**REPORT**

**| Embedded System**

**| Event flag , Mutex , Message queue**

**| 컴퓨터정보공학과**

**| 12141540 박영창**

# 내용

6주차 실습 강의 시간에는 mutex 에 관해 배웠습니다.

* Mutex
  + Mutex는 semaphore와 같은 동작원리이지만 공유 자원의 개수가 1개로 한정되어 있고, priority inversion을 막을 수 있는 기능 ( 현재 mutex를 소유한 task보다 우선순위가 높은 task가 mutex에 접근할 때 mutex를 소유한 task의 우선순위를 임시적으로 mutex 생성 시 설정된 우선순위로 높임으로써 구현 ) 이 내장되어 있다.
  + 선언
    - OS\_EVENT \* mutex

<설명>

OS\_EVENT \* 형으로 선언

<인자>

mutex : 사용할 mutex 이름

* + 초기화
    - OS\_EVENT \* OSMutexCreate(INT8U prio, INT8U \*err)

<설명>

Mutex를 만든다

<인자>

Prio : 사용할 최상위 우선순위

Err : 에러 코드

* + Wait
    - Void OSMutexPend(OS\_EVENT \* ptr, INT16U timeout, INT8U \*err)

<설명>

Mutex ptr로부터 신호를 받는다

<인자>

ptr : OSMutexCreate()에 의해 만들어진 Mutex 정보

timeout : 대기시간

err : 에러 코드

* + Signal
    - INT8U OSMutexPost(OS\_EVENT \* ptr)

<설명>

Mutex ptr에 신호를 전달한다

<인자>

ptr : OSMutexCreate()에 의해 만들어진 Mutex 정보

# 과제 목표

* 4개의 task 생성 ( 1 개의 decision task , 3 개의 voting task )
* 3개의 voting task에서 O 또는 X 둘 중 하나의 문자를 생성한다.
* Decision task에서 3개의 voting task 중 master task를 임의로 선택한다.

위 두 과정을 event flag를 통해 동기화한다.

* 3 개의 voting task 중 master task가 아닌 task는 생성된 문자를 message queue를 사용해서 master task에 넘겨준다.
* Voting task 중 master task는 O 와 X 의 개수를 계산하고, 공유 변수에 mutex를 사용해 저장한다.
* Decision task는 event flag를 활용하여 3 개의 voting task 중 master task에서 flag를 활용하여 굥유 변수를 확인한다.
* O가 더 많다면 파란색 출력, X가 더 많다면 빨간색 출력
* 위 과정을 4초 간격으로 반복한다.

# 문제 해결

**변수**

* **전역변수**
  + INT8U selectMasterTask

Master Task의 번호를 저장

* + INT8U o\_cnt, x\_cnt

O와 X의 개수를 저장

* **Mutex 선언**
  + OS\_EVENT \*mutex

전역변수 보호를 위한 mutex

* **Event Flag 선언**
  + OS\_EVENT \*s\_grp

Master Task에서 O , X의 count를 체크했는지 여부를 나타내는 flag

* + OS\_EVENT \*m\_grp

Voting Task에서 O , X를 생성했는지 여부를 나타내는 flag

* + OS\_EVENT \*n\_grp

Decision Task에서 Master Task를 결정했는지 여부를 나타내는 flag

* **Message Queue 선언**
  + OS\_EVENT \*MessageQueue

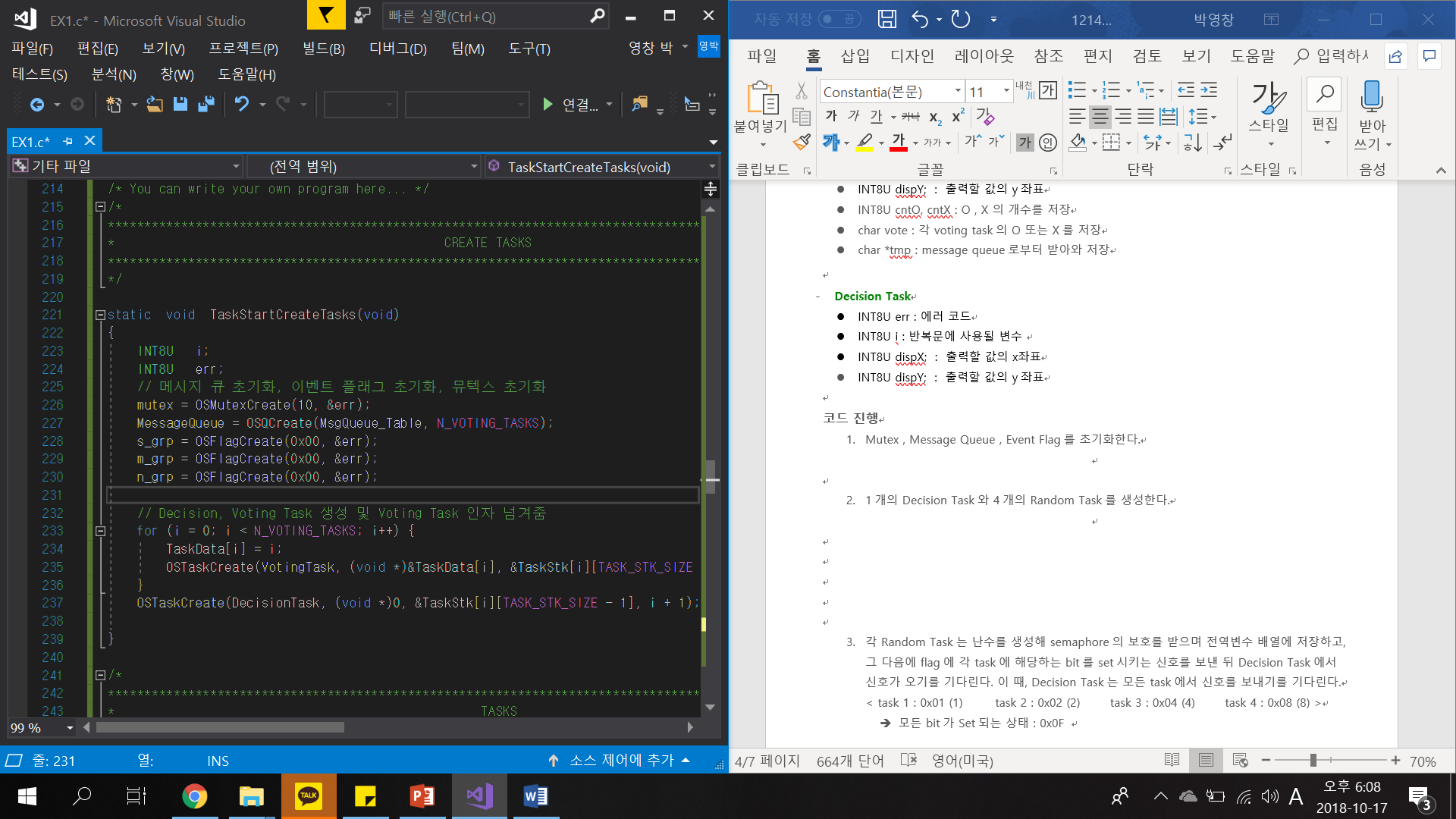
Void \* MsgQueue\_Table[N\_VOTING\_TASKS]

Master Task에게 O , X를 전달할 message queue

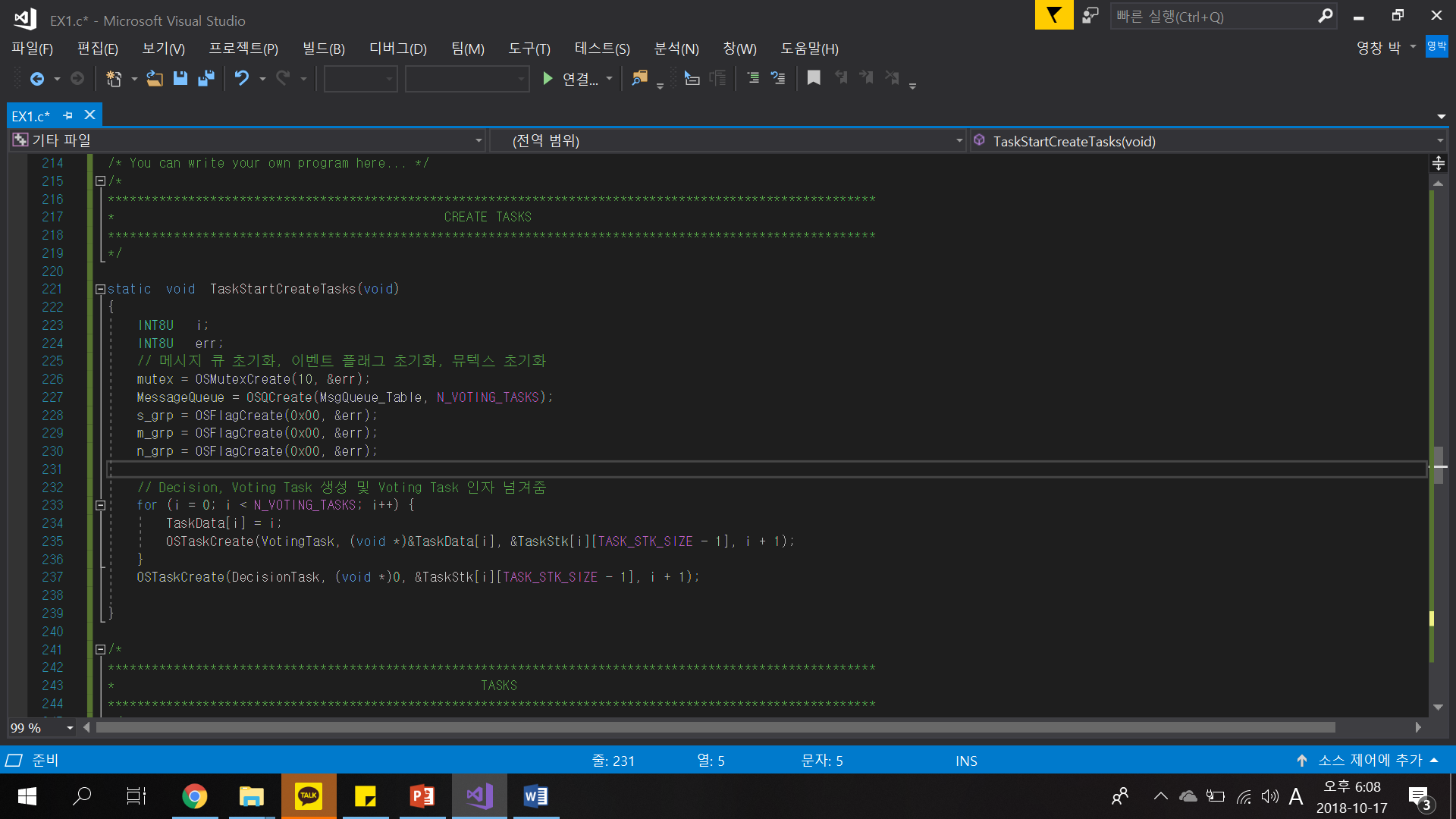
* **상수 define** 
  + #define N\_VOTING\_TASKS 3 : Voting Task의 개수
  + #define N\_DECISION\_TASKS 1 : Decision task의 개수
* **Random Task**
  + INT8U i : 반복문을 위해 사용하는 변수
  + INT8U err; : 에러 코드
  + INT8U number; : 난수 생성
  + INT8U number\_of\_votingtask : task의 우선순위 번호
  + INT8U flagP : task 우선순위별로 OSFlagPost, OSFlagPend의 인자로 넘겨줄 bit
  + INT8U dispX; : 출력할 값의 x좌표
  + INT8U dispY; : 출력할 값의 y좌표
  + INT8U cntO, cntX : O , X 의 개수를 저장
  + char vote : 각 voting task의 O 또는 X를 저장
  + char \*tmp : message queue로부터 받아와 저장
* **Decision Task**
  + INT8U err : 에러 코드
  + INT8U i : 반복문에 사용될 변수
  + INT8U dispX; : 출력할 값의 x좌표
  + INT8U dispY; : 출력할 값의 y좌표

**코드 진행**

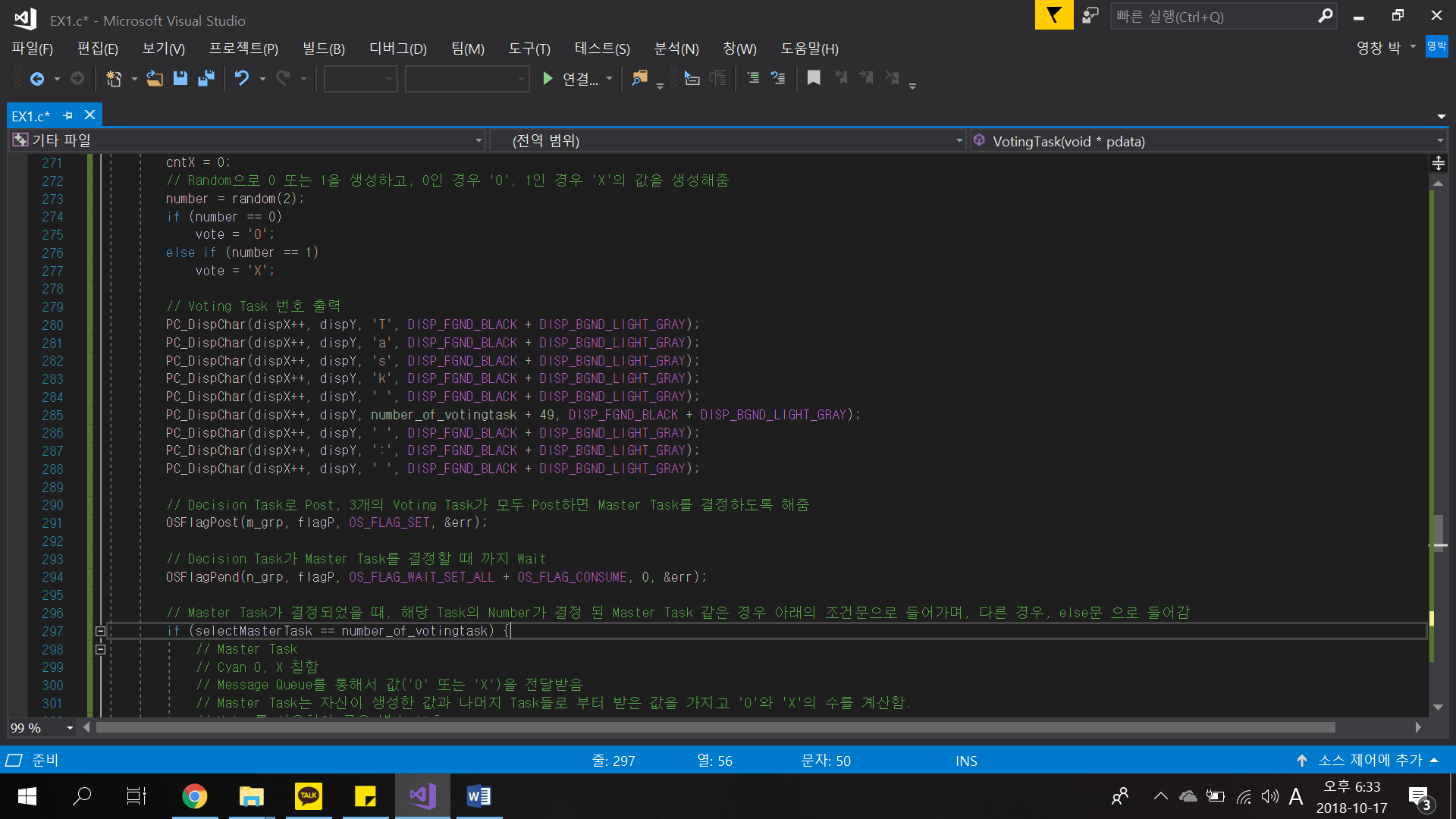
1. Mutex , Message Queue , Event Flag를 초기화한다.

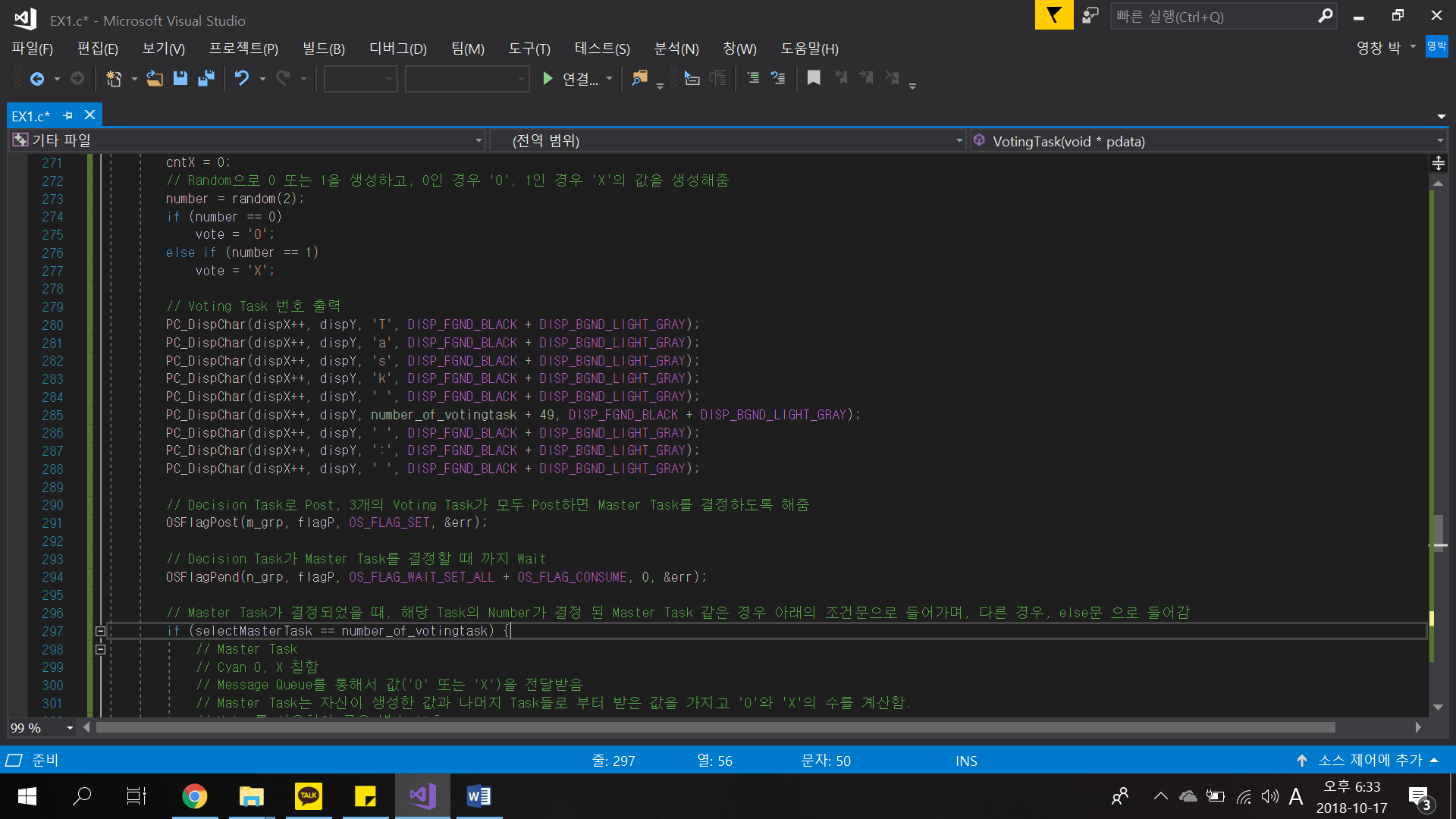


1. 1개의 Decision Task와 4개의 Random Task를 생성한다.

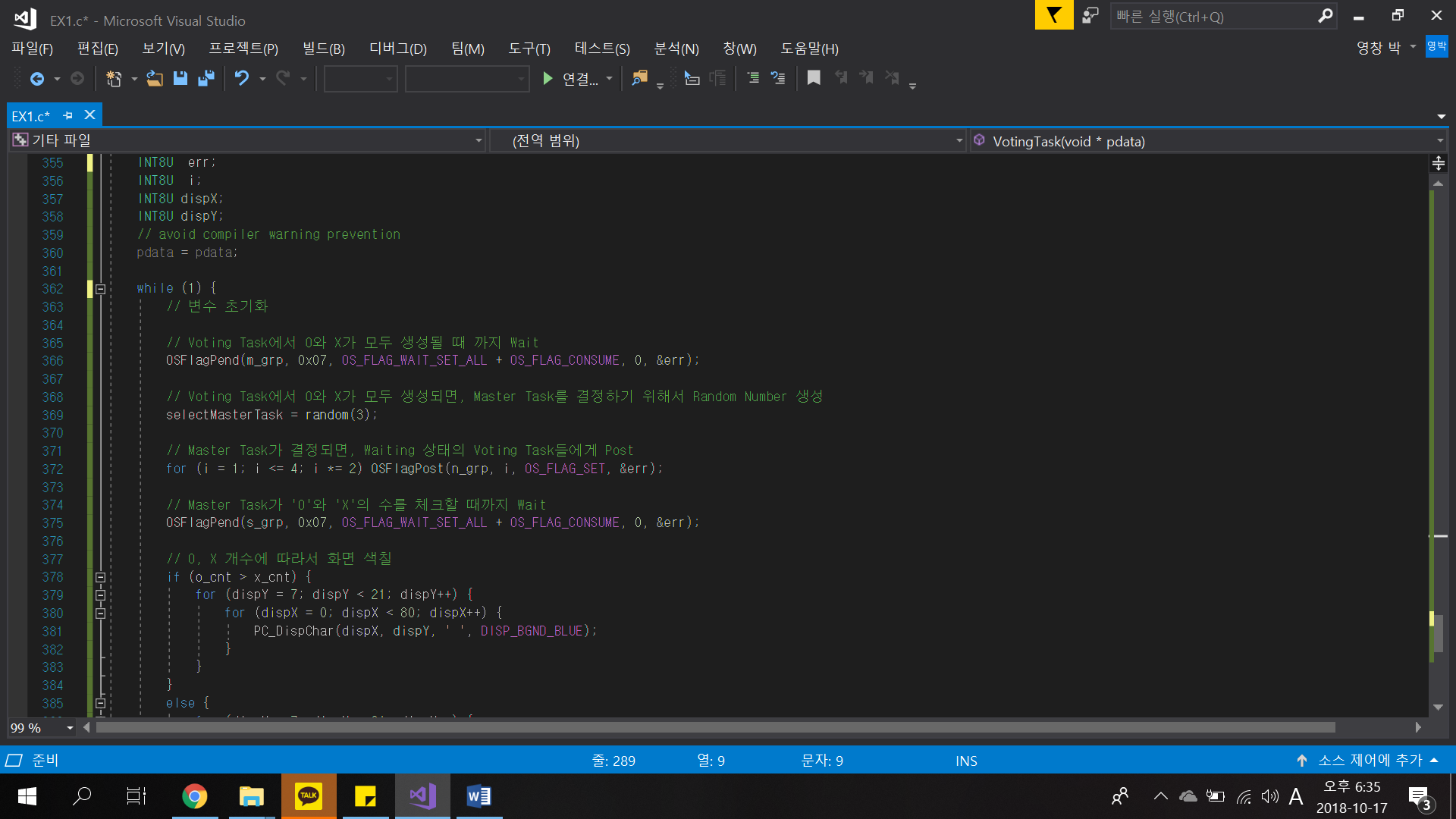


1. 각 voting task에서 O 인지 X인지를 결정한 다음 decision task에게 결정되었다는 신호를 보낸다 뒤 master task가 정해졌다는 신호가 오기를 기다린다.

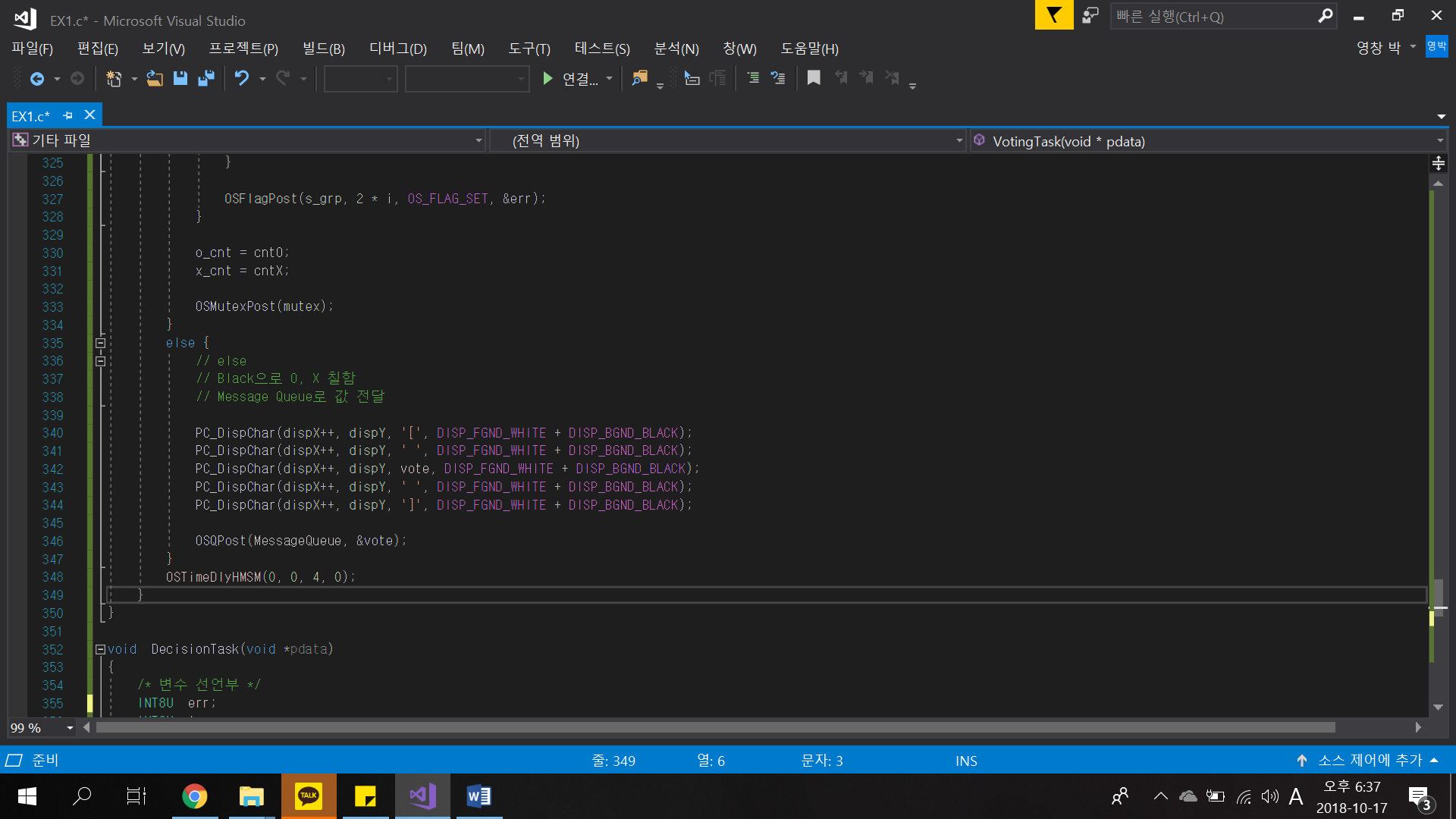




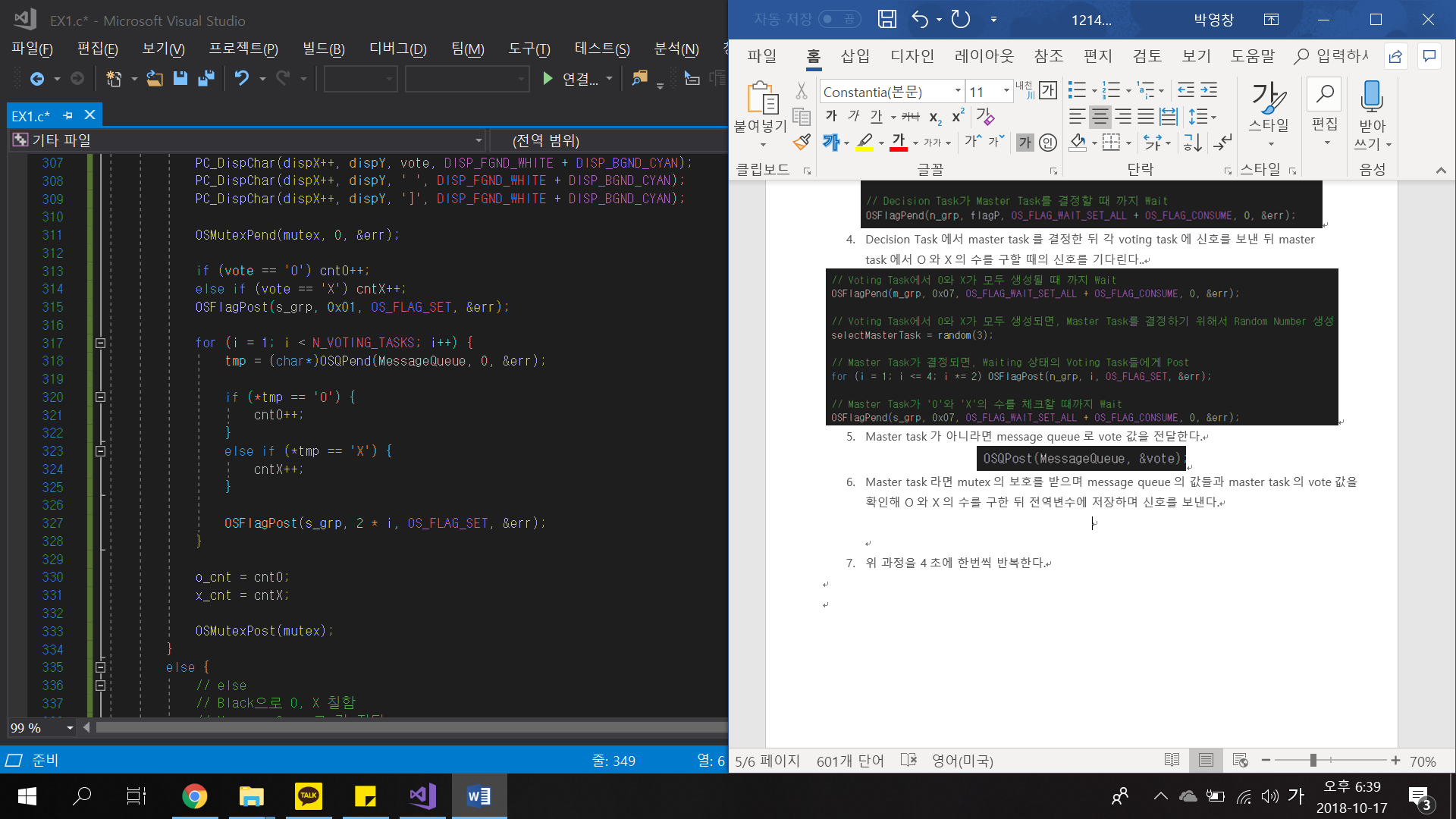
1. Decision Task에서 master task를 결정한 뒤 각 voting task에 신호를 보낸 뒤 master task에서 O와 X의 수를 구할 때의 신호를 기다린다..



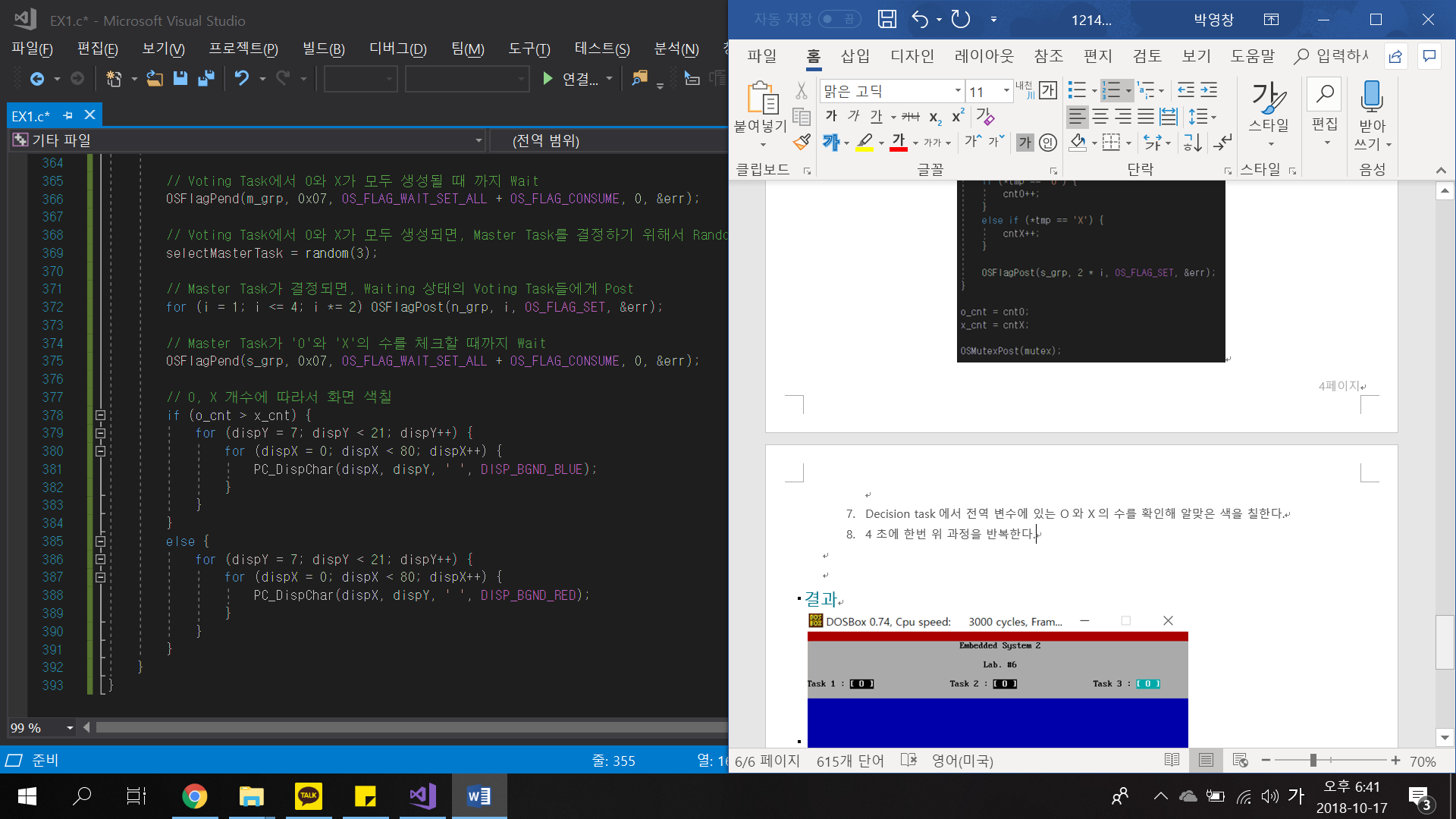
1. Master task가 아니라면 message queue로 vote 값을 전달한다.



1. Master task라면 mutex의 보호를 받으며 message queue의 값들과 master task의 vote값을 확인해 O와 X의 수를 구한 뒤 전역변수에 저장하며 신호를 보낸다.



1. Decision task에서 전역 변수에 있는 O와 X의 수를 확인해 알맞은 색을 칠한다.



1. 4 초에 한번 위 과정을 반복한다.

# 결과

# 

# 