

## C# 2번째 개인 과제

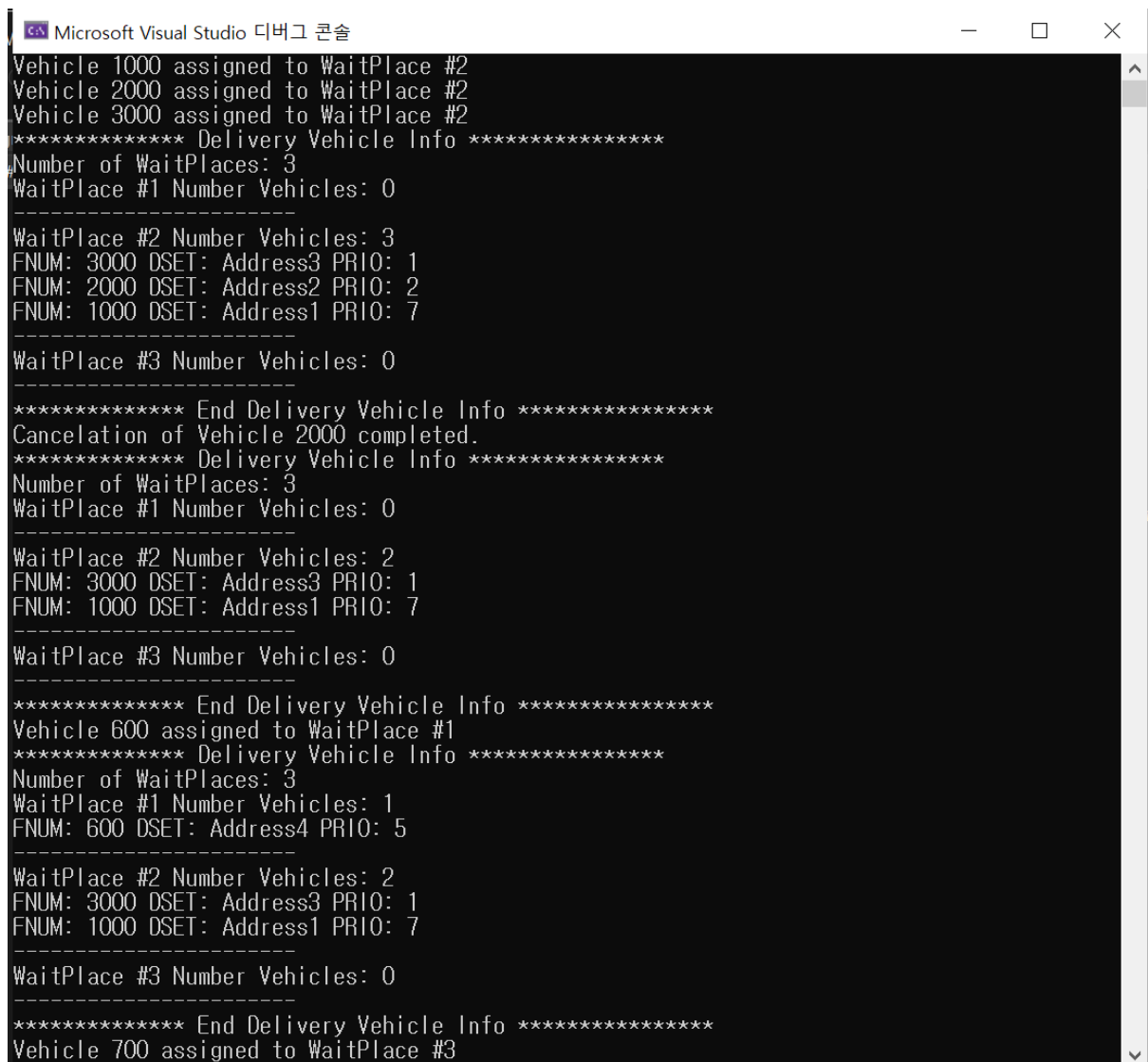
학과, 학번 : 컴퓨터공학과 21011575

이름 : 박준형

### 1. 명령어 입력과 결과 화면

```
//StreamReader로 input.txt에서 입력받는다.  
StreamReader sr = new StreamReader("C:\\Users\\박준형\\source\\repos\\c#_project2\\c#_project2\\INPUT.txt");
```

StreamReader를 사용하여 컴퓨터에 있는 INPUT.txt에 있는 문장들을 한줄씩 입력받아 명령어를 처리하는 과정으로 코딩을 하였고, 해당 명령어들을 통한 결과가 콘솔창을 통하여 보이도록 하였다.



```
Microsoft Visual Studio 디버그 콘솔  
Vehicle 1000 assigned to WaitPlace #2  
Vehicle 2000 assigned to WaitPlace #2  
Vehicle 3000 assigned to WaitPlace #2  
***** Delivery Vehicle Info *****  
Number of WaitPlaces: 3  
WaitPlace #1 Number Vehicles: 0  
-----  
WaitPlace #2 Number Vehicles: 3  
FNUM: 3000 DSET: Address3 PRIO: 1  
FNUM: 2000 DSET: Address2 PRIO: 2  
FNUM: 1000 DSET: Address1 PRIO: 7  
-----  
WaitPlace #3 Number Vehicles: 0  
-----  
***** End Delivery Vehicle Info *****  
Cancelation of Vehicle 2000 completed.  
***** Delivery Vehicle Info *****  
Number of WaitPlaces: 3  
WaitPlace #1 Number Vehicles: 0  
-----  
WaitPlace #2 Number Vehicles: 2  
FNUM: 3000 DSET: Address3 PRIO: 1  
FNUM: 1000 DSET: Address1 PRIO: 7  
-----  
WaitPlace #3 Number Vehicles: 0  
-----  
***** End Delivery Vehicle Info *****  
Vehicle 600 assigned to WaitPlace #1  
***** Delivery Vehicle Info *****  
Number of WaitPlaces: 3  
WaitPlace #1 Number Vehicles: 1  
FNUM: 600 DSET: Address4 PRIO: 5  
-----  
WaitPlace #2 Number Vehicles: 2  
FNUM: 3000 DSET: Address3 PRIO: 1  
FNUM: 1000 DSET: Address1 PRIO: 7  
-----  
WaitPlace #3 Number Vehicles: 0  
-----  
***** End Delivery Vehicle Info *****  
Vehicle 700 assigned to WaitPlace #3
```

```
Microsoft Visual Studio 디버그 콘솔
Vehicle 700 assigned to WaitPlace #3
***** Delivery Vehicle Info *****
Number of WaitPlaces: 3
WaitPlace #1 Number Vehicles: 1
FNUM: 600 DSET: Address4 PRI0: 5
-----
WaitPlace #2 Number Vehicles: 2
FNUM: 3000 DSET: Address3 PRI0: 1
FNUM: 1000 DSET: Address1 PRI0: 7
-----
WaitPlace #3 Number Vehicles: 1
FNUM: 700 DSET: Address5 PRI0: 4
-----
***** End Delivery Vehicle Info *****
Vehicle 800 assigned to WaitPlace #1
***** Delivery Vehicle Info *****
Number of WaitPlaces: 3
WaitPlace #1 Number Vehicles: 2
FNUM: 800 DSET: Address6 PRI0: 1
FNUM: 600 DSET: Address4 PRI0: 5
-----
WaitPlace #2 Number Vehicles: 2
FNUM: 3000 DSET: Address3 PRI0: 1
FNUM: 1000 DSET: Address1 PRI0: 7
-----
WaitPlace #3 Number Vehicles: 1
FNUM: 700 DSET: Address5 PRI0: 4
-----
***** End Delivery Vehicle Info *****
WaitPlace #2 cleared.
***** Delivery Vehicle Info *****
Number of WaitPlaces: 3
WaitPlace #1 Number Vehicles: 2
FNUM: 800 DSET: Address6 PRI0: 1
FNUM: 600 DSET: Address4 PRI0: 5
-----
WaitPlace #2 Number Vehicles: 0
-----
WaitPlace #3 Number Vehicles: 1
FNUM: 700 DSET: Address5 PRI0: 4
-----
***** End Delivery Vehicle Info *****
Vehicle 800 used to deliver.
***** Delivery Vehicle Info *****
Number of WaitPlaces: 3
WaitPlace #1 Number Vehicles: 1
FNUM: 600 DSET: Address4 PRI0: 5
-----
WaitPlace #2 Number Vehicles: 0
-----
WaitPlace #3 Number Vehicles: 1
FNUM: 700 DSET: Address5 PRI0: 4
-----
***** End Delivery Vehicle Info *****
C:\Users\박준형\source\repos\c#\project2\c#\project2\bin\Debug\net6.0\c#\project2.exe(프로세스 18592개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요....
```

Output.txt 방식으로 하려 했으나 계속 Output에 기록되지 않아 어쩔 수 없이 콘솔에 나오게 하는 방식으로 코드를 작성하였습니다.

2. 코드 작성에 사용한 개념 (전체 코드는 압축파일에서 확인할 수 있습니다.)

-> 이곳에서는 어떠한 개념을 이용해서 코드를 작성했는지 설명

```
//자동차에 관한 정보
참조 9개
public class DeliveryVehicle
{
    //자동차 아이디
    public int vehicleId;
    //배달 목적지 문자열
    public string destination;
    //우선순위
    public int priority;
}

//대기 장소 클래스 정보
참조 2개
public class DeliveryVehicleManager
{
    //대기장소 총 개수
    public int numWaitingPlaces;
    //대기장소 안에 차가 있는 배열
    public DeliveryVehicle [][] waitPlaces;
}
```

자동차에 대한 정보를 입력하는 DeliveryVehicle 클래스와 차가 들어가는 대기 장소를 나타내는 DeliveryVehicleManager 클래스를 위와 같이 선언해 두었다.

```
DeliveryVehicleManager deliveryVehicleManager = new DeliveryVehicleManager();

// 입력 파일에 더 이상 읽을 문자가 없을 때 까지 실행
while (sr.Peek() >= 0)
{
    line++;

    int id;
    int prio;
    int length;

    // 입력파일에 한 줄의 문자열을 읽어와 string배열에 입력하여 split한다.
    string[] tmpreadline = sr.ReadLine().Split();
```

해당 대기 장소에 대한 변수를 선언하고, INPUT.txt에서 입력받은 명령을 한줄한줄 확인한다.

```
//첫째 줄은 대기장소의 수를 입력받는다.
if (line == 1)
{
    //대기장소를 뜻하는 deliveryVehicleManager 클래스의 정보를 입력한다.
    deliveryVehicleManager.numWaitingPlaces = int.Parse(tmpreadline[0]);
    total_space = int.Parse(tmpreadline[0]);
    //그리고 입력받은 대기장소에 따른 동적배열을 할당해준다.
    deliveryVehicleManager.waitPlaces = new DeliveryVehicle[deliveryVehicleManager.numWaitingPlaces][];
    //모든 대기장소에 대하여 공간을 0씩으로 동적할당을 해준다.
    for (int i = 0; i < deliveryVehicleManager.numWaitingPlaces; i++)
    {
        deliveryVehicleManager.waitPlaces[i] = new DeliveryVehicle[0];
    }
}
```

배열의 크기를 최소로 하는 방식으로 하라고 하셨기 때문에 대기장소를 입력받으면 대기장소의 크기에 맞추어 동적할당으로 공간을 할당해주고, 아직 차 정보를 입력받기 전이므로 각각의 대기장소의 공간을 크기 0으로 설정하여 주었습니다.

```
//해당 대기장소에 차 한대를 추가해줘야 하기 때문에 배열의 크기를 1 늘려준다.
length = deliveryVehicleManager.waitPlaces[id - 1].Length;
Array.Resize(ref deliveryVehicleManager.waitPlaces[id - 1], length + 1);
```

자동차를 추가 해주는 경우 위와 같이 해당 대기장소의 배열의 크기를 1 증가해주는 식으로 구현 하였습니다.

```
prio = int.Parse(tmpreadline[4].Substring(1));
```

또한, 대기장소를 입력받을 때 W1, W2와 같은 방식으로 입력받기 때문에, 이 입력값에서 1,2 숫자부분을 뽑아내기 위해서 위와 같이 Substring 방식으로 잘라내는 식으로 작성하였습니다.

```
//우선순위가 높은 것부터 낮은 순으로 정렬을 아래와 같은 방식으로 해준다.
DeliveryVehicle tmp = new DeliveryVehicle();
for(int i=0;i<deliveryVehicleManager.waitPlaces[id - 1].Length-1;i++)
{
    for(int j=0; j<deliveryVehicleManager.waitPlaces[id - 1].Length - 1-i;j++)
    {
        if (deliveryVehicleManager.waitPlaces[id - 1][j].priority > deliveryVehicleManager.waitPlaces[id - 1][j+1].priority)
        {
            tmp = deliveryVehicleManager.waitPlaces[id - 1][j];
            deliveryVehicleManager.waitPlaces[id - 1][j] = deliveryVehicleManager.waitPlaces[id - 1][j + 1];
            deliveryVehicleManager.waitPlaces[id - 1][j + 1] = tmp;
        }
    }
}
```

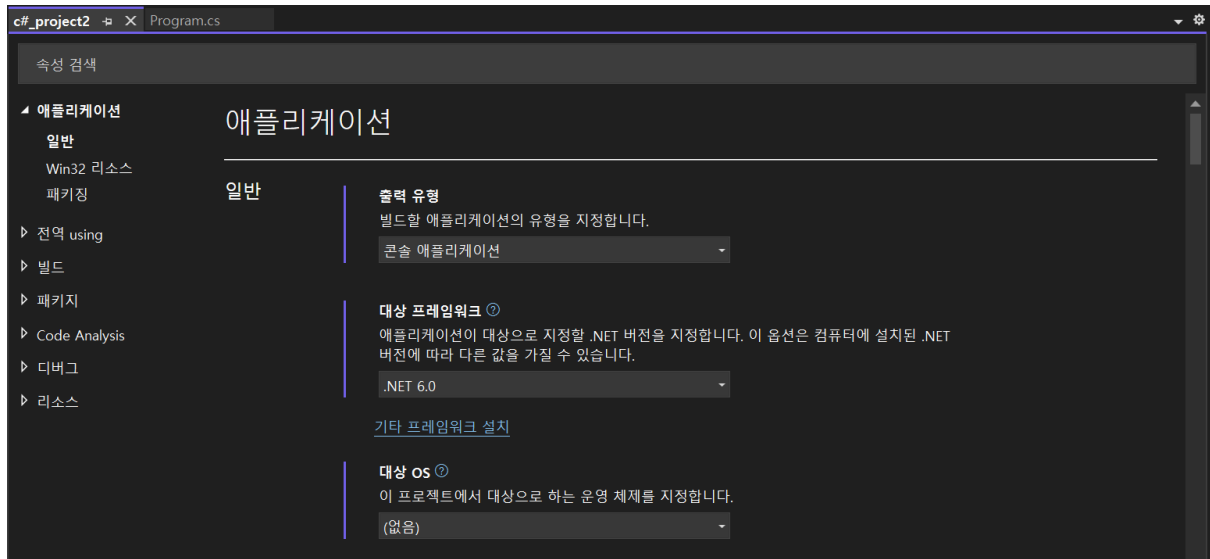
또한, 우선순위가 높은 순부터 정렬해야 하기 때문에 위와 같은 정렬알고리즘을 추가해두었습니다.

### 3. 파일 입출력 알고리즘 및 구현정도

첫째 줄에서 대기장소의 수를 입력받고, 그 이후 ReadyIn 일 경우 배달차를 특정 대기장소에 배정, Ready 일 경우 가장 공간이 작은 대기장소 배정, Status 일 경우 대기장소의 배달차 정보 출력,

Cancel을 입력받을 경우 해당 배달차를 대기장소에서 삭제, Deliver을 입력받을 경우 해당 대기장소의 우선순위가 가장 높은 배달차 배달보내기, Clear일 경우 해당 대기장소의 배달차들의 대기 취소, Quit을 입력받을 경우 프로그램 종료와 같은 위와 명령어들의 내용을 모두 구현하였습니다.

#### 4. 개발환경



NET 6.0 의 프레임워크로 개발을 진행하였습니다.