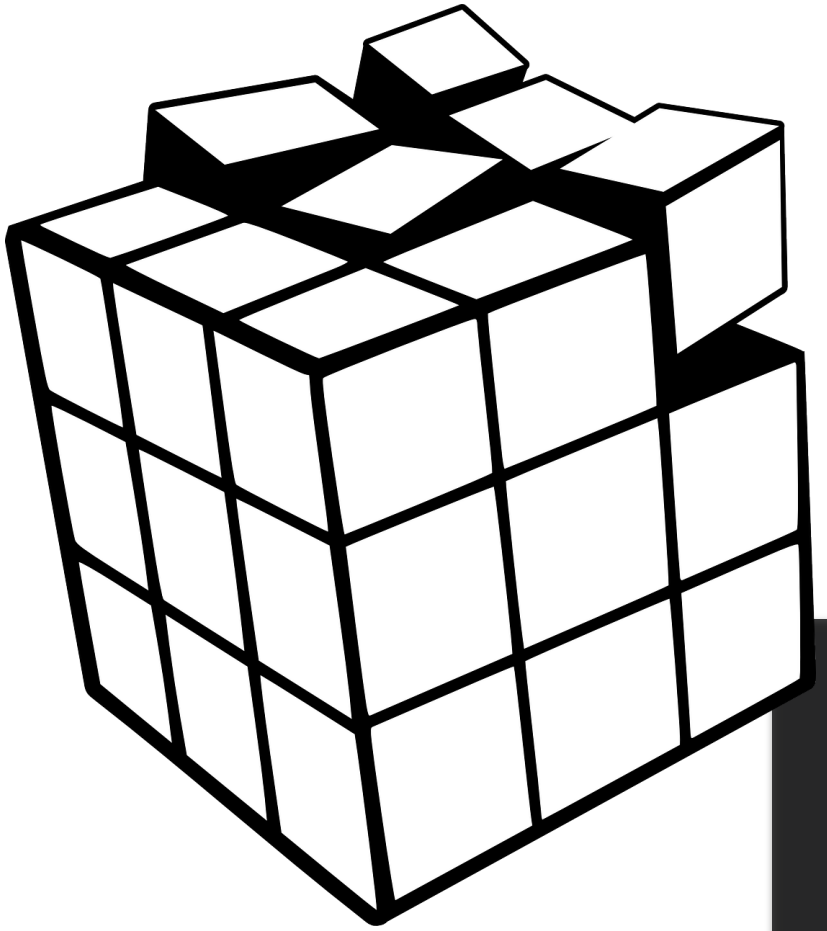


4X4 퍼즐 확장하기

4조. 김상주, 서장일, 양승현, 명승훈

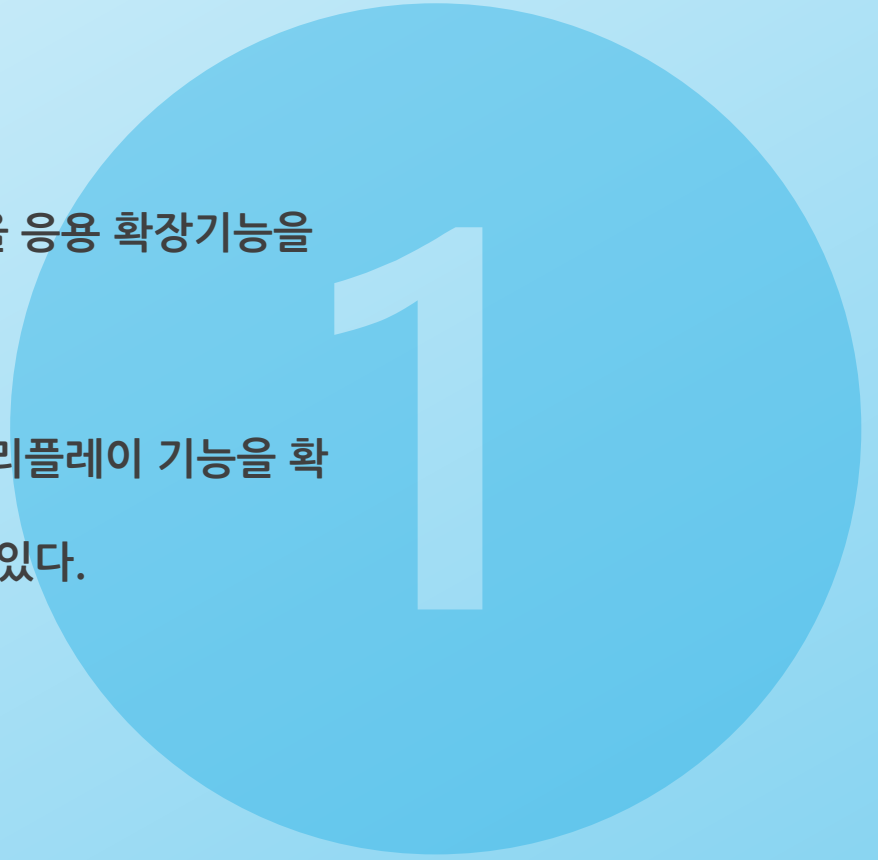
목차

브레인 스토밍

- 단순 파일 처리로 프로그램의 모든 데이터를 저장한다. (퍼즐 맵, 게이머의 움직임, 퍼즐 조각을 움직인 횟수, 시각 정보 등을 모두 저장하여 불러오기)
- 3차원 배열[PAGE][ROW][COL] 을 사용하여 배열을 저장하고 PAGE에 따라 플레이어의 움직임을 저장하고, 게임 저장, 이어가기 기능을 구현한다.
- 기본적으로 파일 처리를 이용하여 데이터를 저장하나, 1번의 방식과 다르게 퍼즐의 크기(DIM)를 기준으로 데이터를 나누고 추가하거나 출력을 가능하게 한다. 이를 이용하여 제시된 확장기능을 수행한다.

브레인 스토밍

- 제시된 문제에서 공통점인 플레이어의 움직임을 기록하여 기존 프로그램을 응용 확장기능을 수행한다.
- 최초 섞인 퍼즐 파일을 저장하고, 플레이어의 움직임(방향키)을 기록하여 리플레이 기능을 확장한다. 이를 기반으로 게임 저장, 불러오기 & 이어가기 기능을 확장할 수 있다.



알고리즘

- 각각의 문제 별로 대략적인 알고리즘을 구상하여 코딩에 임하였습니다.

■ 문제 해결을 위한 알고리즘

1. 퍼즐 맵의 크기 확장 : 게임을 3 x 3과 5 x 5로 확장하라.

define 되어 있는 DIM의 숫자를 변경함에 따라 3 x 3 혹은 5 x 5로 확장할 수 있다.

2. 리플레이 기능 : 경기가 끝나고 나면 게이머가 움직인 내용들을 다시 순서대로 보여준다. 이를 위해, 최초의 섞인 퍼즐 맵을 저장해두어야 하며, 게이머의 움직임도 모두 저장해야 한다.

프로그램의 시작이 SHUFFLECOUNT의 횟수만큼 퍼즐이 움직인 후 nMoveCount를 0으로 초기화된다. 이후 nMoveCount가 증가할 때마다 nMoveCount(퍼즐을 움직인 횟수)와 2차원 배열로 되어있는 퍼즐을 저장하여 게임이 끝난 이후 nMoveCount의 첫 번째부터 마지막 수 까지 대응되는 배열을 출력하여 리플레이 기능을 제작한다. 이후 정상적으로 게임이 종료된다면(저장 및 종료가 아닌 경우) 저장되어 있는 파일을 삭제하여 다음 퍼즐 게임의 데이터 오류를 보완한다.

3. 게임 저장 기능 : 경기 중에 현재 게임 상태를 파일에 저장하고 경기를 종료하는 기능을 구현하라. 물론 현재까지 퍼즐 조각을 움직인 횟수나 시각 정보도 저장되어야 한다.

SHUFFLECOUNT의 횟수만큼 퍼즐이 움직인 후 nMoveCount를 0으로 초기화, 이후 nMoveCount가 증가할 때마다 nMoveCount와 2차원 배열을 저장한다. 저장된 파일은 다음 게임 시작시 이어하기 기능을 사용하지 않으면 초기화가 된다.
저장기능은 특정키를 누르면 작동하며, 방향키와 같이 작동할 수 있도록 한다.

4. 저장된 게임 이어가기 : 저장된 경기를 계속 이어서 진행할 수 있는 기능을 구현하라. 물론 시작 시각과 중간에 종료한 시각도 저장 되어야한다.

만약 저장되어있는 파일이 있다면, 만약 저장된 파일이 남아있다면 이어하기를 진행할 것인지 물어본 후 만약 진행하겠다고 하면 가장 최근 2차원 배열을 불러오기 한 후 게임을 진행한다, 이후 리플레이 기능까지 수행, 만약 진행하지 않겠다고 하면 이전 파일을 삭제한 후 처음부터 시작한다.

5. 그림 퍼즐 기능 : 퍼즐 조각에 숫자가 아니라 문자가 출력되도록 해보라. 이를 위해 1장을 참조해 간단한 아스키 아트로 4x4 그림을 만들어야 할 것이다.

그림퍼즐(GameDisplay 에서 수정) 배열에 +64를 추가하여 대응되는 아스키코드값을 출력하면 알파벳 순서대로 출력된다.

6. 4.7절의 랭킹 보드 프로그램을 참고하여 상위 10개의 게임을 랭킹 파일에 저장하라.

프로젝트 역할 분담

- **김상주**
  - 프로젝트 리드 (영역 분배 및 진행도 확인)
  - 저장 방식 알고리즘 구상
  - 팀원의 함수를 통합하여 프로그램에 적용
  - 전체적인 코드진행, 디버깅 작업 및 디테일 추가
  - 리플레이 함수 제작

3

프로젝트 역할 분담

- **서장일**
  - 저장된 게임 이어가기 기능 구현
  - 이어가기 기능에 세부적인 요소 추가

3

프로젝트 역할 분담

- **명승훈**
  - 게임 저장 기능 구현
  - 저장 기능에 세부적인 요소 추가

3

프로젝트 역할 분담

- **양승현**
  - 그림 퍼즐 기능 구현
  - 알파벳으로 출력 요소를 변경
  - UI등의 세부적인 요소 추가

3

프로젝트 역할 분담

■ 프로젝트 팀원별 역할 분담

1.	양승현	static void f_GameInit();
		# 게임 초기화 함수
		static void f_GameDisplay();
		# 맵과 이동 횟수, 소요시간 등 게임 상황을 화면으로 출력 함수
		{
		# 알파벳 퍼즐(5번 문항) 출력
		}

2.	서장일	static int f_ArrowGetKeyDirection(); # 방향키를 받아들이는 함수
		{
		# 입력된 키가 「Enter」라면, break
		# 입력된 키가 「A」라면, 이어 하기(확장 기능)
		}

3.	양승현	static void f_PuzzleShuffle(int nShuffleCount);
		# 퍼즐 조각을 nShuffleCount번 이동해 섞는 함수
		{
		static bool f_PuzzleArrowMove(int nArrowDirection); # 퍼즐 이동 함수
		}

		static bool f_PuzzleIsDone(); # 퍼즐이 다 맞춰졌는지 검사하는 함수 {
		}

4.	명승훈	static bool f_PuzzleIsDone(); # 퍼즐이 다 맞춰졌는지 검사하는 함수
		{
		static int f_ArrowGetKeyDirection(); # 방향키를 받아들이는 함수
		{
		# 입력된 키가 「방향키」라면, 이동 함수
		# 입력된 키가 「S」라면, 저장하고 게임 종료
		}
		static bool f_PuzzleArrowMove(int nArrowDirection); # 퍼즐 이동 함수
		+ # 저장 함수(확장 기능)
		}

5.	김상주	# 프로젝트 리드(역할 분배 및 진행도 확인) # 배열을 파일로 저장 # 리플레이 함수 제작 # 리플레이시 따로 저장된 별도의 파일("별도의 파일이라 함은, 저장, 이어하기 기능에 필요한 파일과는 별개의 파일이다. 즉 파일을 2개이상 사용하게 된다.")을 사용하고, 이 파일은 모든 로직이 끝난후 가장 마지막 단에서 실행되게 한다. # 팀원의 함수를 통합하여 프로그램에 적용 # 전체적인 코드 디버깅

소스코드 – 리플레이

- 리플레이를 위해 추가한 함수

```
//----------------------[리플레이 확장 기능을 위해 추가한 함수]----------------
/// <summary>
/// 리플레이를 위해 2차원 배열을 1차원 배열로 저장하는 함수
/// </summary>
/// <param name="filename">: 사용되는 파일 명칭</param>
/// <param name="ArrayReplay">: 복사한 배열 입력</param>
static void f_ReplayDataSave(const char* filename, int* ArrayReplay);
/// <summary>
/// 리플레이 함수
/// </summary>
/// <param name="filename">: 사용되는 파일 명칭</param>
/// <param name="nCount">: 플레이어가 움직인 횟수(==nMoveCount)</param>
static void f_ReplayGame(const char* filename, int nCount);
/// <summary>
/// 리플레이 아스키아트 출력 함수
/// </summary>
static void f_ReplayPrint();
```

- 리플레이를 위한 구조체

```
/// ...
struct structReplayData
{
    int ArrayReplayData;
    int nReplayCount;
};
```

- 리플레이를 위한 배열 복사

```
//----------------------[확장 기능을 위해 추가한 부분]---------------
static int ArrayMapCopy[DIM * DIM] = { 0 }; //퍼즐 맵 복사 배열
static structReplayData arrayReplay[256]; //리플레이 저장을 위한 배열 초기화
```

소스코드 – 리플레이

- 리플레이를 위한 데이터 저장 함수

```
//리플레이만을 위한 데이터 저장 함수
static void f_ReplayDataSave(const char* filename, int* ArrayReplay)
{
    FILE* fp = fopen(filename, "a");

    if (fp == NULL)
    {
        return;
    }

    for (int i = 0; i < (DIM * DIM); i++)
    {
        fprintf(fp, "%4d", ArrayReplay[i]);
    }
    fclose(fp);
}
```

- 리플레이 구현 함수

```
static void f_ReplayGame(const char* filename, int nCount)
{
    FILE* fp = fopen(filename, "r");

    int nPlayerMove = 0;

    for (int j = 0; j < nCount + 1; j++)
    {
        for (int i = (j * (DIM * DIM)); i < (DIM * DIM) + (j * (DIM * DIM)); i++)
        {
            fscanf(fp, "%d ", &arrayReplay[i].ArrayReplayData);
        }

        f_ReplayPrint();
```

소스코드 – 리플레이

- 리플레이 구현 함수

```
    for (int i = (j * (DIM * DIM)); i < (DIM * DIM) + (j * (DIM * DIM)); i++)
    {

        if (arrayReplay[i].ArrayReplayData == 0)
        {
            printf("  ■ ");
        }
        else
        {
            printf("%3c ", arrayReplay[i].ArrayReplayData + 64);
        }

        //printf("%3c ", arrayReplay[i].ArrayReplayData + 64);

        if ((i + 1) % DIM == 0)
        {
            printf("\n");
        }
    }
    printf("\n");
    Sleep(REPLAY_PRINT_SPEED);
}

fclose(fp);
```

- 리플레이 화면을 위한 아스키아트

```
static void f_ReplayPrint()
{
    system("cls");

    printf("_____                 _                 ____                                \n");
    printf("| ___ \\               | |               |  __ \\                               \n");
    printf("| |_/ /  ___  _ __   | |  __ _  _   _  | | \\/  __ _  _ __ ___    ___ \n");
    printf("|    /  / _ \\| '_ \\ | | / _` || | | | | | __  / _` || '_ ` _ \\  / _ \\\n");
    printf("| |\\ \\ |  __/| |_) || || (_| || |_| | | |_\\ \\| (_| || | | | | ||  __/\n");
    printf("\\_| \\_| \\___|| .__/ |_| \\__,_| \\__, |  \\____/ \\__,_||_| |_| |_| \\___|\n");
    printf("             | |                 __/ |                                \n");
    printf("             |_|                |___/                                 \n\n");

    printf("    4조의 퍼즐 게임을 플레이해 주셔서 감사합니다!\n\n");
    printf("    아래는 귀하께서 퍼즐을 풀이하신 화면입니다.\n\n\n");
}
```

소스코드 – 저장 기능

- 저장 기능을 위한 기능 함수

```
//----------------------[저장 확장 기능을 위해 추가한 함수]----------------------
/// ...
static void f_PuzzleDataSave(const char* filename);
/// ...
static int f_PuzzleMoveOrSave();
/// ...
void f_SavePrint();
```

- 특정 키를 입력 받아 저장 활성화를 위한 선언

```
enum SaveSign { SAVE = 115 }; //아스키코드 10진수 115는 문자 's'
```

- 저장 함수에 사용되는 아스키아트 출력 함수

```
static void f_SavePrint()
{
    system("cls");

    printf("\t _____                          _____                    _              \n");
    Sleep(ASKIART_PRINT_DELAY);
    printf("\t/  ___|                        |  __ \\                  | |             \n");
    Sleep(ASKIART_PRINT_DELAY);
    printf("\t\\ `--.   __ _ __   __ ___   __ | |_/ / _   _  ___ ____| | ___         \n");
    Sleep(ASKIART_PRINT_DELAY);
    printf("\t `--. \\ / _` |\\ \\ / / / _ \\ |  __/ | | | ||_  /|_  /| | / _ \\       \n");
    Sleep(ASKIART_PRINT_DELAY);
    printf("\t/\\__/ /| (_| | \\ V /| |  __/ | |    | |_| | / /  / / | ||  __/ _  _ \n");
    Sleep(ASKIART_PRINT_DELAY);
    printf("\t\\____/  \\__,_|  \\_/  \\___| \\_|     \\__,_|/___|/___||_| \\___|(_)(_)\n\n\n");

    const char* strSaveSummary1 = ("\t\t\t게임을 저장하였습니다.");
    const char* strSaveSummary2 = ("\t\t\t프로그램을 종료합니다.");
```

소스코드 – 저장 기능

- 저장 기능을 위한 기능 함수

```
//입력된 키가 방향키라면 움직이고 "S"키 라면 프로그램을 저장하고 종료하는 함수
static int f_PuzzleMoveOrSave()
{
    int nKey = _getche();
    int nReturnKey = 0;

    if (nKey == SAVE)
    {
        f_SavePrint();
        f_PuzzleDataSave("PuzzleDataSave.txt");
        exit(0);
    }
    else
    {
        nReturnKey = f_ArrowGetKeyDirection(nKey);
    }
    return nReturnKey;
}
```

```
//퍼즐 데이터를 이차원배열로 저장하는 함수
static void f_PuzzleDataSave(const char* filename)
{
    FILE* fp = fopen(filename, "w");

    clock_t   tEndTime = clock();
    double tDuration = (double)(tEndTime - tStartTime) / CLOCKS_PER_SEC;

    if (fp == NULL)
    {
        return;
    }

    for (int i = 0; i < DIM; i++)
    {
        for (int j = 0; j < DIM; j++)
        {
            fprintf(fp, "%4d ", ArrayMap[i][j]);
        }
    }
    fprintf(fp, "%3d%6.1f", nMoveCount ,tDuration);

    fclose(fp);
}
```

소스코드 – 그림 퍼즐 기능

- 가장 간단하게 풀어본 알파벳 퍼즐

```
static void f_GameDisplay()
{
    system("cls");

    printf("\tFifteen Puzzle\n\t");
    printf("--------------\n\t");

    for (int i = 0; i < DIM; i++)
    {
        for (int j = 0; j < DIM; j++)
        {
            if (ArrayMap[i][j] > 0)
            {
                printf("%3c", ArrayMap[i][j]+64); //그림퍼즐 완성!!!!!!!

            }
            else
            {
                printf("   ");
            }

        }
        printf("\n\t");
    }
```

- 아스키 코드 10진수 65는 문자 'A'임을 이용하여 간단하게 해결

```
printf("%3c", ArrayMap[i][j]+64);
```

실행 화면

- 초기 화면

- 리플레이 화면

실행 화면

- 저장 화면

4

프로젝트 결과 및 소감

- 1, 2, 3, 5, 6번 문제를 해결하였으며, 4번 문제는 방향키가 중복적으로 적용되어 조건문을 무시하는 경우가 있는 것으로 추정하는 중이며 이를 해결하려 노력을 했으나 디버깅에 실패하였습니다.

소감

- 김상주
- 프로젝트 목적인 4 x 4 퍼즐게임 기능 확장을 완벽히는 수행하지 못하였습니다. 그러나 실패라고는 생각하지 않습니다. 저는 이번 프로젝트에서 한 가지 모험을 했습니다. 누군가에게 코딩을 의존하는 것을 방지하고자 팀원 모두에게 영역을 나누어 프로젝트를 제시하였고 이를 기반으로 팀을 이끌었습니다. 이러한 과정에서 저희들은 모두 한 단계씩 성장했다고 저는 자부할 수 있습니다. 근거로 팀장인 저부터 팀원들의 코드를 세세히 파악하여 통합하는 과정 속에서 수많은 시행착오를 거치며 많은 오류를 고쳐나갔으며, 팀원들은 각자의 영역에서 어떻게 하면 주어진 목표를 달성할 수 있을까? 하는 고민을 거쳐 기능 함수들을 만들어왔습니다. 이는 분명한 성장이라 생각하며 “어려워서”, “실패할까 두려워서” 하지 못할 것 같았던 높은 벽들을 부시며 함께 이 자리에 왔습니다. 비록 모두 해결하지는 못하였지만 저는 팀원들이 자랑스럽습니다.

소감

- 서장일
- 길다면 길고 짧다면 짧은 2주간의 중간 프로젝트를 진행하면서 지치고 힘든 부분이 많았습니다. 처음 1주 차에는 주어진 형식의 코드를 이해하는데 많은 시간을 써서 시간이 부족했다면 2주 차에는 간간이 발생하는 오류부터 컴파일 오류가 발견되지 않더라도 정확한 값을 받아오지 못하는 오류가 발생해서 골머리를 앓았습니다. 저희 조는 디스코드로 새벽까지 코딩 작업을 하곤 했는데 사기가 떨어지고 힘들어하는 모습이 보일 때마다 팀장인 김상주가 사기를 북돋아 주었고, 팀원들이 해결되지 않는 부분은 힌트를 줘서 팀원들 스스로 해결할 수 있게끔 도움을 줘서 아직 부족한 부분이 많지만 그래도 어느 정도 얻어 가는 부분이 있다고 생각해서 좋았습니다. 팀장이 각자의 역량에 맞춰 역할분담을 해준 덕분에 여기까지 올 수 있었다고 생각합니다. 끝으로 텀 프로젝트를 진행함으로써 교수님께서 분업을 강조하신 이유를 알 것 같습니다.

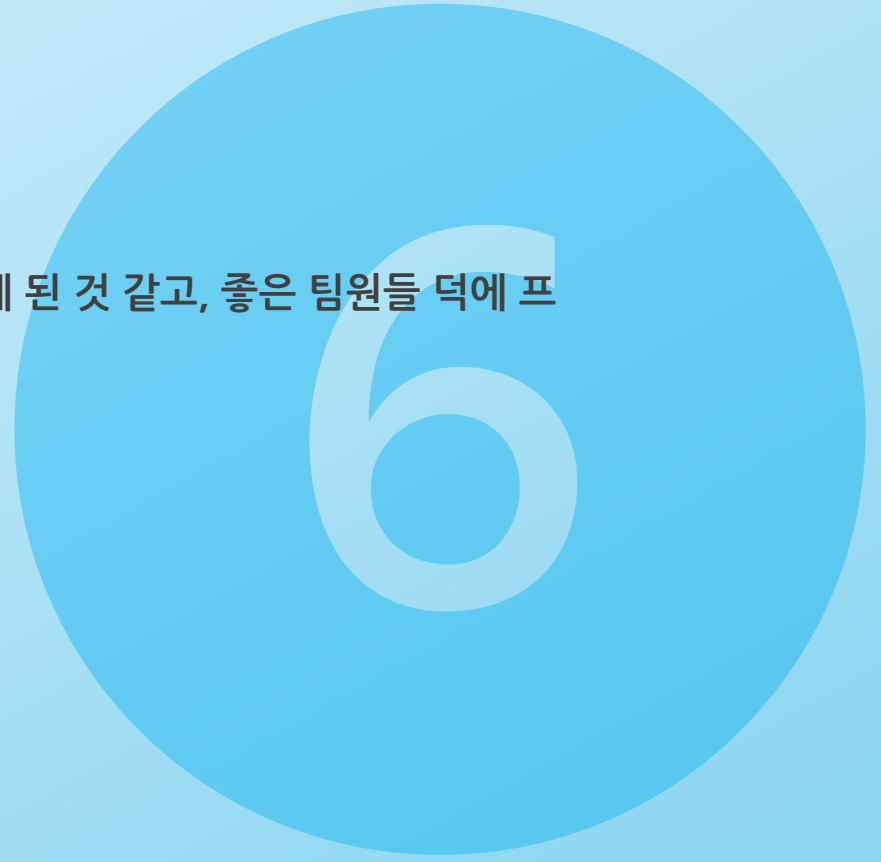
소감

명승훈

- 이번 프로젝트를 하면서 이거 내가 할 수 있을까라는 생각이 먼저 들었지만 팀장님이 우리를 잘 이끌어주었기에 할 수 있었으며 내가 직접 코딩을 하면서 프로그램이 어떤 알고리즘으로 돌아가는지와 내 프로그램과 다른 사람의 프로그램을 합칠 때 주의할 점도 배웠던 거 같다. 또 팀원들 모두 프로젝트 참여에 우호적이어서 이상 없이 프로젝트를 할 수 있었던 거 같다.

소감

- 양승현
  - 수업에서 제대로 이해하지 못했던 부분들을 팀원들과 프로젝트를 통해 알게 된 것 같고, 좋은 팀원들 덕에 프로젝트를 잘 마친 것 같습니다.



감사합니다.