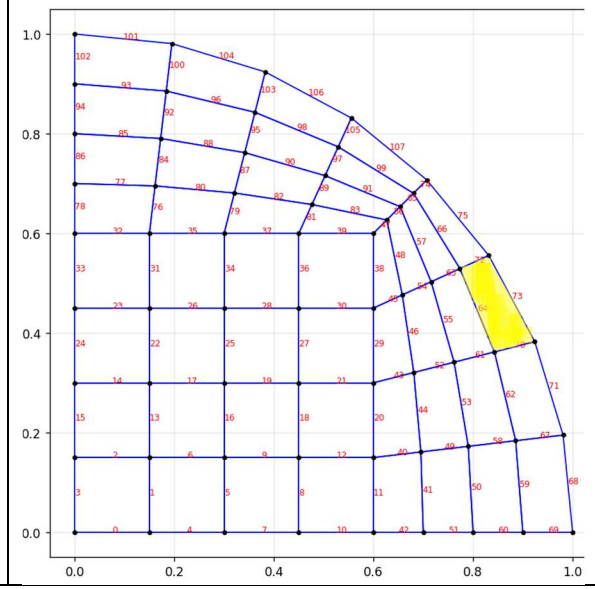
```
=============
1 – Номер ребра по двум узлам
2 – Номера рёбер по номеру элемента
3 – Информация о ребре
4 – Визуализировать сетку с рёбрами
0 – Выход
Введите номер команды: 1
Узел A: 42
Узел B: 46
Ребро между узлами 42 и 46 имеет номер 87
```





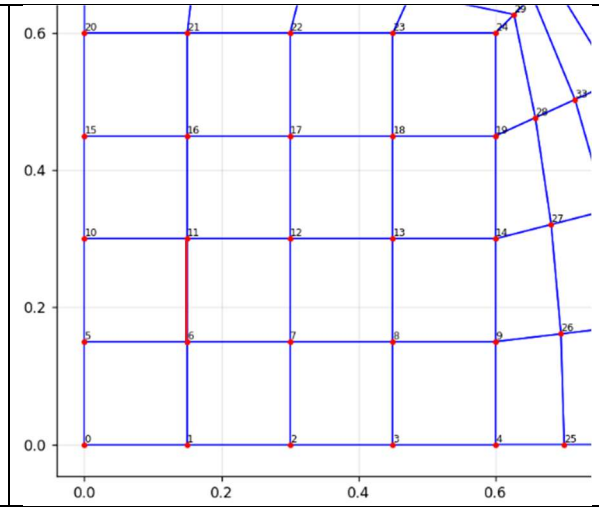
```
=============
1 – Номер ребра по двум узлам
2 – Номера рёбер по номеру элемента
3 – Информация о ребре
4 – Визуализировать сетку с рёбрами
0 – Выход
Введите номер команды: 2
Введите номер элемента: 30
Рёбра элемента: [64, 72, 73, 70]
```

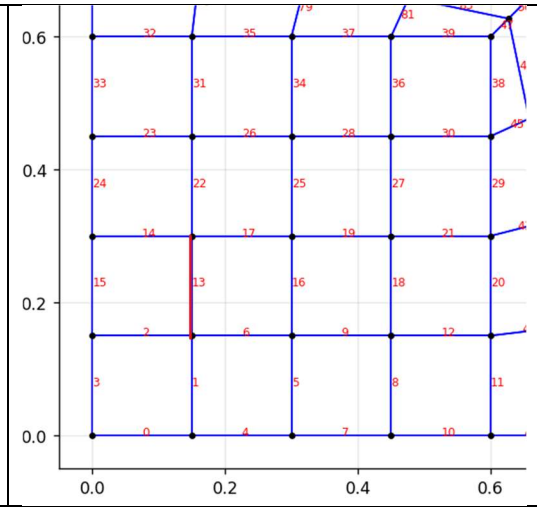
25: [53, 61, 62, 58]
26: [55, 63, 64, 61]
27: [57, 65, 66, 63]
28: [59, 67, 68, 69]
29: [62, 70, 71, 67]
30: [64, 72, 73, 70]
31: [66, 74, 75, 72]
32: [32, 76, 77, 78]
33: [35, 79, 80, 76]
34: [37, 81, 82, 79]
35: [39, 47, 83, 81]



```
============
1 – Номер ребра по двум узлам
2 – Номера рёбер по номеру элемента
3 – Информация о ребре
4 – Визуализировать сетку с рёбрами
0 – Выход
Введите номер команды: 3
Введите номер ребра: 13
Ребро 13: узлы (6, 11)
Принадлежит элементам: [4, 5]
```

4: [2, 13, 14, 15]
5: [6, 16, 17, 13]





Код программы

```
import matplotlib.pyplot as plt

def read_mesh(filename):
    with open(filename, "r") as f:
        lines = [line.strip() for line in f if line.strip()]

    nodes = {}
    elements = []

    i = 0
    if lines[i] != "NODES":
        raise ValueError("Ожидалась секция NODES")
    i += 1

    n_nodes = int(lines[i])
    i += 1

    for _ in range(n_nodes):
        parts = lines[i].split()
        nid = int(parts[0])
        x = float(parts[1])
        y = float(parts[2])
        nodes[nid] = (x, y)
        i += 1

    if lines[i] != "ELEMENTS":
        raise ValueError("Ожидалась секция ELEMENTS")
    i += 1

    n_elems = int(lines[i])
    i += 1

    for _ in range(n_elems):
        parts = lines[i].split()
        elem_id = int(parts[0])
        n1 = int(parts[1])
        n2 = int(parts[2])
        n3 = int(parts[3])
        n4 = int(parts[4])
        mat = int(parts[5])
        elements.append([n1, n2, n3, n4, mat])
        i += 1

    return nodes, elements

def build_edges(elements):
    edge_dict = {}
    edge_list = []
    elem_edges = []
    edge_to_elems = {}
```

```
    for elem_id, elem in enumerate(elements):
        n1, n2, n3, n4, _ = elem

        local_edges = [
            (n1, n2),
            (n2, n3),
            (n3, n4),
            (n4, n1)
        ]

        local_ids = []

        for a, b in local_edges:
            key = tuple(sorted((a, b)))

            if key not in edge_dict:
                edge_dict[key] = len(edge_list)
                edge_list.append(key)
                edge_to_elems[edge_dict[key]] = []

            eid = edge_dict[key]
            local_ids.append(eid)
            edge_to_elems[eid].append(elem_id)

        elem_edges.append(local_ids)

    return edge_list, elem_edges, edge_dict, edge_to_elems

def find_edge_id(edge_dict, a, b):
    key = tuple(sorted((a, b)))
    return edge_dict.get(key, None)

def get_element_edges(elem_edges_table, elem_id):
    return elem_edges_table[elem_id]

def get_edge_info(edge_list, edge_to_elems, edge_id):
    nodes = edge_list[edge_id]
    elems = edge_to_elems[edge_id]
    return nodes, elems

def plot_edges(nodes, elements, edges):
    plt.figure(figsize=(10, 10))

    # элементы
    for elem in elements:
        n1, n2, n3, n4, _ = elem
        xs = [nodes[n][0] for n in (n1, n2, n3, n4, n1)]
        ys = [nodes[n][1] for n in (n1, n2, n3, n4, n1)]
        plt.plot(xs, ys, "b-", linewidth=1)

    # номера ребер
    for eid, (a, b) in enumerate(edges):
        x = (nodes[a][0] + nodes[b][0]) / 2
```

```
        y = (nodes[a][1] + nodes[b][1]) / 2
        plt.text(x, y, str(eid), fontsize=7, color="red")

    # узлы
    for nid, (x, y) in nodes.items():
        plt.plot(x, y, "ko", markersize=3)

    plt.gca().set_aspect("equal", adjustable="box")
    plt.grid(True, alpha=0.3)
    plt.show()

def menu(nodes, elements, edges, elem_edges_table, edge_dict, edge_to_elems):
    while True:
        print("\n=============")
        print("1 – Номер ребра по двум узлам")
        print("2 – Номера рёбер по номеру элемента")
        print("3 – Информация о ребре")
        print("4 – Визуализировать сетку с рёбрами")
        print("0 – Выход")

        cmd = input("Введите номер команды: ").strip()

        if cmd == "1":
            a = int(input("Узел A: "))
            b = int(input("Узел B: "))
            eid = find_edge_id(edge_dict, a, b)
            if eid is None:
                print("Такого ребра нет.")
            else:
                print(f"Ребро между узлами {a} и {b} имеет номер {eid}")

        elif cmd == "2":
            elem_id = int(input("Введите номер элемента: "))
            print("Рёбра элемента:", get_element_edges(elem_edges_table, elem_id))

        elif cmd == "3":
            e = int(input("Введите номер ребра: "))
            (a, b), elems = get_edge_info(edges, edge_to_elems, e)
            print(f"Ребро {e}: узлы ({a}, {b})")
            print("Принадлежит элементам:", elems)

        elif cmd == "4":
            plot_edges(nodes, elements, edges)

        elif cmd == "0":
            print("Выход.")
            break

        else:
            print("Неверная команда!")

if __name__ == "__main__":
    nodes, elements = read_mesh("radial_mesh.txt")
    edges, elem_edges_table, edge_dict, edge_to_elems = build_edges(elements)
```

menu(nodes, elements, edges, elem_edges_table, edge_dict, edge_to_elems)