1. Ход работы

1.1. Код приложения

```
int main(void)
{
from collections import deque

# Граф представлен в виде словаря смежности
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

# Обход графа в глубину (DFS)
def dfs(graph, start):
    visited = set()  # Множество посещенных вершин
    stack = [start]  # Стек для хранения вершин для обхода

    while stack:
        vertex = stack.pop()  # Извлекаем вершину из стека

        if vertex not in visited:
            print(vertex)
            visited.add(vertex)

            # Добавляем смежные вершины в стек
            stack.extend([v for v in graph[vertex] if v not in visited])

# Обход графа в ширину (BFS)
def bfs(graph, start):
    visited = set()  # Множество посещенных вершин
    queue = deque([start])  # Очередь для хранения вершин для обхода

    while queue:
        vertex = queue.popleft()  # Извлекаем вершину из очереди

        if vertex not in visited:
            print(vertex)
            visited.add(vertex)
```

```
            # Добавляем смежные вершины в очередь
            queue.extend([v for v in graph[vertex] if v not in visited])

# Обход графа в глубину (DFS)
print("DFS:")
dfs(graph, 'A')

# Обход графа в ширину (BFS)
print("BFS:")
bfs(graph, 'A')
}

return 0;
}
```

1.2. Теория

1) Поместить узел, с которого начинается поиск, в изначально пустую очередь.

2) Извлечь из начала очереди узел и пометить его как развёрнутый.

• Если узел является целевым узлом, то завершить поиск с результатом «успех».

• В противном случае, в конец очереди добавляются все преемники узла, которые ещё не развёрнуты и не находятся в очереди.

3) Если очередь пуста, то все узлы связного графа были просмотрены, следовательно, целевой узел недостижим из начального; завершить поиск с результатом «неудача».

4) Вернуться к п. 2

2. Вставка изображения результата отладки

Отладка программы представлена на рис.1.

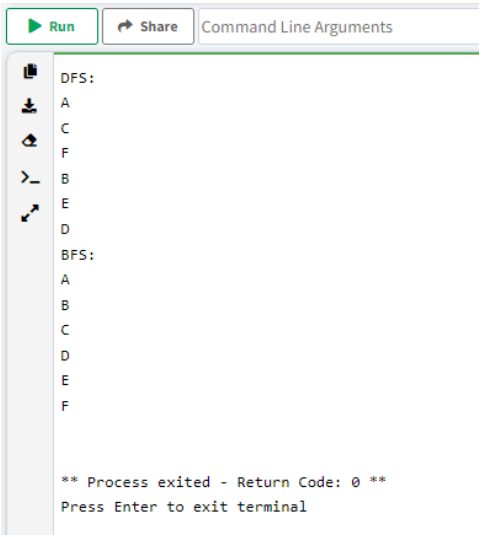

Рис. 1. Отладка программы