



201432360 양정요

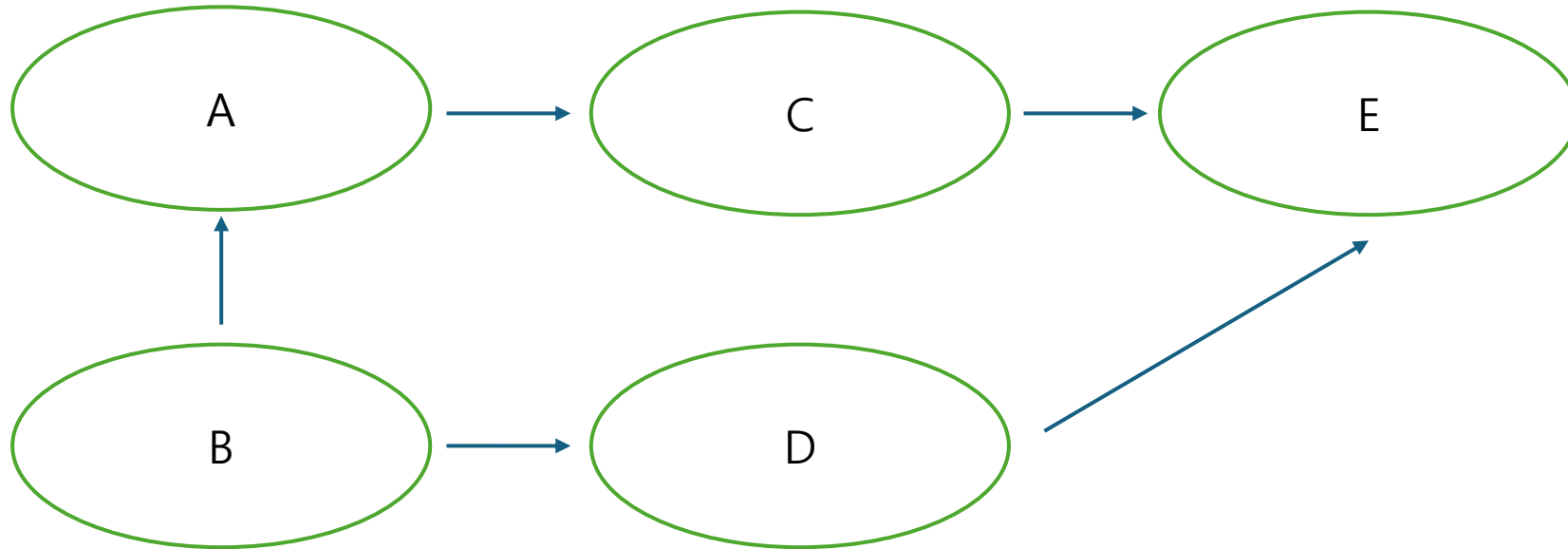
201934208 김대현

201933450 박상원

# Kahn's algorithm

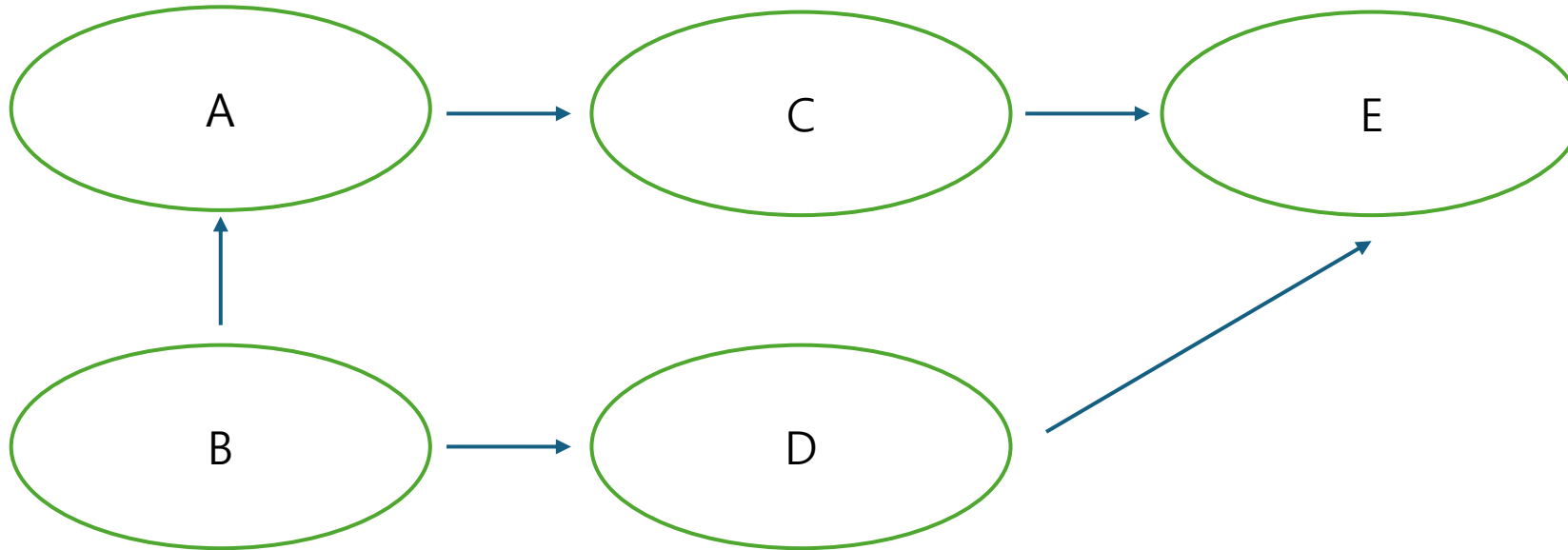
# Topological sorting

- The linear order of vertices in a directed graph



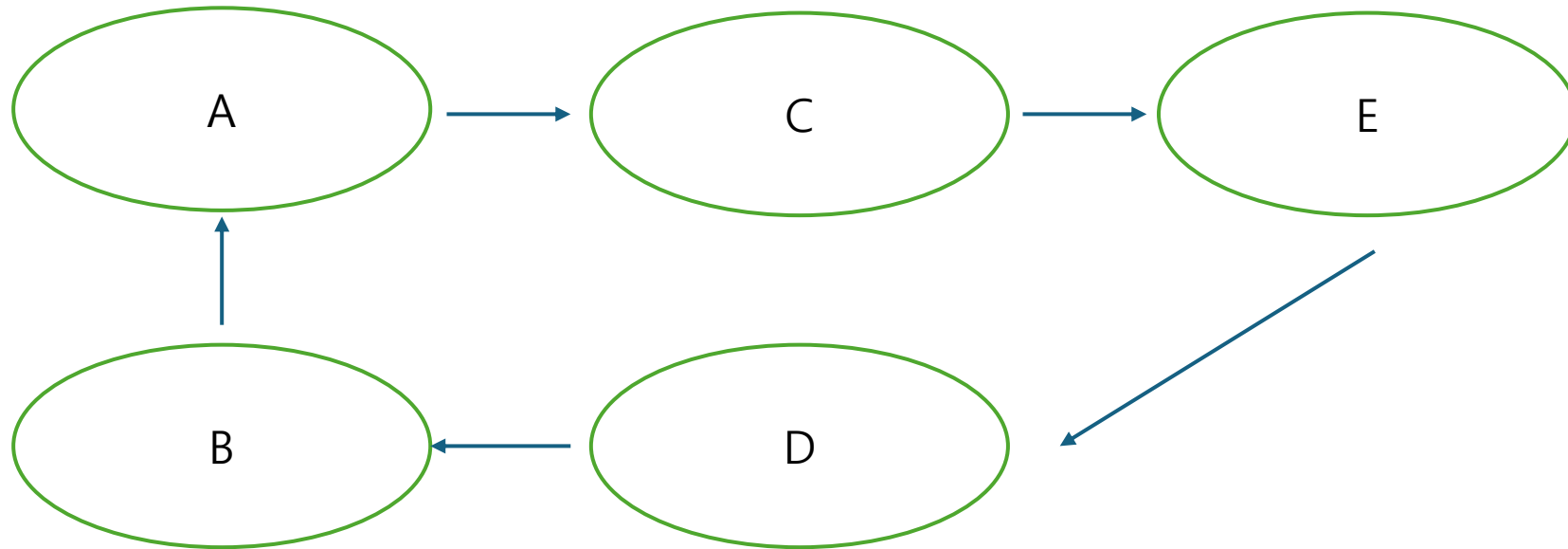
# In-degree

- The in-degree of node  $n$  is the number of edges directed towards node  $n$ .



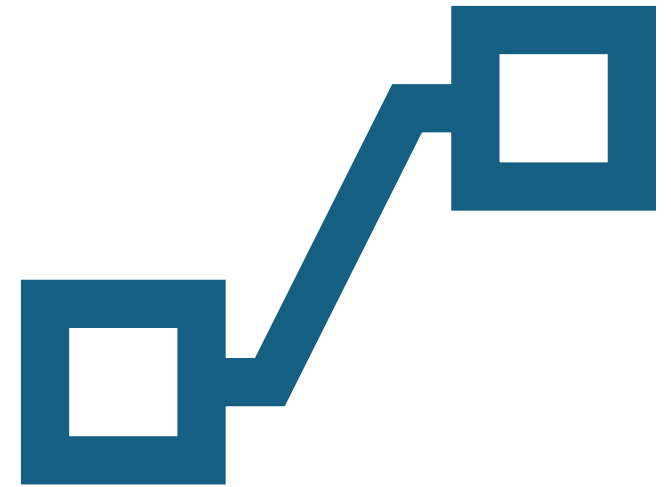
# Not Topological sorting

- If a cycle exists, topological sorting is not possible.



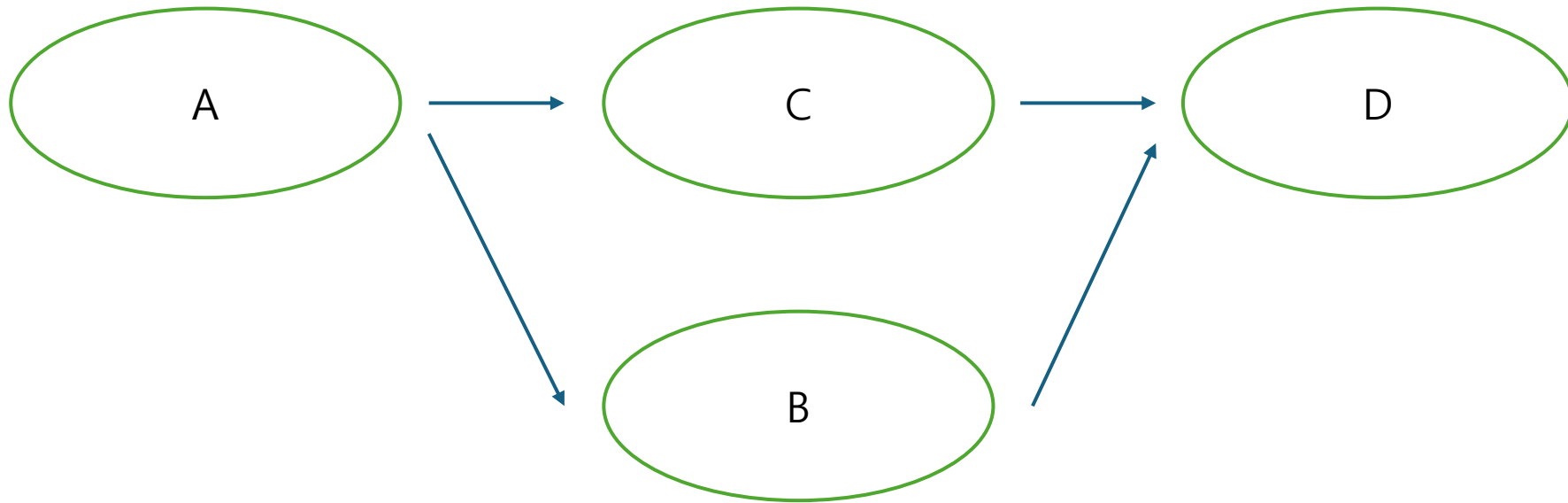
# Kahn's algorithm implementation

1. Compare the in-degrees of each node.
2. Push nodes with an in-degree of 0 into the queue.
3. Pop the queue and add the node to the result list (until the queue is empty).
4. Decrease the in-degree of the nodes connected to the popped node by 1.



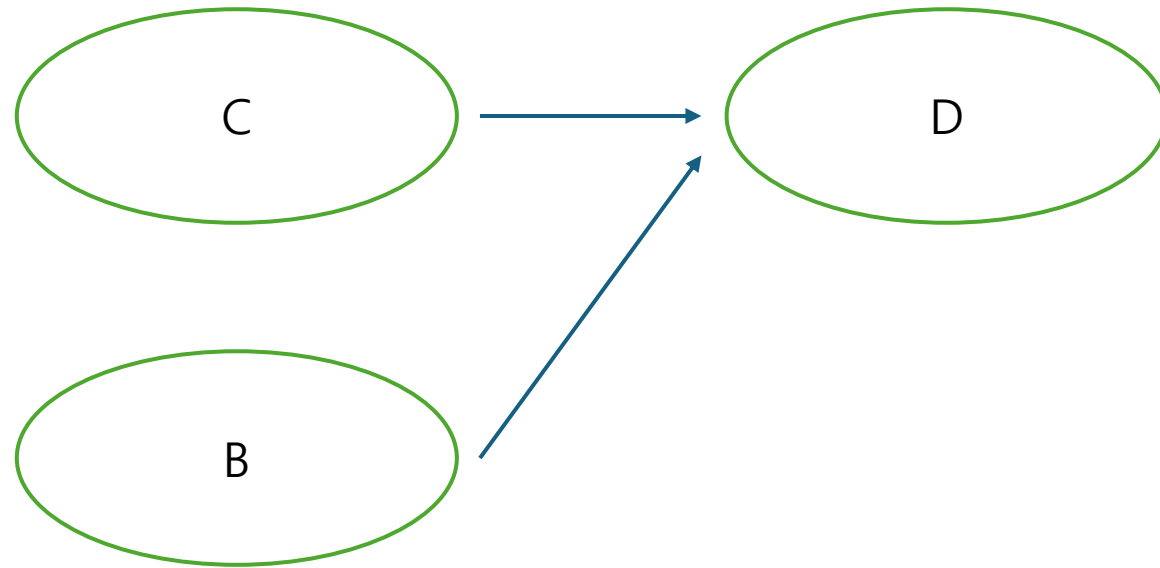
# Example

- A: 0, B: 1, C: 1, D: 2



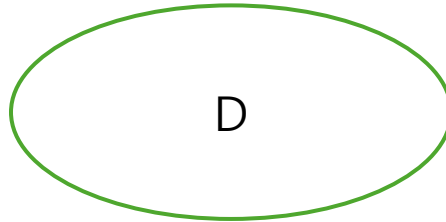
# Example

- B: 0, C: 0, D: 2



# Example

- D: 0





# Example

- Result: [A, B, C, D]



# Screenshot of evaluation result

## 줄 세우기 스페셜 저지

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	52160	30882	20523	57.156%

### 문제

N명의 학생들을 키 순서대로 줄을 세우려고 한다. 각 학생의 키를 직접 재서 정렬하면 간단하겠지만, 마땅한 방법이 없어서 두 학생의 키를 비교하는 방법을 사용하기로 하였다. 그나마도 모든 학생들을 다 비교해 본 것이 아니고, 일부 학생들의 키만을 비교해 보았다.

일부 학생들의 키를 비교한 결과가 주어졌을 때, 줄을 세우는 프로그램을 작성하시오.

### 입력

첫째 줄에  $N(1 \leq N \leq 32,000)$ ,  $M(1 \leq M \leq 100,000)$ 이 주어진다. M은 키를 비교한 회수이다. 다음 M개의 줄에는 키를 비교한 두 학생의 번호 A, B가 주어진다. 이는 학생 A가 학생 B의 앞에 서야 한다는 의미이다.

학생들의 번호는 1번부터 N번이다.

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
79039297	qorwnschangeint	2252	맞았습니다!!	4492 KB	44 ms	C99 / 수정	2005 B	2시간 전

# How to solve

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX_N 32000
5  #define MAX_M 100000
6
7  // Define the Node structure for adjacency list
8  typedef struct Node {
9      int vertex;
10     struct Node* next;
11 } Node;
12
13 // Define the Queue structure to manage vertices for topological sort
14 typedef struct Queue {
15     int* data;
16     int front;
17     int rear;
18     int size;
19 } Queue;
20
21 // Function to add an element to the queue
22 void enqueue(Queue* q, int vertex) {
23     q->data[q->rear++] = vertex;
24 }
25
26 // Function to remove an element from the queue
27 int dequeue(Queue* q) {
28     return q->data[q->front++];
29 }
```

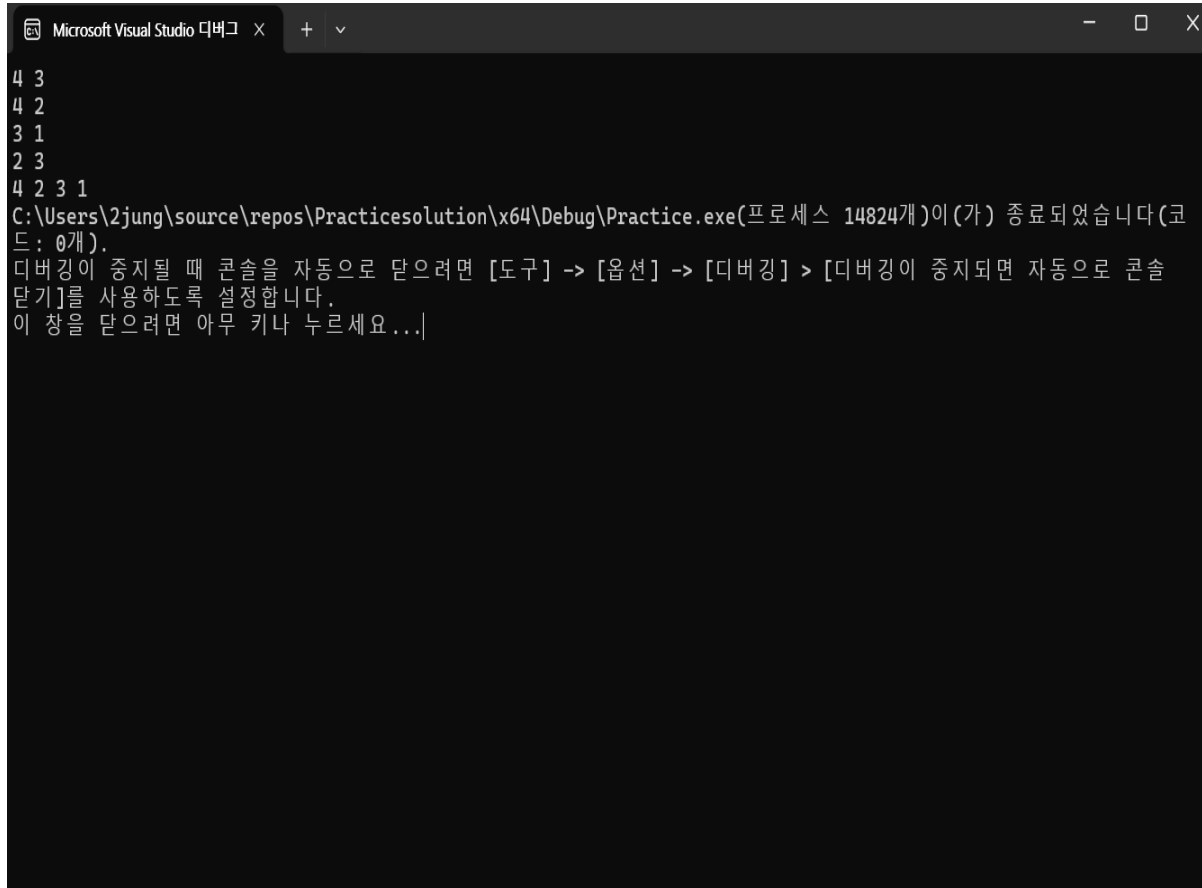
```
31 // Function to check if the queue is empty
32 int is_empty(Queue* q) {
33     return q->front == q->rear;
34 }
35
36 // Function to create a new node for the adjacency list
37 Node* create_node(int vertex) {
38     Node* newNode = (Node*)malloc(sizeof(Node));
39     newNode->vertex = vertex;
40     newNode->next = NULL;
41     return newNode;
42 }
43
44 // Function to add an edge to the adjacency list
45 void add_edge(Node* adjList[], int src, int dest) {
46     Node* newNode = create_node(dest);
47     newNode->next = adjList[src];
48     adjList[src] = newNode;
49 }
50
51 // Function to perform topological sort using Kahn's Algorithm
52 void topological_sort(int n, int in_degree[], Node* adjList[]) {
53     Queue q;
54     q.data = (int*)malloc(n * sizeof(int));
55     q.front = 0;
56     q.rear = 0;
57     q.size = n;
58 }
```

# How to solve

```
59 // Add vertices with 0 in-degree to the queue
60 for (int i = 1; i <= n; i++) {
61     if (in_degree[i] == 0) {
62         enqueue(&q, i);
63     }
64 }
65
66 // Process the vertices in the queue
67 while (!is_empty(&q)) {
68     int current = dequeue(&q);
69     printf("%d ", current); // Print the result
70
71     Node* temp = adjList[current];
72     while (temp != NULL) {
73         in_degree[temp->vertex]--;
74         if (in_degree[temp->vertex] == 0) {
75             enqueue(&q, temp->vertex);
76         }
77         temp = temp->next;
78     }
79 }
80 free(q.data);
81 }
82 }
```

```
84 int main() {
85     int n, m;
86     scanf("%d %d", &n, &m);
87
88     // Initialize in-degree array and adjacency list
89     int* in_degree = (int*)calloc(n + 1, sizeof(int));
90     Node* adjList[MAX_N + 1] = { NULL };
91
92     // Read edges and construct the graph
93     for (int i = 0; i < m; i++) {
94         int a, b;
95         scanf("%d %d", &a, &b);
96         add_edge(adjList, a, b);
97         in_degree[b]++;
98     }
99     // Perform topological sort
100     topological_sort(n, in_degree, adjList);
101
102     // Free dynamically allocated memory
103     for (int i = 1; i <= n; i++) {
104         Node* temp = adjList[i];
105         while (temp != NULL) {
106             Node* toFree = temp;
107             temp = temp->next;
108             free(toFree);
109         }
110     }
111     free(in_degree);
112     return 0;
113 }
```

# Result



```
Microsoft Visual Studio 디버그 콘솔
4 3
4 2
3 1
2 3
4 2 3 1
C:\Users\2jung\source\repos\Practicesolution\x64\Debug\Practice.exe(프로세스 14824개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```