**HIDATO PUZZLE  
중간보고서**

15조

20142765 조영선

20142772 최승호

20142721 심재욱

20142731 우승민

20162824 박지선

20142773 최인정

**목차**

[**1.** **변동사항** 3](#_Toc530614249)

[**2.** **역할 분담 및 진행상황** 3](#_Toc530614250)

[2.1. 역할분담 3](#_Toc530614251)

[2.2. 프로젝트 진행상황 3](#_Toc530614252)

[**3.** **프로젝트 구성** 4](#_Toc530614253)

[**4.** **클래스 구조 및 세부설명** 4](#_Toc530614254)

[4.1. PuzzleManager 4](#_Toc530614255)

[4.2. HidatoSolver 5](#_Toc530614256)

[1) Preprocessing 5](#_Toc530614257)

[2) findStart 6](#_Toc530614258)

[3) Search 6](#_Toc530614259)

[4) getNeighbors 8](#_Toc530614260)

[**5.** **실행 결과** 9](#_Toc530614261)

[**6.** **개선사항** 10](#_Toc530614262)

[6.1. 개선 대상 10](#_Toc530614263)

[6.2. test 방법 10](#_Toc530614264)

[6.3. 실행 결과 11](#_Toc530614265)

[6.3.1. 초기 실행결과 11](#_Toc530614266)

[6.3.2. findStart() 개선 이후 11](#_Toc530614267)

[6.3.3. findStart() & getNeighbors() 개선 12](#_Toc530614268)

[**7.** **프로젝트 계획** 12](#_Toc530614269)

[**8.** **github 주소 및 참고자료** 13](#_Toc530614270)

# **변동사항**

* Solver와 generator 개발 통합
* Solver와 generator 개발을 동시에 진행하려던 기존의 계획을 수정하여 solver완료 후 generator 순차적으로 개발 예정
* 중간 점검 목표 변경
* 개발 팀 통합에 따라 중간 점검 목표를 generator & solver prototype 제작에서 solver 개발 완료로 변경

# **역할 분담 및 진행상황**

## 역할분담

* 조영선: 팀장, solver 모듈화, solver 설계 및 구현, PM
* 최승호: 외부 코드 분석, solver 구현, 주석 관리
* 우승민: solver 세부 구현, 입출력 구축, 코드 형식 통합
* 심재욱: solver 세부 구현, 알고리즘 개선, test
* 최인정: 코드 분석, 보고서 작성, 발표
* 박지선: 회의록 작성, 피드백 정리, 참고자료 수집

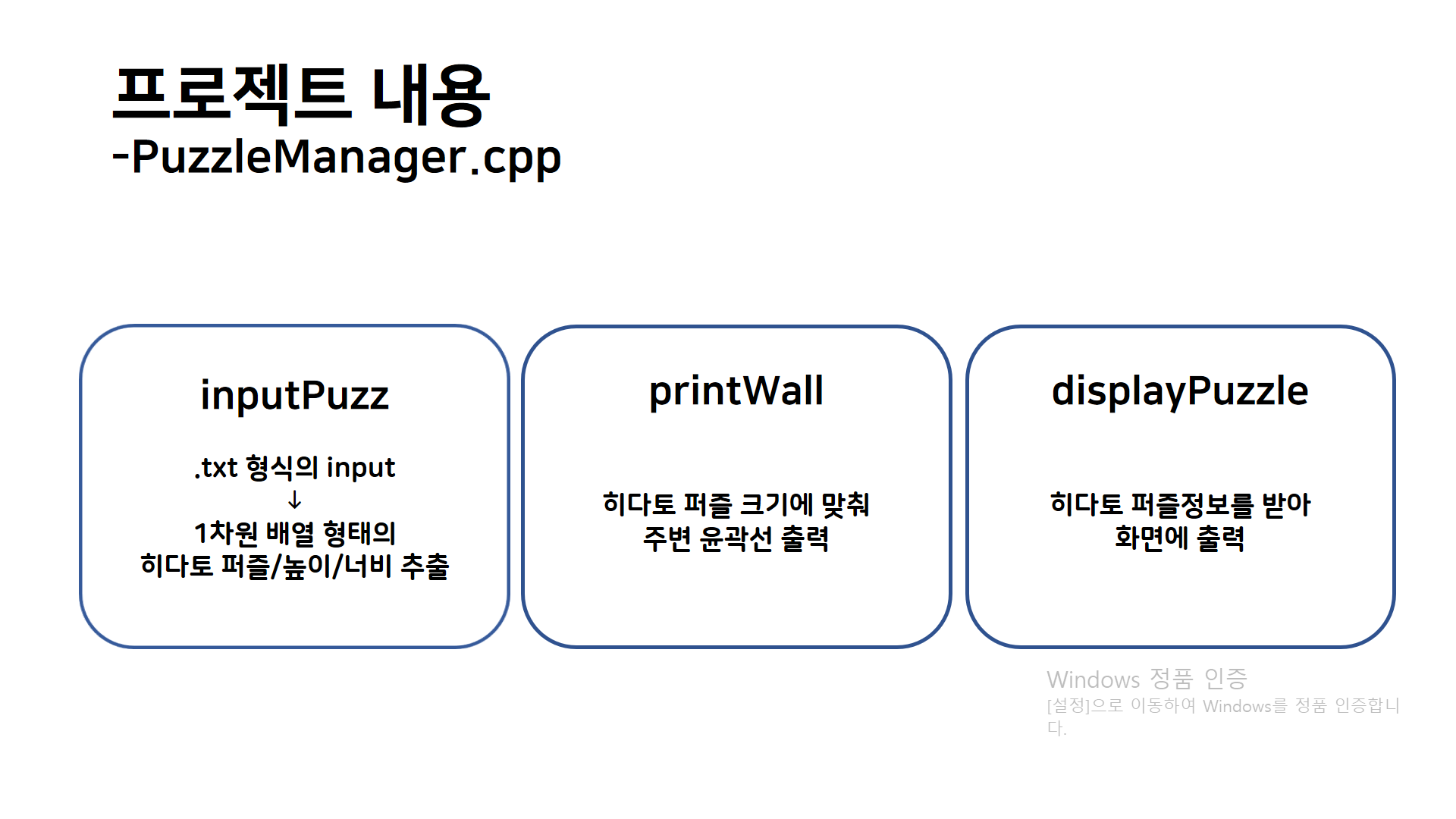
## 프로젝트 진행상황

|  |  |
| --- | --- |
| **일정** | **진행사항** |
| 1주차(11.02 ~11.06) | SOLVER TASK 분석 및 구조 설계 |
| 2주차(11.08 ~11.13) | SOLVER 1차 개발 및 디버깅 |
| 3주차(11.14 ~11.20) | SOLVER TEST  SOLVER알고리즘 개선  중간 점검 및 보고서 제작 |

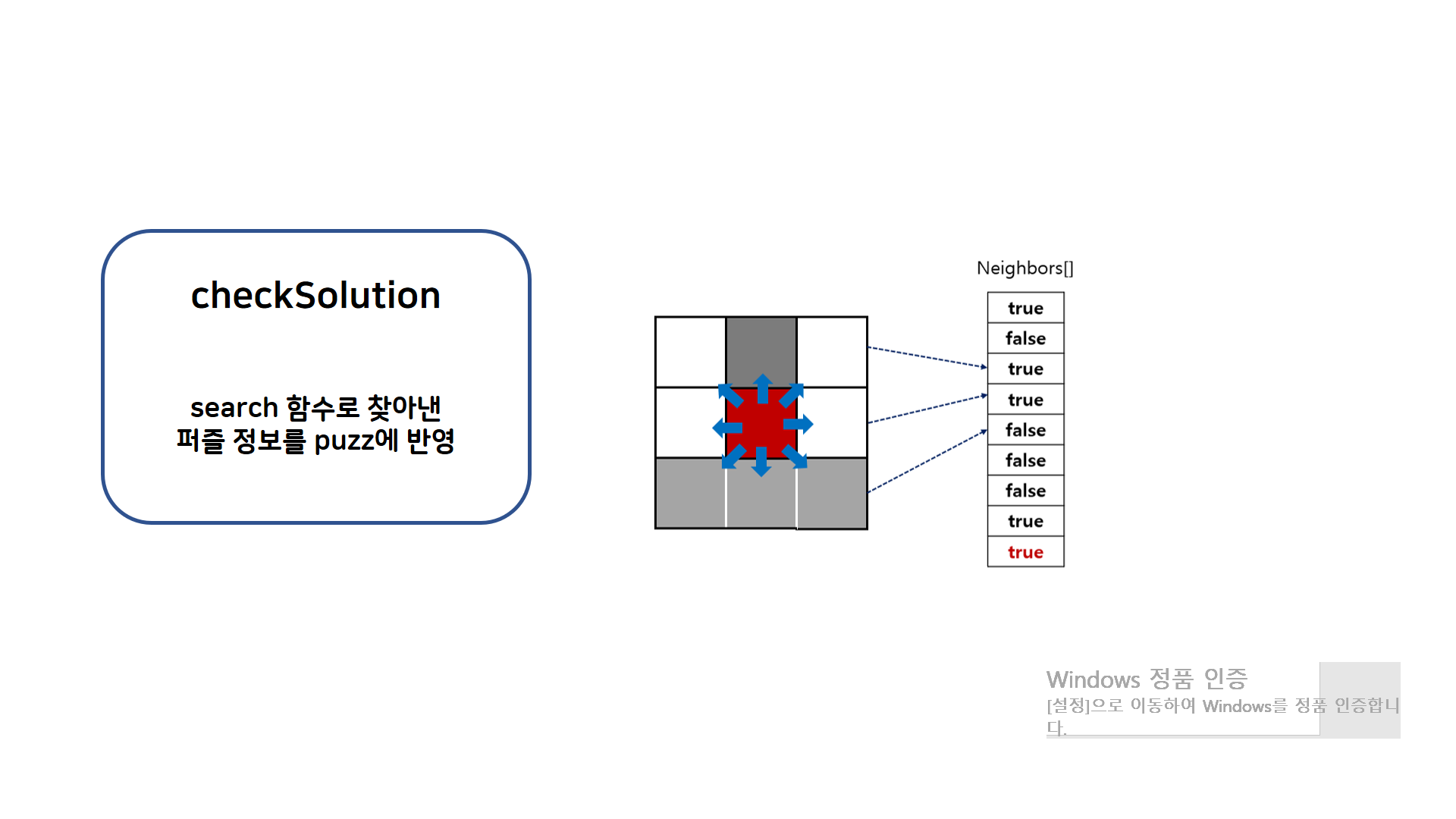
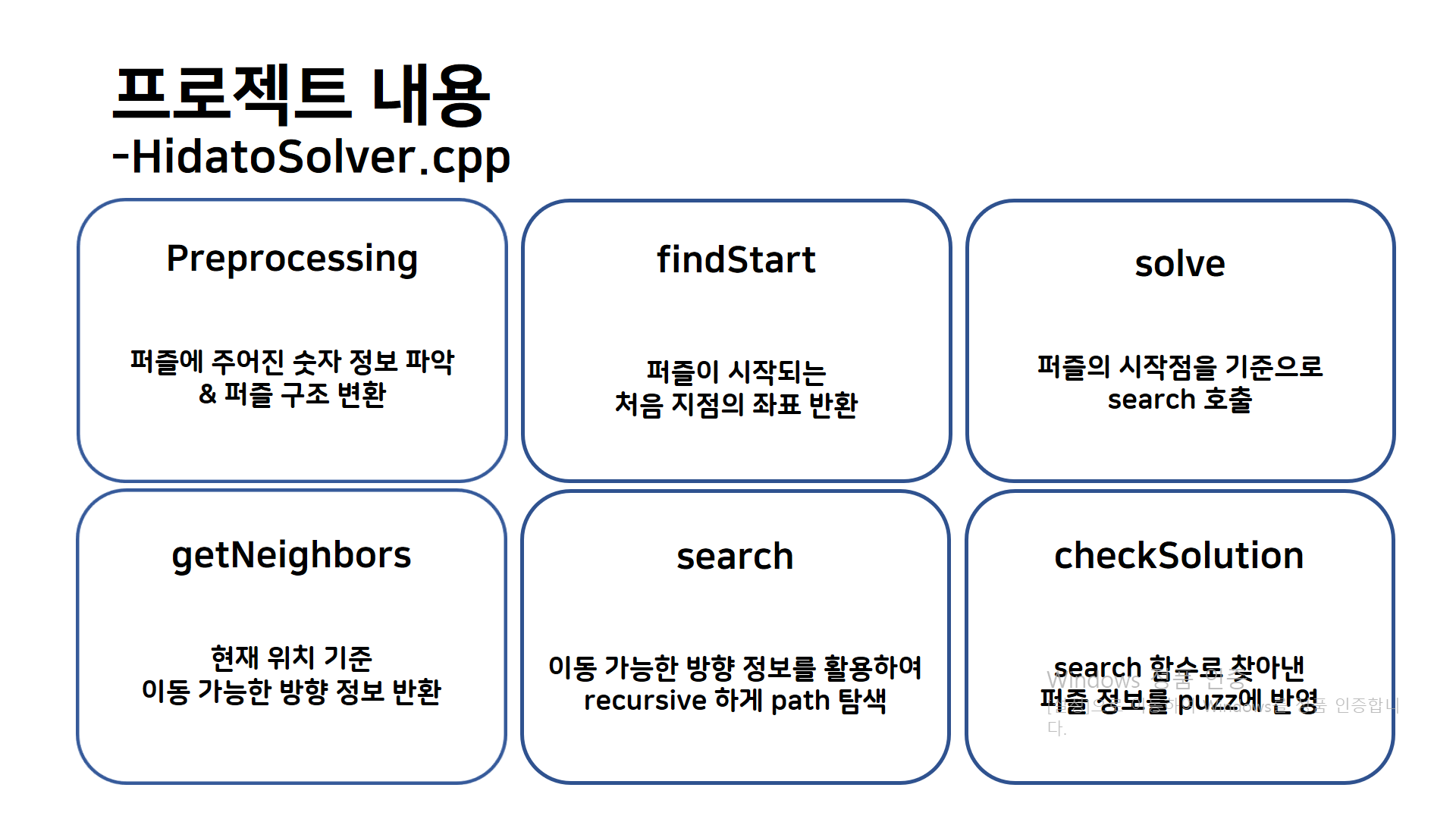
# **프로젝트 구성**

# **클래스 구조 및 세부설명**

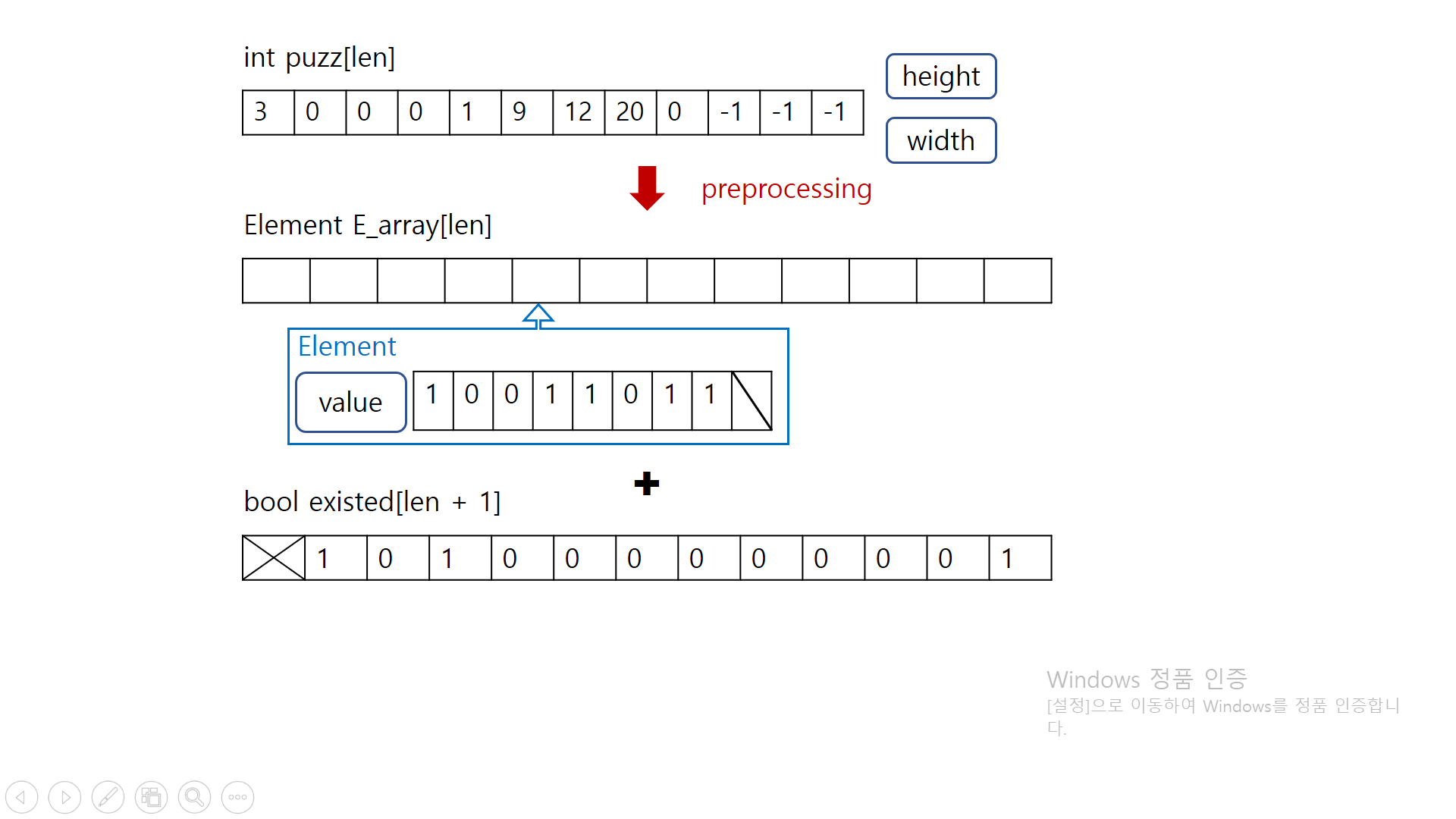
## PuzzleManager



## HidatoSolver

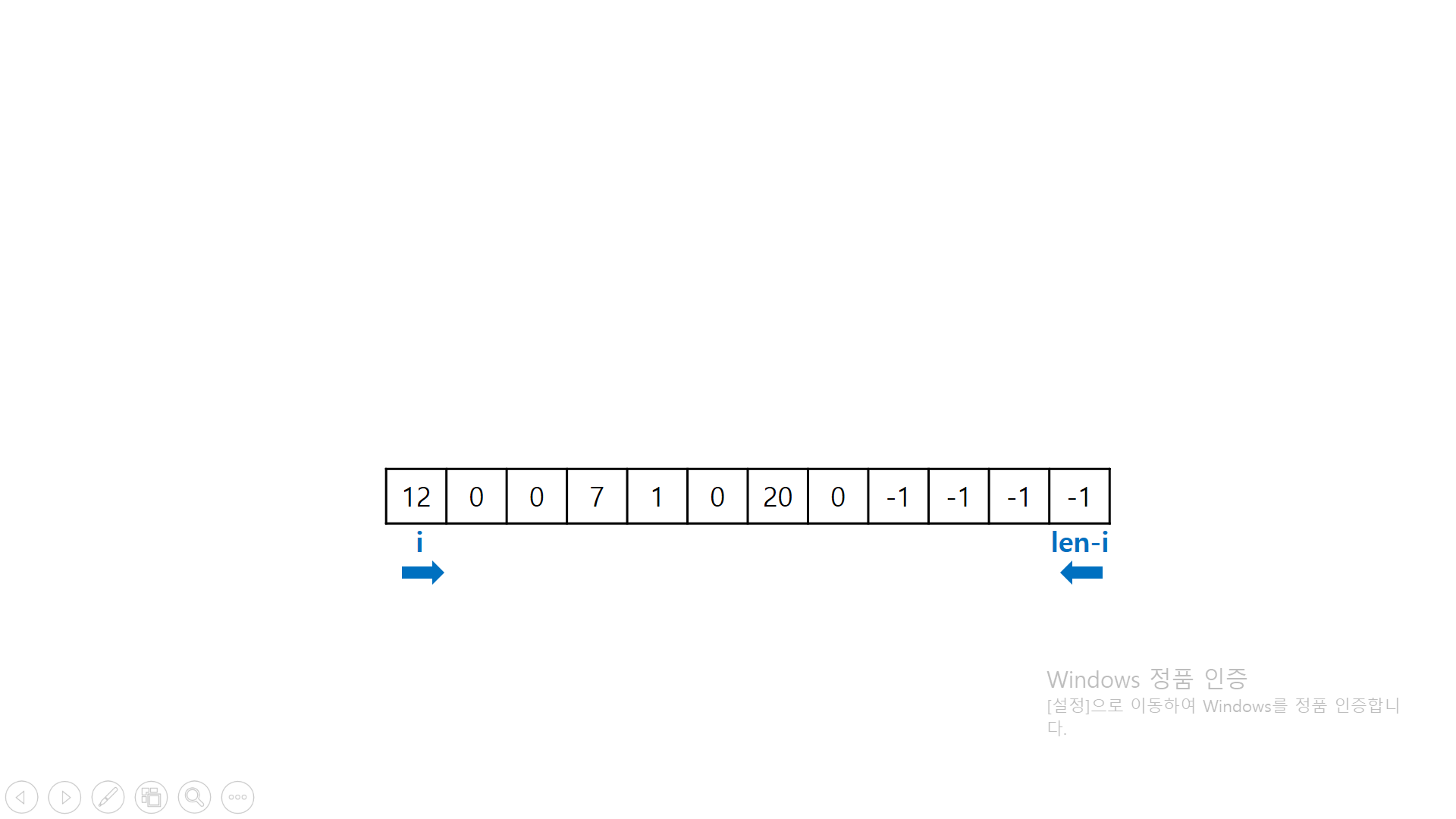


### Preprocessing



* 퍼즐 해결을 위해 퍼즐의 구조를 변환하고 퍼즐의 마지막 숫자와 퍼즐 내부 숫자 정보를 얻기 위한 메소드
* Int type배열 puzz와 높이, 너비 값을 받아 element type배열인 E\_array와 bool type배열인 existed, 마지막 숫자인 max 값 생성
* Puzz를 loop하며 내부 숫자 값을 E\_array의 value값으로 지정하고, existed의 해당 위치 값을 true로 설정하여 숫자가 존재함을 표시
* Puzz의 loop에서 최대값 정보를 max에 넣어 전체 퍼즐의 마지막 숫자 정보를 파악

### findStart

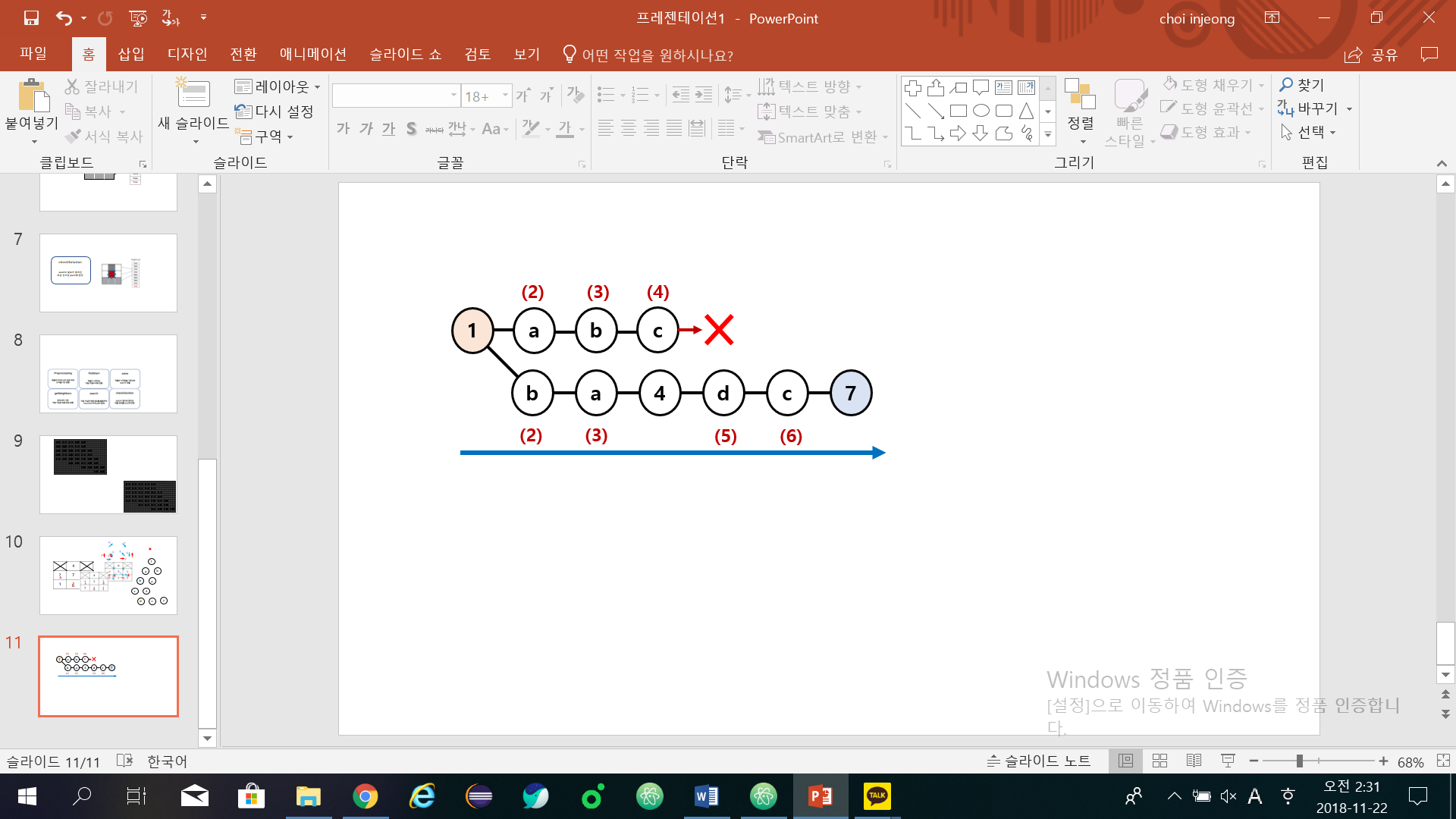


* 퍼즐의 처음 시작지점인 1을 퍼즐 내부에서 찾기 위한 메소드
* 칸의 value가 1인 경우, 해당 칸의 x, y 좌표 할당 후 true 반환

1이 아닌 경우, false 반환

* 퍼즐의 처음부터 끝까지 순차적으로 loop하며 퍼즐 전체를 탐색하는 기존의 형태를 퍼즐의 처음과 끝에서 동시 탐색하는 형태로 수정하여 효율성 증대

### Search



* 현재 위치하는 퍼즐의 좌표와 찾을 숫자 값을 받아 DFS 방식으로 재귀 호출하여 퍼즐의 해답을 만드는 메소드
* getNeighbors()를 호출하여 현재 좌표에서 8개의 방향 중 이동 가능한 방향 정보를 생성한 후 이동 가능한 이웃 칸을 순회하며 진행
* 현재 찾을 숫자가 처음에 주어진 퍼즐에 존재할 때,

-찾던 숫자를 이웃 칸에서 발견한 경우: 찾을 숫자에 +1 하여 다시 search() recursive call

-발견하지 못한 경우: false 반환

* 현재 찾을 숫자가 처음에 주어진 퍼즐에 존재하지 않을 때,

-현재 빈 칸인 이웃의 value를 현재 찾는 order로 가정

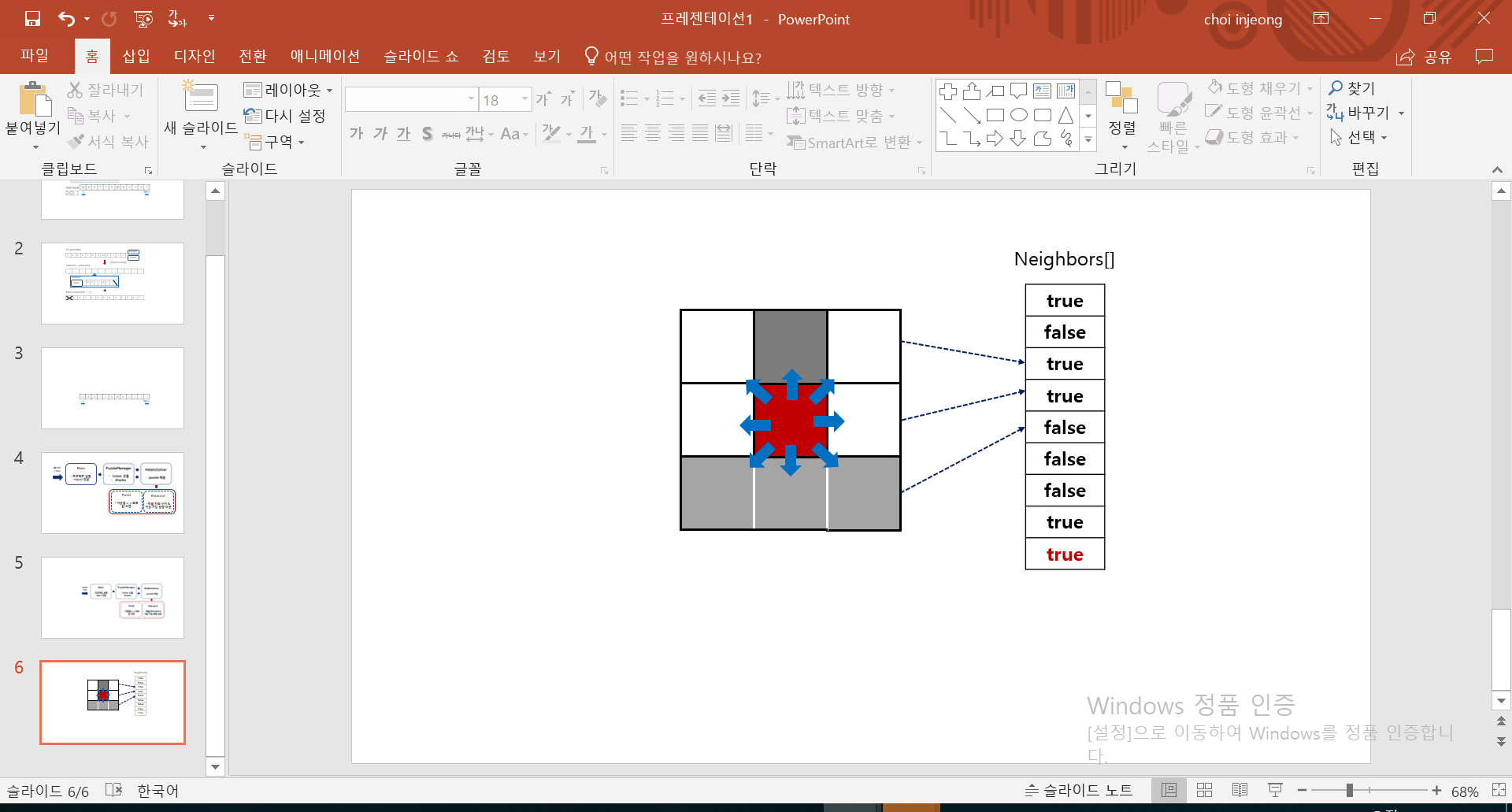
-찾는 숫자를 +1 하여 search() 호출

-호출한 search()가

-true를 반환: 가정이 맞다고 판단하여 계속 진행

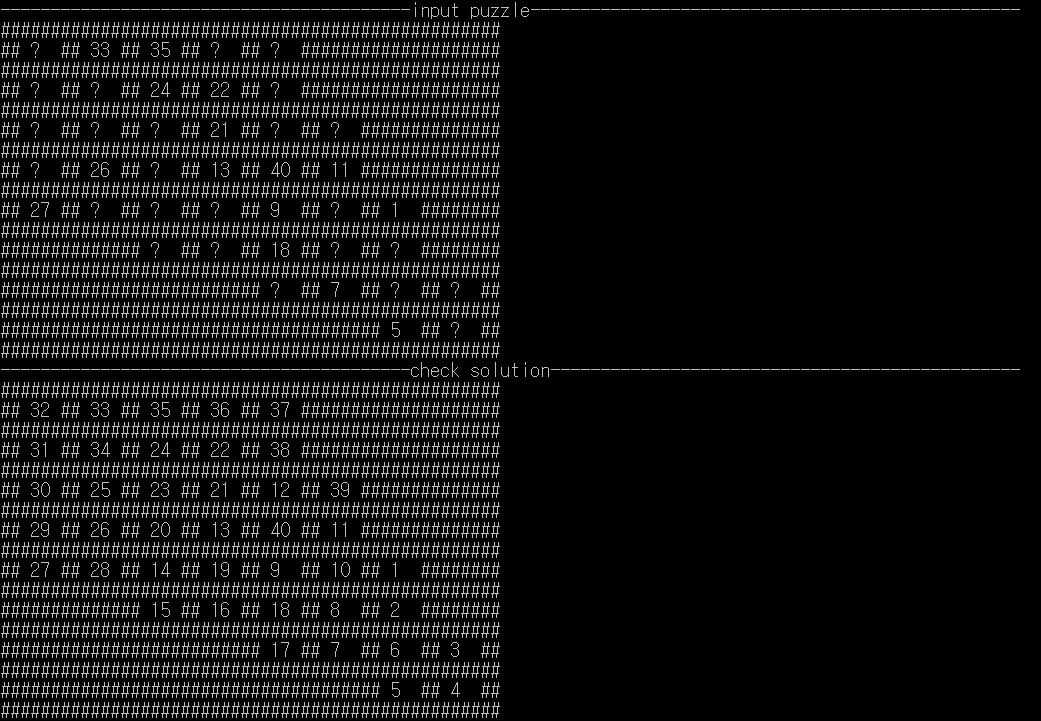
-false를 반환: 가정이 틀리다 판단하여 가정을 recover

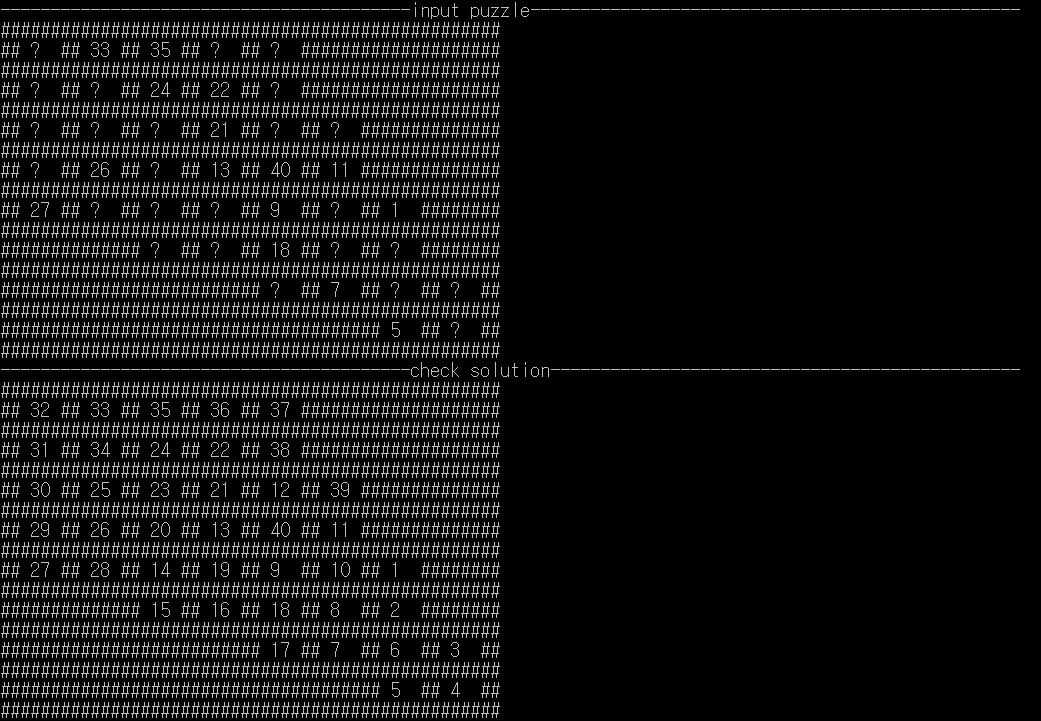
### getNeighbors



* 퍼즐 칸의 좌표를 받아 해당 좌표에서 이동할 수 있는 칸의 방향 정보와 해당 칸이 getNeighbors()를 이미 실행했다는 기록 정보를 만드는 메소드
* 현재 칸과 이웃하는 칸으로 이동했음을 가정 후, 현재 이동한 칸으로의 이동 가능 여부를 체크
* 만약 이동한 칸이 빈칸 or 숫자인 경우, 해당 방향의 이동가능 여부를 의미하는 element의 neighbors 값을 true로 변경
* 현재 퍼즐 좌표에 대한 이웃 중복 확인을 막기위해 현재 칸의 neighbors 배열 마지막 칸에 기록 정보를 남겨, 시간 효율성 증대

# **실행 결과**



**[input puzzle]**

**[solution]**

# **개선사항**

## 개선 대상

1. 시작 지점 찾기(findStart)
   * + 기존: 퍼즐의 처음부터 끝까지 순차적으로 loop하며 퍼즐 전체를 탐색
     + 개선 후: 2개의 if문으로 퍼즐의 처음과 끝을 동시 탐색하여 반복 문 실행 횟수 감소
2. 이동 가능 여부 확인(getNeighbors)
   * + 기존: 퍼즐을 구성하는 칸에 대한 getNeighbors() 실행 여부를 남기지 않아 경우에 따라 한 칸에 대해 다중 확인 발생
     + 개선 후: 방향 정보 외에 메소드 실행 여부를 bool 타입으로 저장하여 메소드 실행 중복 방지

## test 방법

* 퍼즐 출력을 주석 처리하고, solve() 함수를 10만번 실행
* Main 에서 소요 시간을 10회 출력

(측정단위: nanosecend)

## 6.3. 실행 결과

### 6.3.1. 초기 실행결과

### 6.3.2. findStart() 개선 이후

### 6.3.3. findStart() & getNeighbors() 개선

# **프로젝트 계획**

|  |  |
| --- | --- |
| **일정** | **진행계획** |
| 4주차(11.21 ~ 11.27) | GENERATOR TASK 분석 및 구조 설계 |
| 5주차(11.28 ~ 12.03) | GENERATOR 1차 개발 완료 |
| 6주차(12.04 ~ 12.13) | GENERATOR TEST  알고리즘 최적화  프로젝트 마무리 및 최종 보고서 작성 |

# **github 주소 및 참고자료**

* Github 주소: <https://github.com/YoungsunCho/Algorithm_HidatoPuzzle>
* 참고자료: <https://rosettacode.org/wiki/Solve_a_Hidato_puzzle>

<https://github.com/fogleman/Hidato>

<https://github.com/YoungsunCho/Algorithm_HidatoPuzzle/blob/master/reference/hidato-master.zip>

<https://github.com/YoungsunCho/Algorithm_HidatoPuzzle/blob/master/reference/solver.cpp>