<1>
안녕하세요. 게임 <오목>을 발표할 H조입니다.

<2>
발표는 게임 소개, 3*3 형식의 틱택토게임, 10*10 형식의 오목, 게임실행 순으로 진행하도록 하겠습니다.
우선 첫번째 게임 소개 파트에서는 저희가 오목게임을 어떻게 고르게 되었는지 그 이유와 게임방법을 소개해드릴 예정입니다. 게임 구현 시에 사용한 함수도 먼저 소개해드리겠습니다.
두번째 틱택토 파트에서는 기본틱택토의 전체 코드를 먼저 보여드리고 구현된 코드를 한 부분씩 자세히 분석해볼 예정입니다. 그런 다음 몇 가지 조건을 추가하여 발전된 틱택토 게임의 코드를 구현하고 또 분석해보도록 하겠습니다.
세번째 오목 파트에서는 틱택토보다 더 심화된 코드로 구현해 볼 것인데요, 틱택토 파트와 마찬가지로 먼저 전체 코드를 보여드린 후 구현하고 분석하는 순서로 진행하겠습니다. 또한 조건을 여럿 추가하고 최종적으로 완성된 오목 게임의 코드를 구현, 분석해 볼 것 입니다.
마지막으로는 게임 실행 파트를 통해 직접 오목게임을 보여드리는 시간까지 가질 예정입니다

<3>
혹시 어릴 적 오목게임을 한번이라도 해본 적 있으신 분들 계실까요?
학창시절에 책상위에 직접 오목판 그려서 했던 것처럼 다들 한번씩은 오목게임을 해본 경험이 있으실텐데요. 이와 같이 오목은 저희 추억의 게임이자 전통놀이라고 생각될 만 한 게임입니다. 그만큼 게임방식도 간단하고 이해하기 쉬운 게임이라고 할 수 있습니다. 그리고 프로그램 코딩을 하는 과정에서 칸 수에 따라 다양한 난이도로 조절할 수 있을 것 같아 심화시키기 좋을 것 같아 선정까지 하게 되었습니다.

<4>
그럼 틱택토의 게임방법을 설명해드리도록 하겠습니다.
이 게임은 기존 오목 게임이 단 둘이 플레이 하는 것처럼 플레이어 X 와 O 단둘이 진행합니다. 판은 삼목 기준 가로,세로 3x3칸으로 총 9칸이며, 플레이어가 번갈아가면서 칸을 채우는 형식입니다. 칸을 채우는 것을 반복하다 가로, 세로, 대각선에 같은 모양의 칸을 먼저 채우는 플레이어가 승리하는 방식의 게임입니다.

[사용함수]
<5>
코드에 사용된 함수부터 먼저 짚고 넘어가도록 하겠습니다.
Display함수는 사용자에게 텍스트나 숫자 등의 형태로 정보를 시각적으로 제공하기 위해 화면에 출력하는 함수입니다.
갯차함수는 표준 입력에서 한 문자를 읽어오는 함수입니다.
스트링카피함수는문자열을 복사하는 함수로 한 문자열을 다른 문자열에 복사합니다.
스트링렝스는 문자열의 길이를 반환하는 함수입니다.
Beep함수는 소리를 내는 함수입니다.

스트링컴페어함수는 두개의 문자열을 비교하는 함수입니다.
셋콘솔텍스트어트리뷰트함수는 콘솔텍스트의 색상을 변경하고 텍스트를 출력할 때 색상을 설정하는 함수입니다.
<6>
사용함수를 모두 보았으면 이제 코드로 넘어가보겠습니다.

[tic – tac – toe LAB part]
<7>
다음 보이시는 코드는 틱택토 게임을 구현한 전체코드입니다.

<8>
여기서 가장 상단의 헤더파일를 먼저 보겠습니다.
첫 번째 줄의 define _ CRT_SECURE_NO_WARNINIGS 는 "안전성의 이유로 발생하는 에러를 방지하기 위한 정의"를 선언하는 문장으로, 저희는 다들 아시는 스캔에프 함수를 쓸 것이기 때문에 필요합니다.
두 번째 줄의 include 스탠다드아이오 는 기본 입출력을 위한 라이브러리 코드입니다.
세 번째 줄의 define BOARD_SIZE 3 는 2차원인 틱택토의 보드 사이즈를 3으로 설정하는 단계입니다

<9>
다음은 바둑판을 보여주는 void display 문을 상세히 살펴보도록 하겠습니다.

<10>
변수 int i, j는 보드의 가로세로 인덱스를 의미합니다.
네 번째 줄의 void display는 B값인 O와 X를 넣기 위해 캐릭터 배열을 설정해주고 아까 3으로 할당된 크기의 게임보드를 출력하는 역할을 수행합니다.
7번째 줄에 빨간 줄로 표시되어 있는 printf(" ")는 표시된 바와 같이 콘솔창에서 행과 열 사이의 빈 공백을 의미합니다.
다음 9번째 줄에 노랑색으로 표시되어 있는 printf투디인티저콤마아이는 콘솔창에서 출력되어 있는 숫자들의 자릿수를 의미하는데 여기서 투디 인티저로 표시한 이유는 자릿수를 넓게 잡아 보기 좋도록 하기 위함입니다. 이는 강의자료1의 chapter 3 부분을 참고하시면 좋을 것 같습니다.
10번째 줄에 초록색으로 표시되어 있는 printf는 콘솔창에서 바둑판의 형태를 잡아주는 테두리 부분에 해당합니다.

<11>
13번째 줄의 노란색으로 줄쳐진 printf 부분을 봐주시면 되겠습니다. 여기도 윗부분 코드처럼 가독성을 위해 3d 인티저를 사용하여 출력되는 숫자를 오른쪽으로 정렬하고, 최소한 3자리의 공간을 차지하도록 지정합니다. 여기서는 출력되는 숫자가 세자리보다 작기 때문에 왼쪽에 공백이 추가됩니다.
15번째 줄의 빨간색 영역은 콘솔창에서 보이는 보드판의 내부 영역을 보여주며 입력받을 OX 값을 출력하는 부분입니다.

<12>
다음은 보드판을 출력하는 void main() 문을 상세히 보도록 하겠습니다.

<13>
코드 분석에 앞서 변수들을 정리하자면
board는 보드 사이즈를,
Turn 은 플레이어 x를,
Int r,c 는 행과 열, int I,j 는 인덱스를,
int count는 게임을 진행한지 몇 번째 차례인지를 의미합니다.

이제 줄별로 하나하나 보도록 하겠습니다.
20번째 줄의 char board는 저희가 실행하고자 하는 보드 사이즈를 3x3으로 지정한 코드입니다.
21번째 줄의 char turn = 'X';는 게임을 시작할 플레이어를 x로 지정하기위해 첫 턴을 x로 설정한 코드입니다.
22,23,24번째는 아까 설명한 변수를 설정하는 줄이고,
25,26,27번째 줄의 for 문은 3인 보드사이즈에 따라 0,1,2 총 9번 동안 돌며 보드 안을 공백으로 채워줍니다.
28번째 줄에 따라 count값을 1로 설정해주는데, 왜냐하면 아래에 입력받을 때 두와일문을 사용하기 때문인데요, 두와일문은 조건이 참이냐에 관계없이 무조건한번은 실행되기 때문에 처음실행 카운트인 1로 초기화하고 시작하도록 합니다.
30번째 줄로 설정한 빈 보드판을 콘솔창에 보이도록 display 해줍니다.

<14>
다음으로는 가장 아래의 실행문인 두와일문을 보도록 하겠습니다.

<15>
먼저 가장 큰 바깥쪽 두와일문부터 보시면, 카운트 보드사이즈*보드사이즈라고 되어있는데 저희가 미리 앞에서 정해두었던 3*3 형식에 따라 아홉번을 실행하도록 설정하였습니다. 만약 이를 넘길 시 실행은 종료됩니다.
다음으로 안쪽 두와일문을 보겠습니다.
두와일문 안쪽은 현재 플레이어에게 행과 열을 입력받도록 안내하고, 입력받은 값을 r행과 c열 에 할당합니다. 이를 통해 플레이어는 자신의 차례에 해당하는 보드의 위치를 지정할 수 있게 됩니다.
여기 안쪽 두와일문의 와일을 보면 보드가 비어있을 동안 실행되어야 하므로
와일 보드가 빈칸이 아닐때 로 조건을 달아줍니다.

<16>
41번째 줄은 r행 c열인 모드에 x나 O에 해당하는 기호를 표시하도록 하고
42번째 줄은 이를 표시한 보드를 출력하는 역할을 합니다.
이제 파란으로 표시된 부분을 봐주세요

여기서는 삼항 연산자를 사용합니다.
삼항연산자는 물음표와 콜론으로 표현될 수 있는데요, 물음표를 기준으로, 여기서는 턴 값이 x면 O , O면 로 바꾸도록 하는 역할을 합니다.

<17>
이제 3*3의 콘솔창을 보도록 하겠습니다.
행 열 숫자값을 네번째로 입력한 부분을 봐주세요.
여기서 행 열을 1 2을 입력하면 앞에서 먼저 입력된 값이 1,2자리를 차지하고 있기 때문에 입력이 되지 않습니다.
마지막 결과를 보면 이렇게 9칸을 모두 채웠을 경우 종료됩니다.
<18>
이제 우승조건을 추가하여 코드를 발전시켜보겠습니다.

[tic – tac – toe hw part]

<19>
우선 우승조건함수를 정의하고 게임보드를 설정해줍니다.
<20>
우승조건을 수평,수직,대각,역대각 넷 중 하나로 보고,
<21>
플레이어의 입력값을 출력하여 우승여부까지 보이도록 합니다

<22>
우선 가장 위쪽 코드부터 볼텐데요, 다른것은 모두 동일하되 우승조건함수인 wincheck를 추가해주겠습니다. Wincheck 함수는 가로, 세로, 대각선 방향으로 일렬로 연속된 동일한 값을 가지는 요소들을 확인하여 승리 여부를 판단하는 역할을 합니다. 여기에선 3*3의 보드판의 r행 c열에서 이를 판단하게 됩니다.

[우승조건]

<23>
이제 우승조건의 초기변수를 설정해주겠습니다.
위에서 했던것처럼 wincheck함수설정이 완료되면
21번째 줄에서 수평수직대각역대각카운트값을 0으로 초기화 시켜줍니다.
또한 다음줄에서 board B의 rc행렬값을 문자열 오목으로 설정합니다.

<24>
이제 우승조건을 하나씩 살펴보겠습니다.
수평조건을 먼저 볼텐데요
수평조건은같은 행에서 다른열의 값이 같냐를 보기 때문에 행은 고정한 채로 인덱스 i를 하나씩 늘려가며 체크해주겠습니다.

이때 행렬 ri가 사용자가 둔 돌과 같으면, 즉 오목과 같으면 수평카운트를 하나씩 더해갑니다.
이 카운트가 연달아 3개면 빙고이므로 참이라는 의미의 1값을 리턴해줍니다.
하지만 만약 열값을 연달아 체크하는 과정에서 3개중 하나라도 같은 돌이 아니면 빙고가 아니므로 수평카운트는 다시 0이 되게 됩니다.

<25>
수직 조건도 수평조건과 마찬가지로 진행되지만
같은 열에서 다른 행의 값이 같냐를 보기 때문에 열을 고정한 채로 i 인덱스를 하나씩 늘려가며 체크해주겠습니다.
이때 행렬 ic가 사용자가 둔 돌과 같으면, 즉 오목과 같으면 수직카운트를 하나씩 더해갑니다.
이 카운트가 연달아 3개면 빙고이므로 참이라는 의미의 1값을 리턴해줍니다.
하지만 체크하는 과정에서 3개중 하나라도 같은 돌이 아니면 빙고가 아니므로 수직카운트는 다시 0이 되게 됩니다.

<26>
세번째 조건인 대각선 조건을 보겠습니다.
저희는 3가지의 다른 방법을 준비하였습니다.

우선 가장 간단한 첫번째 방법이 있습니다.
대각선인 행렬 00,11,22 값이 모두 동시에 omok과 같으면 대각선 카운트를 3으로 바로 입력해줍니다.
카운트 값을 3으로 입력하지 않고도 바로 return 1로 승리로 맞춰줄 수도 있습니다.

두번째 방법은 for문을 이용하는 방법입니다. I j 를 0부터 시작하여 하나씩 늘릴동안 실행합니다.
For문 안에 있는 와일문을 사용하여 해당 보드칸이 오목과 같으면, 즉 사용자가 둔 돌과 같으면 대각선 카운트를 셉니다.
이렇게 연달아 세서 3개가 된다면 1값을 리턴하여 승리로 간주합니다.
하지만 해당 보드칸이 오목과 같지 않으면 대각선 카운트는 다시 0으로 초기화됩니다.

3*3 틱택토에서 대각선 우승이 가능할 때는 사실 행과열이 같을 때 즉 행렬 0,0 1,1 2,2 뿐입니다. 따라서 방법 2와 같이 진행하는 경우는 볼 필요가 없는 그 이상의 값을 보고 있기 때문에 비효율적이라고 할 수 있습니다.

<27>
따라서 세번째 방법인 while 문으로 보겠습니다.
앞선 코드와 동일하게 대각선을 체크해야 하므로 i와 j를 0부터 시작하여 해당 보드칸이 오목과 같은 것이 연달아3개가 된다면 참이기 때문에 1값을 리턴해 줍니다.
대각선 값을 체크해야 하기 때문에 ij값은 하나씩 늘려가며 와일문을 돌립니다.
만약 연달아 세개 빙고가 되지 않았다면 와일문을 탈출하게 되므로 대각선카운트는 다시 0으로 초기화됩니다.

<28>
이제 역대각선 코드를 볼 건데요
앞선 대각선 코드와 마찬가지로 직접 비교하는 방법, for문을 쓰는 방법, 와일문을 쓰는 방법을 살펴보겠습니다.
첫번째 가장 간단한 방법에 따라 볼 때는 역대각선이기 때문에 행렬 0,2 1,1 2,0값을 확인해줍니다.
두번째 방법도 대각선과 비슷하되 역대각선을 체크해야 하기 때문에 i는 증가하고 j는 감소하도록 하여 for문을 돌려줍니다.
역대각 카운트가 연달아 세개가 되면 빙고이면 참이므로 1을 리턴해주고, while문 탈출시 역대각선 카운트는 0이 됩니다.
For문이용은 비효율적이므로 while문을 사용한 세번째 방법으로 발전시켜보았습니다.

<29>
세번 째 와일문은 역대각선을 체크해야 하므로 i는 0, j=2부터 시작하여 i는 하나씩 증가하고 j는 하나씩 감소합니다. 동일하게 와일문을 탈출하면 역대각선카운트는 0으로 초기화됩니다.

<30>
다음 코드는 리턴 값을 정리해놓은 것입니다.
우승조건에서 첫번째 직접 비교의 방법으로 한다면 필요한 코드인데요
카운트가 3이 되는 것만 지정하고 리턴 값은 지정하지 않았기 때문에 대각선과 역대각선의 조건에서 3일 경우 빙고가 되어 1을 리턴하고 아니라면 0을 리턴하는 코드를 추가해줍니다.

<31>
그 다음 메인함수는 기존 3*3코드와 동일하게 보드판을 만들고 플레이어가 입력한 값을 출력하는 역할을 합니다.
따라서 저희는 우승여부조건코드와 동점일 때의 코드를 살펴보겠습니다.
누가 이겼는지 체크하기 위해 wincheck라는 사용자지정함수를 사용하였구요
아까 받은 반환값으로 우승이 결정되게 될 것입니다
반환값이 1이라면 누군가가 이긴 것이기 때문에 누가 이겼는지 출력해주고 if문에 걸리지 않았다면 누군가가 우승한 것이 아니기 때문에 턴을 바꾸며 게임을 계속 진행하게 됩니다.
동점이라면 wincheck함수가 0이고 지정된 보드판의 칸 수에 하나 더한 것 이상일때 nobody wins가 출력됩니다. 보드판이 모두 채워졌음에도 실행된 게임의 차례인 카운트가 이를 넘어간다면 이긴 사람이 없는 것이기 때문에 동점이라고 할 수 있는 것입니다.

<32>
3*3발전코드의 콘솔창을 보며 실행이 제대로 되었는지 확인해보겠습니다.
첫번째 사진은 역대각선으로 플레이어 O가 이긴 경우입니다.
그 아래쪽 사진은 플레이어x가 수평빙고를 완성하여 이긴 경우입니다.
오른쪽 위의 사진은 잘못된 행렬을 입력한 경우라 적용되지 않고 다시 행렬을 입력하도록 합니다.
무승부이면 보이시는 콘솔창과 같이 Nobody win!으로 게임이 종료됩니다.

[수평 오목]

<33>
이제 3*3 틱택토에서 10*10 오목으로 난이도를 높여 진행해보겠습니다.

<34>
완전한 오목 코드를 구현하기 전 오목의 형태 중 수평으로 놓여진 오목인 수평오목 코드를 분석하고 발전시키도록 하겠습니다.
우선 수평오목은 앞선 tic-tac-toe 방식에 다른 조건들을 추가해 발전시킨 코드라고 볼 수 있습니다. 또한 틱택톡과 달리 돌이 5개가 완성되면 승리하는 게임인데요, 여기서 추가적으로 발전된 조건에는 무엇이 있는지 보도록 하겠습니다.

<35>
발전된 조건입니다. 만약 연속 여섯개 이상의 말이 놓인다면 우승조건에서 벗어납니다.
빈칸이 아닌 칸을 선택하면 다시 입력 프롬프트가 나오고 승리자가 없으면 Nobody wins!를 출력하게 됩니다.

<36>
다음은 앞서 발표한 tic –tac-toe를 발전시킨 수평오목의 전체 코드입니다.

<37>
다른 부분들은 모두 같지만 보드판 크기를 10으로 늘려야 하기 때문에 3번째 줄의 board_size를 10으로 설정하여 10*10과 같은 배열로 선언해줍니다.

<38>
다음은 int winCheck 문을 통해 수평오목 코드의 우승조건을 보겠습니다.
우선 보이시는 첫 번째 빨간 영역의 첫줄을 통해 index 값을 정수로 선언해주고 수평카운트를 0으로 초기화 시킵니다.
그리고 두 번째 줄을 통해 플레이어에게 입력받은 돌은 omok키로 설정하도록합니다.

가운데 for문도 위의 코드와 동일하지만 수평오목이기 때문에 가로줄인 r행은 고정하고, 세로줄인 i인덱스값은 하나씩 키워가며 반복하도록합니다.
또한 5개를 연속으로 맞춰야하고 5개가 초과해서 나열되면 안된다는 조건때문에 수평카운트가 5여야하고 다음칸은 빈칸이라는 조건이 추가되어야합니다.
마지막으로 i++하며 연달아 체크하는 과정에서 하나라도 사용자가 둔 돌과 다르다면 빙고가 될 수 없으므로 그런 경우 수평 카운트는 다시 0이 되도록 합니다.

<39>
아래의 if문은 명시적으로 보여주기위해 추가한 부분입니다. 틱택토에서는 3개였지만 오목에서는 수평 연속이 5개면 1값을 돌려주어 우승을 나타내고 아니라면 0값을 리턴하여 원래상태로 초기화합니다. 하지만 저희는 위의 조건문에서 이미 5개연속이고 다음칸은 비어있어야한다는 조건을 걸었기 때문에 위의 반복문을 통해 수평카운트 값이 5가 된다면 우승을 나타내는 1을, 그게 아니라면 패를 나타내는 0을 돌려주게 됩니다.

<40>
아래쪽 구문은 앞의 3*3 tic-tac-toe와 동일합니다. 보드판을 빈칸으로 만들어주고, 출력하는 역할을 합니다.
do ~ while 문은 플레이어의 턴을 바꿔주고 우승여부와 출력을 담당합니다.

<41>
이제 콘솔창을 보며 다양한 결과를 비교해보겠습니다.
(애니메이션1)첫번째 경우 x가 다섯개연속이지만 옆에 O가 있으므로 다음턴으로 넘어가게됩니다.
(애니메이션2)두번째 경우도 마찬가지로 O가 6개 이상이기 때문에 우승하지 못하고 다음 턴으로 넘어갑니다.
(애니메이션3)세번째 경우는 앞선 3*3 코드와 마찬가지로 이미 입력되어있는 자리에 또다시 입력을 시도하였을 때 다시 입력받도록 하는 것입니다.
(애니메이션4)마지막으로 O옆에 빈칸이 있고, 5개 연속 O가 차지하였기 때문에 O의 승리로 끝나게 됩니다.

[오목]
<42>
다음으로는 앞서 본 수평오목을 발전시켜 생성한 오목 코드를 보도록 하겠습니다.

<43>.
오목의 경우 수평오목 방식에 추가 기능 및 조건을 추가해준 게임입니다
여기서 추가기능 및 조건은 수평을 포함한 수평, 수직, 대각선, 역대간선의 경우를 추가하여 표현해주고 게임을 시작하기 전 게임의 진행 여부를 답변 받고 player가 자신의 닉네임을 설정하도록 합니다. 더불어 콘솔창에서의 게임판과 돌의 색상을 변경하고 중간중간 상황에 따라 효과음을 넣어주며 마지막으로 개임 승리 시 캐릭터가 출력되도록 추가하였습니다.

<44>
다음 보이는 코드가 오목의 전체 코드이고 이 중 헤더파일을 시작으로 세부적으로 분석해 보도록하겠습니다.

<45>
우선 보이시는 부분은 오목코드의 헤더파일 부분입니다. 두 번째 줄의 Include <Windows.h>는 Beep() 함수를 사용하기 위해 헤더 정의를 내리는 줄입니다.
앞선 수평오목에서의 헤더파일과 거의 동일하지만 오목 코드의 경우 효과음과 콘솔창의 색상변경 등과 같은 기능을 추가하기 위해 2번째 줄에서 보이시는 include <Windows.h> 코드를 이용합니다. 즉, 이는 Beep()나 getstdhandle(), setconsoleTextattrivute() 등의 함수를 사용하기 위한 헤더를 정의해 주는 것이라 볼 수 있습니다.
4번째 줄에 있는 include <string.h> 코드는 스트링컴페어와 스트링카피를 사용하기 위한 헤더 정의 부분으로 보시면 되겠습니다.

<46>
다음은 콘솔창의 입출력, 색상 설정 등과 관련된 작업을 수행하는데 사용되는 handle console 코드를 보겠습니다. 보이시는 Handle hconsole = getstdhandle()문을 통해 콘솔 핸들을 불러옵니다. 그 후 SetConsoleTextAttribute(hConsole, 14)가 사용된 이유는 바둑판과 비슷한 밝은 노랑 색상으로 콘솔의 텍스트 색상을 변경하기 위해서입니다. 이 색상은 바둑판의 배경 색상을 나타내기 위해 사용되었으며 참고로 14번은 밝은 노랑 색상에 해당하는 번호입니다.

<47>
바둑판의 배경 색상을 설정한 후, 바둑돌의 위치를 출력하는 for 반복문 안에서 SetConsoleTextAttribute(hConsole, 15)가 사용되었습니다. 이는 바둑돌의 색상을 기본 색상으로 변경하기 위한 것이며 15번이 기본 색상에 해당하는 번호입니다.. 바둑돌의 색상을 바꾸기 위해서는 이전에 설정한 색상을 되돌려야 하기 때문에 이 코드를 사용하여 기본 색상으로 변경한 후에 바둑돌을 출력합니다.
그리고 그 안의 for문과 if ~ else 문을 이용해 x의 차례일 경우 백돌을, o의 차례일 경우 흑돌을 출력하며 돌을 구분해주고 순서가 번갈아가며 진행될 수 있도록 구현했습니다. 먼저 실행되는 X의 차례에 백돌을 출력하는 이유는 기존의 오목 게임이 그러하듯 백돌이 먼저 시작하는 것이 원칙이기 때문입니다.

<48>
여기서 SetConsoleTextAttribute(hConsole, 14)는 바둑판의 배경 색상을 설정하기 위해 사용되고, 이후에 SetConsoleTextAttribute(hConsole, 15)는 바둑돌의 색상을 기본 색상으로 변경하기 위해 사용되었음을 알 수 있습니다.

<49>
다음은 수평, 수직, 대각선, 역대각선 경우에 따른 조건을 보도록 하겠습니다. 우선 여기서 수평오목에서는 조건이 다섯개 연속임과 동시에 다음칸이 비어있어야 한다는 조건이 필요했지만 오목 코드에서는 사용자가 입력한 함수인 오목과 다르면 된다는 조건으로 바꾸어 작성했습니다.
그래서 수평의 경우 돌이 연달아 5개인 동시에 그 다음칸은 omok과 달라야한다는 조건을 달기 위해 and 연산자를 사용하였으며 수평의 경우 행은 고정된 채 열만 한칸씩 옆으로 나열되어진 형태여야 해서 1씩 증가하도록 설정했습니다.

여기서 주의 할 점은 앞선 수평오목 코드 작성 시에는 돌이 5개 연속일 때라는 조건 때문에 == ' '을 표현해주었지만 (애니메이션) 아래의 사진처럼 돌이 5개 연속 나열되었음에도 불구하고, 다음 칸에 다른 돌이 있었기에 오목이 완성되었다고 인식하지 못하였습니다.
이를 해결하기 위하여, 오목 코드에서 돌이 5개 연속인 동시에 다음 칸은 오목이 아님을 조건으로 수정하였습니다. 위의 조건이 성립하는 경우 return을 이용해 1을 그리고 위의 조건이 성립되지 않은 경우에 0을 돌려줍니다.
같은 방식으로 수직의 경우 열이 고정된 채 행이 한칸씩 위아래로 나열되어진 형태이기 때문에 행이 1씩 증가되도록 코드를 짜보았습니다.

<50>
다음 대각선의 경우는 왼쪽에서 오른쪽 아래로 가는 경우인데 이는 행과 열이 점차 한 칸씩 증가하는 걸로 표현할 수 있습니다. 그래서 보시는 것과 같이 행과 열을 1씩 증가시켜 코드를 짰고 반대로 역대각선은 오른쪽에서 왼쪽 아래로 내려오는 형태로 행은 한칸씩 증가하고 열은 반대로 한칸씩 감소하도록 코드를 짜 구현할 수 있었습니다.

<51>
위의 4가지 경우인 수평, 수직, 대각선, 역대각선 중 하나라도 개수가 5개인게 성립한다면 1을 그렇지 않다면 0을 돌려주어야합니다.

<52>
다음은 게임의 실행여부를 묻는 부분을 코딩해보겠습니다. 우선 char turn[8]을 통해 문자의 크기를 8로 지정해주었습니다.
게임을 실행할지에 대한 질문이 주어지고 그에 대한 답으로 대문자 Y를 입력하면 실행 그렇지 않으면 자동으로 종료되도록하고 종료되지 않고 진행된다면 "오목 게임을 시작하겠습니다"라는 문자이 함께 출력되도록 하였습니다.

<53>
다음으로는 변수 player마다 자신의 닉네임을 입력받아 사용할 수 있도록 하는 코드입니다.
우선 변수 palyer1 과 2의 문자 크기를 8로 지정해줍니다. 그 후 do ~ while문을 통해 각 변수의 글자수가 7 이상일 동안 즉 7 이하가 나올 때까지 "player의 닉네임을 7자 이내로 입력해주세요"라는 문장이 반복해서 나오도록 합니다. 다음은 오목 컬러라는 변수를 캐릭터형으로 선언한 후, 'x'를 대입하였습니다.

<54>
다음은 각 사용자의 차례에 행과 열을 입력받아 게임이 실행되도록하는 부분의 코드입니다. 두 번의 do ~while문이 사용되는데 두 번째로 사용되는 do~while문부터 보도록하겠습니다.
우선 처음 나오는 printf("Player %s(행 열): ", turn);에서 %s는 turn 변수의 값을 출력하는 역할을 합니다. 여기서 %c대신 %s를 사용하는 이유는 %c는 문자열 하나만 되지만 %s는 전체 다 가능하기 때문입니다. turn 변수는 player1과 player2의 이름을 번갈아가며 저장하는 변수입니다. 따라서 turn 변수의 값은 player1과 player2의 이름을 번갈아가며 저장하고, %s는 turn 변수의 값을 출력하기 위해 사용됩니다. 그 아래 scanf는 사용자가 돌을 놓고자하는 행과 열을 입력받는 코드이며 마지막 while(getchar()) 부분은 입력 버퍼를 비우는 역할을 합니다. 이 부분은 개행 문자('₩n')가 나올 때까지 입력 버퍼에 남아있는 모든 문자들을 제거하는 역할을 수행합니다. 이를 수행하는 이유는 사용자가 키를 누를 때마다 입력된 문자들이 입력 버퍼에 저장되고 프로그램은 입력 버퍼에서 데이터를 읽어와 처리하기 때문에 2개 이상으로 이상하게 입력받았을 때 입력 버퍼를 비우는 것입니다.

다음 아래에 있는 if 문의 경우 누군가 이미 돌을 놓은 자리에 입력한 경우 , 아니면 x 와 o가 맞게 자리를 입력한 경우에 따라 Beep함수를 사용하여 다른 효과음을 줄 수 있도록 설정하였습니다.
여기서 Beep함수는 음파와 대기시간을 입력받아 그에 따라 다양한 음을 표현할 수 있는 함수입니다.

<55>
이 부분에서는 우승자가 나왔을 때 콘솔창을 보여주는 코드입니다.
처음 앞서 설정한 wincheck 조건에 성립한다면 turn함수를 사용해 우승한 순서의 사용자를 가져와 "wins!"라는 단어와 함께 출력되도록 해줍니다. 그리고 여기서 다시 한번 더 콘솔창의 색상을 지정하고 변경해주기 위해 앞서 설명드렸던 handlehconsole 문을 가져와 선언해주고 색을 그에 맞게 지정해줍니다. 아래는 승리한 경우 앞에서 설정한 것과 또 다른 효과음을 지정해 주기 위해 beep함수를 4번 재생하도록 구현하였습니다.

<56>
이 경우 winner가 나오면 앞선 문장과 효과음들과 함께 캐릭터가 보이도록 print함수를 반복하여 코드를 짰습니다.
아래의 if else 문은 앞선 우승자가 나왔을 경우의 if 문의 else에 해당하는 부분으로 우승자가 나오지 않았다면 계속해서 x와 0가 서로 돌아가면서 omok을 이어나가도록하는 코드입니다.
그리고 마지막에 strcpy 함수와 strcmp함수가 나오는데 이는 삼항연산자 안에서 strcmp함수로 비교를 해주고 turn이 플레이어 O인지 x인지 정한 후 strcpy로 복사하여 바꿔주는 걸 말합니다.

<57>
마지막으로 등장하는 이 부분은 비길 시를 보여주는 코드입니다. wincheck 조건에 성립하지 않고 게임판의 개수 이상으로 카운트가 넘어가는 경우 이를 비겼다고 여기고 "nobody wins!" 라는 문장과 함께 비길 시에 나타나는 또 다른 효과음을 넣어주었습니다.
그리고 이 경우 우승자가 존재하지 않으니 return을 0으로 보내줍니다. (영상 재생- 뒤에 부분 마지막 입력하는 부분에서만 재생해주기)

<58>
다음으로는 지금까지 분석했던 오목게임을 함께 실행해보도록 하겠습니다.
시간 관계 상 게임의 실행은 저희가 미리 짠 순서에 따라 진행하도록하겠습니다.

.

-우선 코드를 실행시키면 게임을 실행하기 앞서 게임을 실행할지의 여부를 묻는 문장이 나옵니다. 이에 대문자 Y를 입력하고(Y입력하기) enter키를 눌러줍니다.
-그럼 "오목 게임을 시작하겠습니다!"라는 문장과 함께 player1의 닉네임을 7자 이내로 입력하라는 문장이 출력됩니다. 여기에 저는 닉네임을 H로 설정해주고(H입력하기) 7자 이내로 닉네임을 입력하였기 때문에 바로 player2의 닉넥임을 입력받는 것을 확인할 수 있습니다.
-만약 player2의 경우 닉네임을 7자 이내가 아닌 그 이상의 숫자를 입력하였을 경우를 보겠습니다.
-(0123456789 입력하기) 보시는 것과 같이 player의 이름이 옳게 입력될때까지 입력창이 반복되는 것을 볼 수 있습니다. 그럼 다시 player2의 이름을 제 임의로 솜솜이라고 지정하도록 하겠습니다.
- player들의 닉네임을 옳게 설정하면 보이는 것과 같이 바둑판이 나오는 것을 확인할 수 있는데요 , 앞서 콘솔창에서의 바둑판 색상을 14번 밝은 노랑으로 지정하였던 코드가 잘 실행되었음을 확인할 수 있습니다.
-이와 함께 게임의 실행 순서를 x였던 palyer1번부터 실행하도록 하였기에 player1인 H부터 게임을 시작하도록 하겠습니다. 게임을 실행하면서 입력하는 player에 따라 그리고 앞서 설명드린 여러 상황에 따라 다른 효과음이 함께 재생되니 그 부분 참고해서 들어주시면 감사하갰습니다. (0 0 입력)
-
-(번갈아 가며 행, 열 입력받고 입력하기)
솜솜이 : (2 0)
H : (1 1)
솜솜이 : (3 0)
H : (2 2)
솜솜이 : (4 0)
H : (3 3)
솜솜이 : (5 0)
H : (5 0)
방금과 같이 앞서 이미 입력된 자리에 입력한 경우 옳은 값을 입력하도록 다시 입력창이 나오는 것을 확인해보실 수 있습니다.
H : (4 4)

- 네 이렇게 승패가 결정되면 효과음이 나오고 이겼다는 문장이 앞서 print함수를 반복해서 만들었던 그림과 함께 출력되는 것을 볼 수 있습니다. (콘솔창 닫기)

<59>
지금까지 게임을 직접하며 코드를 실행해 보았고 마무리로 조원들과 게임 프로젝트를 진행하며 느낀점을 정리해보았습니다. 대부분의 조원들이 수업 중 배웠던 코드를 이용해 게임을 만드는 과정에서 원인모를 오류를 해결하는데에 어려움을 겪었다고하였습니다. 하지만 더불어 알았던 코드뿐만 아니라 새로운 다른 코드들을 접할 수 있었고 이를 학우분들께 설명하기 위해 더 자세히 알아보고 공부할 수 있던 시간이었다고 했습니다.

<60>
이것으로 발표를 마무리하겠습니다. 감사합니다.