2021년 1학기

**프로그래밍과 문제해결**

Assignment #2

담당 교수 : 윤은영

학번 : 20210084

학과 : 무은재학부

이름 : 김지민

POVIS ID : kjm1672

================명예서약(Honor code)===============

“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”

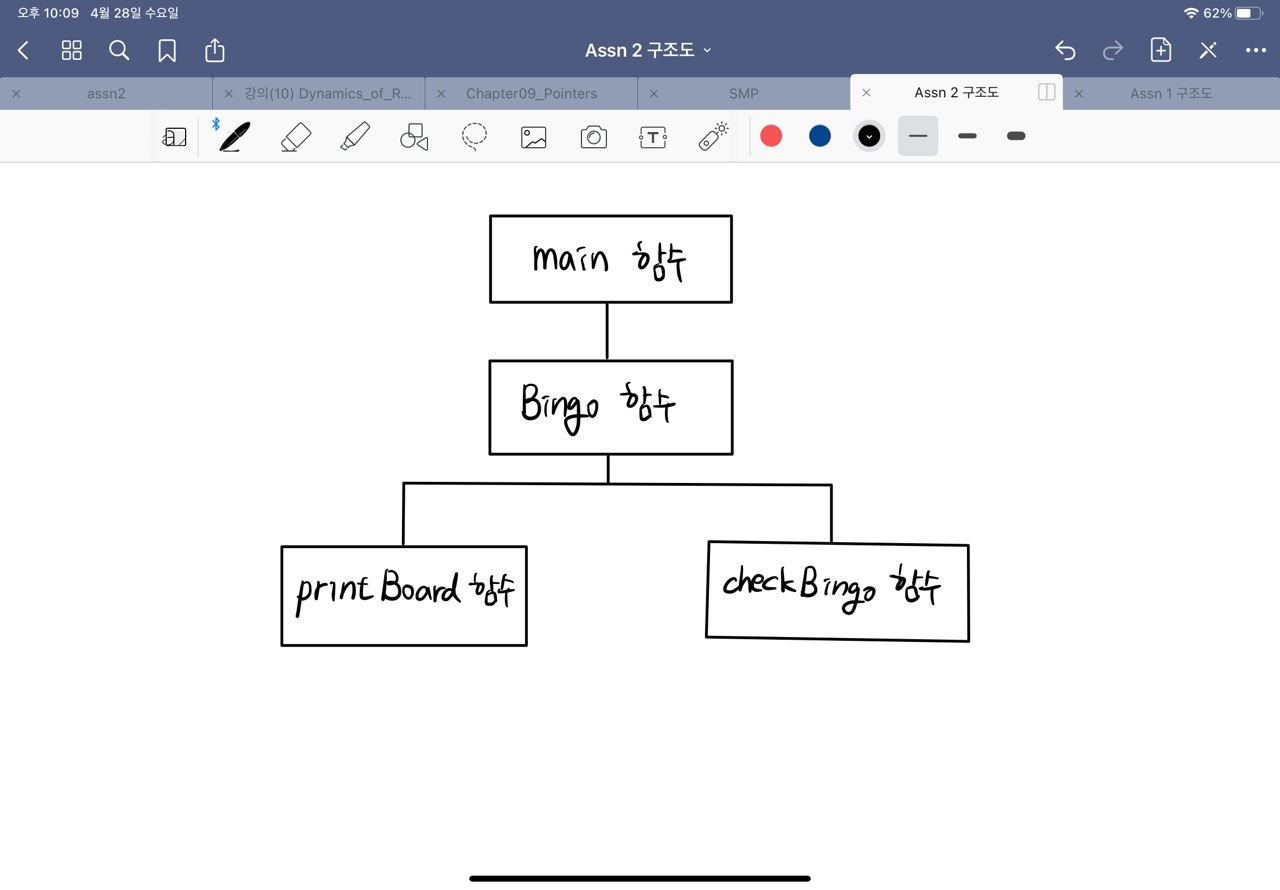
==============================================

1. 문제의 개요

본 프로그램을 간략히 소개하면 다음과 같다.

* N \* N 크기의 빙고 판을 구현하여 플레이어와 컴퓨터가 대결한다.
* 이때 사용자로부터 빙고 판 생성 방법(1 : 랜덤, 2: 파일 입력)을 입력 받는다.
* 랜덤 생성시 빙고 판의 크기(N)을 입력 받는다. (최소 3에서 최대 5)
* 알파벳은 빙고 판의 크기만큼만 사용하도록 한다. 예를 들어 3 \* 3 크기라면 A~I까지의 9개 문자만 사용하도록 한다.
* 플레이어와 컴퓨터가 번갈아 가면서 알파벳을 선택한다. 이때, 플레이어가 먼저 알파벳을 입력한다. 입력한 알파벳이 있는 칸은 # 으로 표시하도록 한다.
* 빙고 한 줄이라도 완성이 되면 게임은 종료된다.

이때 구조 차트 (structure chart)는 아래와 같이 표현될 수 있다.

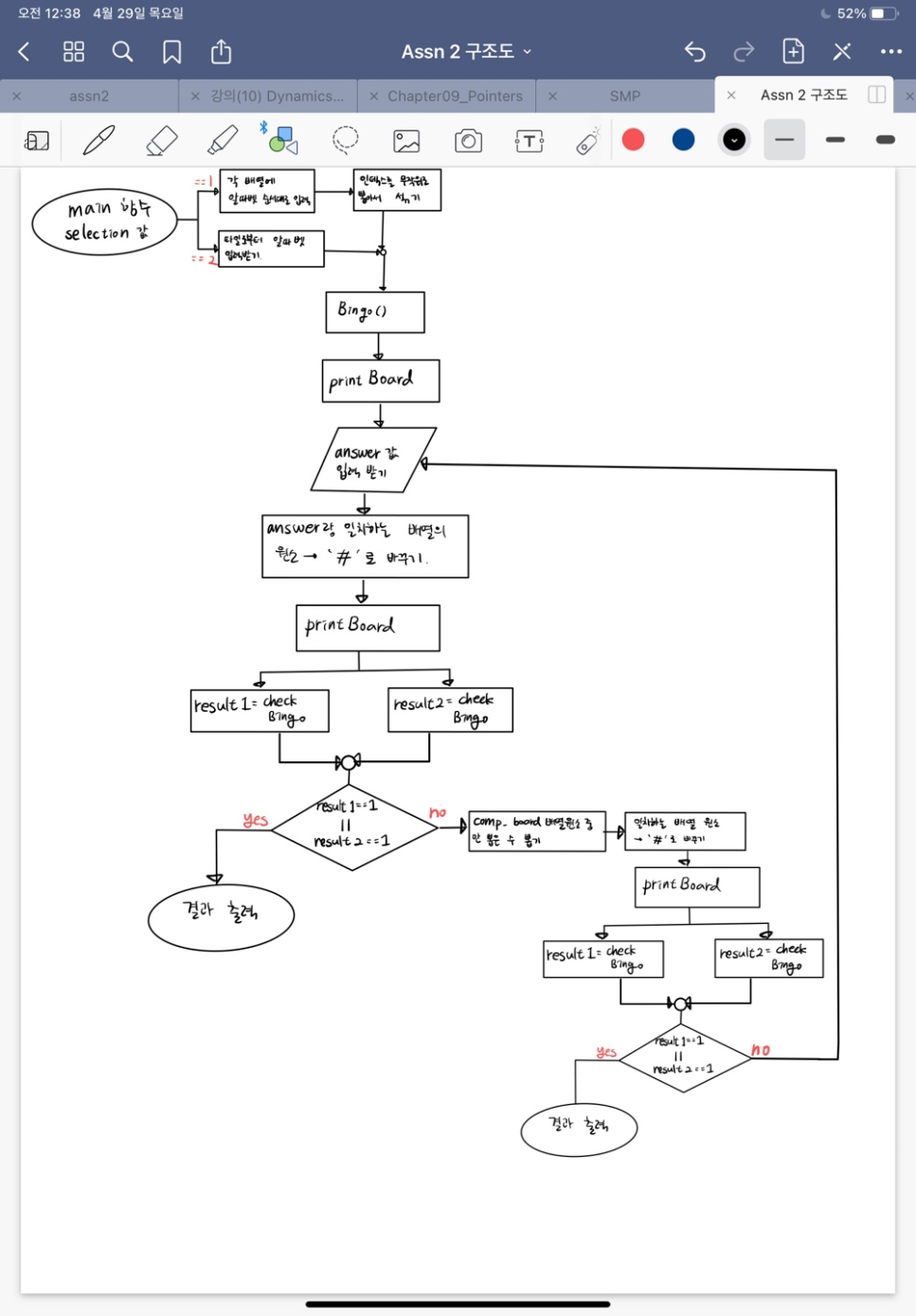


1. 알고리즘

본 프로그램 작성을 위한 알고리즘을 Presudo 코드 형태로 나타내면 다음과 같다.

|  |
| --- |
| Presudo-algorithm |
| 1. 헤더파일 <stdio.h>, <stdlib.h>, <time.h>, <Window.h>를 포함시킵니다. 2. 사용자로부터 빙고 판 생성 방법을 선택 받습니다.   - 1: 랜덤 생성 , 2: 파일 입력  - 1, 2 외 입력이 들어올 경우, 올바른 입력이 들어올 때까지 반복하여 입력 받습니다.  3. 선택된 생성 방법으로 빙고 판을 출력합니다. 이때 컴퓨터의 빙고 판 내용은 표시되지 않습니다.  -1(Random) : 알파벳은 앞에서부터 빙고 판 칸 수에 해당하는 개수만큼 사용할 수 있습니다. (3 \* 3 크기라면, A~I까지 9개의 문자만 사용할 수 있습니다.) 이때 주어진 알파벳은 모두 사용해야 합니다. (중복 X) 플레이어와 컴퓨터의 빙고 판을 만들기 위해 2차원 배열을 선언하여 사용합니다. rand()함수를 활용하여 매번 시작할 때마다 다른 빙고 판이 출력되도록 합니다.  - 2(Read) : 미리 만들어진 텍스트 파일인 board.txt 를 읽어서 빙고 판을 생성합니다. 플레이어와 컴퓨터의 빙고 판을 만들기 위해 2차원 배열을 선언하여 사용합니다.  4. 플레이어부터 시작하여 플레이어와 컴퓨터가 번갈아 가면서 알파벳을 입력합니다. 해당 문자는 #으로 표시됩니다. (플레이어, 컴퓨터 빙고 판 모두) 컴퓨터가 선택할 때는 1초의 지연 후 컴퓨터가 선택한 알파벳과 빙고 판을 출력합니다.  - for 문과 if 문을 이용하여 배열의 어느 원소에 해당 문자가 있는지 검사하고 그 원소의 값을 #으로 바꿉니다.  - 입력 받은 알파벳이 빙고 판의 알파벳 범위에서 벗어나거나 이미 입력 받은 값이면 다시 입력 받도록 합니다. 이때 1차원 배열을 사용하여 사용자로부터 입력 받은 값들을 저장할 수 있도록 합니다.  - <Window.h> 헤더파일을 포함시켜 지연 시간을 설정할 수 있도록 합니다.  5. Bingo 함수를 설정합니다. 이 함수에서는 매 입력마다 printBoard 함수를 호출하여 빙고 판의 현황을 알 수 있도록 합니다. checkBingo 함수도 호출함으로써 두 빙고 판에 완성된 줄이 있는지 검사합니다. 이 과정을 while 문을 이용하여 반복합니다.  - checkBingo 함수에서 for문과 if문을 활용하여 가로, 세로, 왼쪽 대각선, 오른쪽 대각선 줄을 모두 검사하도록 합니다. 이때, 만약 하나라도 있다면 1을 가지고 돌아가도록 합니다. (없다면 0)  - checkBingo 함수의 결과가 1이라면 반복문을 벗어나도록 합니다. 이때, 플레이어와 컴퓨터 빙고 판의 검사 결과를 구분하여 누가 승리하였는지 판별할 수 있도록 합니다.  6. 현재 빙고 판, 선택된 알파벳, 게임 결과를 result.txt라는 파일에 출력할 수 있도록 합니다. 파일 출력 또한 printBoard함수에서 구현합니다. |

위 알고리즘을 flowchart 로 표현하면 아래와 같다.



1. 프로그램 구조 및 설명
2. 초기 화면 출력.

* main 함수 : 빙고 판 생성 방법을 사용자로부터 입력 받는다. do while문을 활용하여 1 (Random) 또는 2(Read from File)를 입력 받을 때까지 반복한다. 입력 값은 selection 변수에 저장하고, if 문을 활용하여 selection값이 1인 경우와 아닌 경우를 구분한다.

1. 2차원 배열 랜덤 생성

* main 함수

1. do while문을 활용하여 빙고 판의 크기에 대해 3~5 범위의 값을 입력 받을 때까지 반복한다. 이 값은 num 변수에 저장한다.
2. user\_board[MAX\_SIZE][MAX\_SIZE], comp\_board[MAX\_SIZE][MAX\_SIZE]라는 2차원 배열을 선언하여 플레이어와 컴퓨터의 빙고 판 배열을 저장할 수 있도록 한다.
3. 이중 for문을 이용하여 각 2차원 배열에 빙고 판의 크기만큼 알파벳을 순서대로 입력한다.
4. index1, index2 변수를 선언한 다음 또 다른 이중 for 문을 만든 뒤, 각 변수에 rand()함수를 활용하여 0~num-1 사이의 값을 무작위로 저장한다. [index 1][index2]를 인덱스로 가지는 원소를 앞에서부터 차례대로 바꾸는 방식으로 각 배열의 원소의 순서를 바꾼다.

즉,

for (i = 0; i < num; i++)

for (j = 0; j < num; j++)

{

index1 = rand() % num + 0;

index2 = rand() % num + 0;

temp = user\_board[i][j];

user\_board[i][j] = user\_board[index1][index2];

user\_board[index1][index2] = temp;

}

이렇게 이중 for문과 rand()함수를 활용하여 인덱스를 무작위로 뽑은 다음 섞는 방식으로 배열의 원소 내용을 랜덤하게 바꿀 수 있다. comp\_board 배열에도 똑같은 방식을 적용한다.

1. 2차원 배열 파일 입력
2. indata라는 파일 포인터 변수를 선언한 뒤, board.txt 파일을 읽기 모드로 연다.
3. 파일 속 첫 번째 숫자는 빙고 판의 크기에 해당한다. 이를 fscanf 함수로 num 변수에 저장한다.
4. 플레이어 빙고 판이 그 다음 줄부터 있으므로 이중 for문을 이용하여 user\_board 2차원 배열에 차례대로 저장한다. 이때, 공백 (스페이스, 엔터, 탭)은 무시해야 하므로 do while 문을 이용하여 공백이 아닌 문자를 입력 받을 때까지 반복한다. 똑같은 방식으로 comp\_board 2차원 배열에도 문자를 저장한다.
5. 빙고 출력

* void printBoard: main함수에서 입력된 플레이어, 컴퓨터의 빙고 배열과 빙고 판의 크기를 매개변수로 입력 받는다. (편의상, 변수 명은 main함수하고 통일하였다.) 이중 for문을 활용하여 ‘+-‘으로 구성된 줄을 빙고 판의 크기에 맞게 출력한다. 이때, 컴퓨터의 빙고 판의 알파벳은 출력되면 안되므로, 이중 for문과 if 문을 활용하여 배열 원소의 값이 ‘#’일 때만 출력하고, 아닌 경우엔 공백으로 출력하도록 한다.

1. 빙고 진행

* void Bingo

1. main함수에서 입력된 플레이어, 컴퓨터의 빙고 배열과 빙고 판의 크기를 매개변수로 입력 받는다. (편의상, 변수 명은 main함수하고 통일하였다.)
2. 사용자로부터 알파벳을 입력 받고 이를 answer 변수에 저장한다. do while문을 활용하여 1) 범위 밖 알파벳이 아닌지 2)이미 입력 받았던 알파벳이 아닌지 검사하고 하나라도 걸리면 다시 반복한다.
3. answer 값은 selected[20]라는 1차원 배열에 저장한다. 또한 answer값과 일치하는 원소가 없는지 플레이어, 컴퓨터의 배열 모두 검사한다. 일치하는 원소는 ‘#’로 값을 바꾼다.
4. 컴퓨터 입력 값은 현재 빙고 판에 남아있는 알파벳 중 무작위로 뽑아야 한다. index1과 index2 변수를 선언한 뒤 rand()함수를 활용하여 인덱스를 무작위로 뽑는다. 이때 do while문을 활용하여 이미 선언된 알파벳인지 검사한다. 다음과 같이 코드를 구성할 수 있다.

do

{

index1 = rand() % num;

index2 = rand() % num;

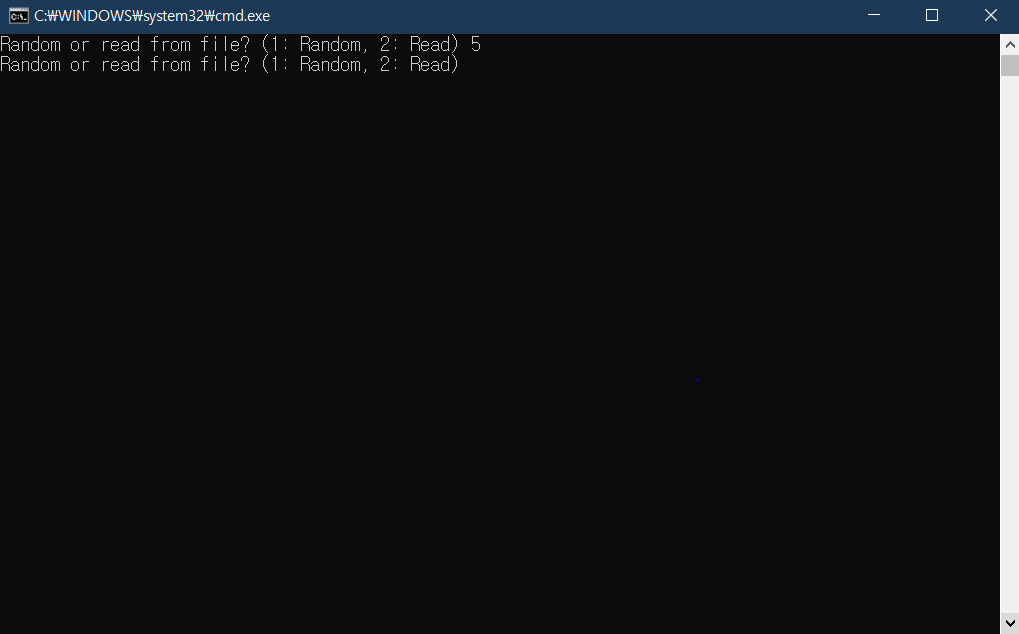
} while (comp\_board[index1][index2] == '#');

이 값과 일치하는 플레이어 배열의 원소 값과 해당 컴퓨터 배열 원소 값을 ’#’으로 바꾼다.

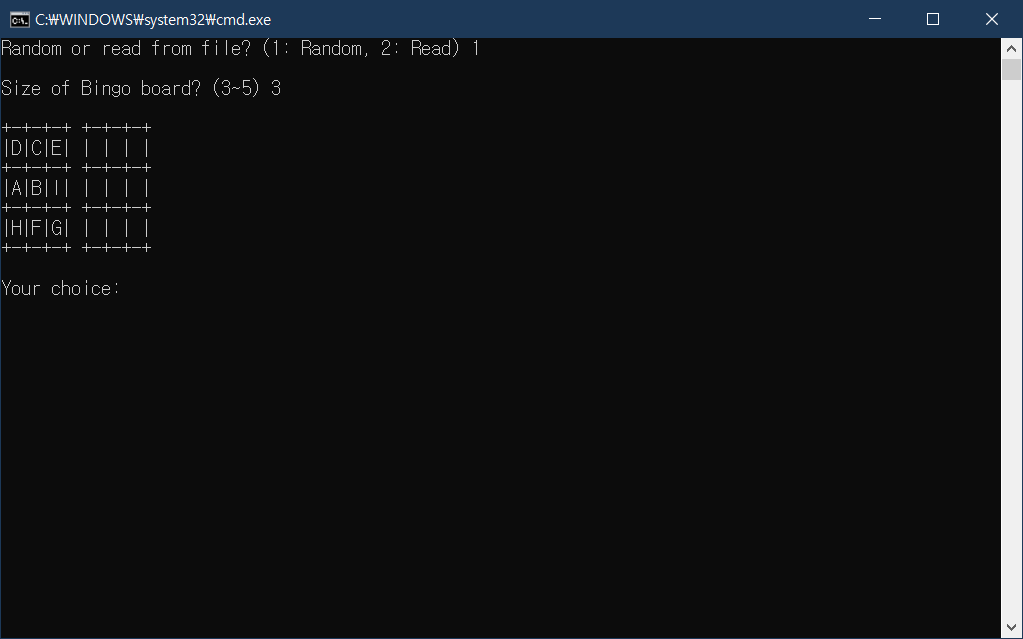
1. 플레이어와 컴퓨터의 입력이 끝날 때마다 printBoard 함수와 checkBingo 함수를 호출함으로써 빙고 판의 현황을 출력하고 완성된 줄이 있는지 검사한다. 이는 while문을 활용하여 계속 반복한다. 이때, 각 배열에 대하여 checkBingo 함수를 호출한 다음, 그 결과를 result1 (플레이어), result2 (컴퓨터) 변수에 각각 저장하는데, 하나라도 1이라면 while문을 탈출한다.
2. result1 ==1이면, “You Win!” 출력 / result ==2이면, “Computer Wins” 출력 / 둘 다 1이면 “Draw”출력.
3. 파일포인터 변수 output을 선언하고 result.txt를 w모드로 연다. 입력을 받을 때마다 fprintf 함수를 활용하여 매 입력 값과 빙고 판의 현황을 출력한다. 이때 잘못된 입력 값은 출력하지 않는다.

* int checkBingo: 2차원 배열 하나와 빙고 판 크기를 매개변수로 입력 받는다. 이중 for문과 if문을 활용하여 가로, 세로, 왼쪽 대각선, 오른쪽 대각선 줄을 모두 검사한다. result 변수를 선언하고 0으로 초기화한 후, 한 줄이라도 완성되었을 경우 result 값을 1로 변경한다. 최종적으로 result 값을 반환한다.

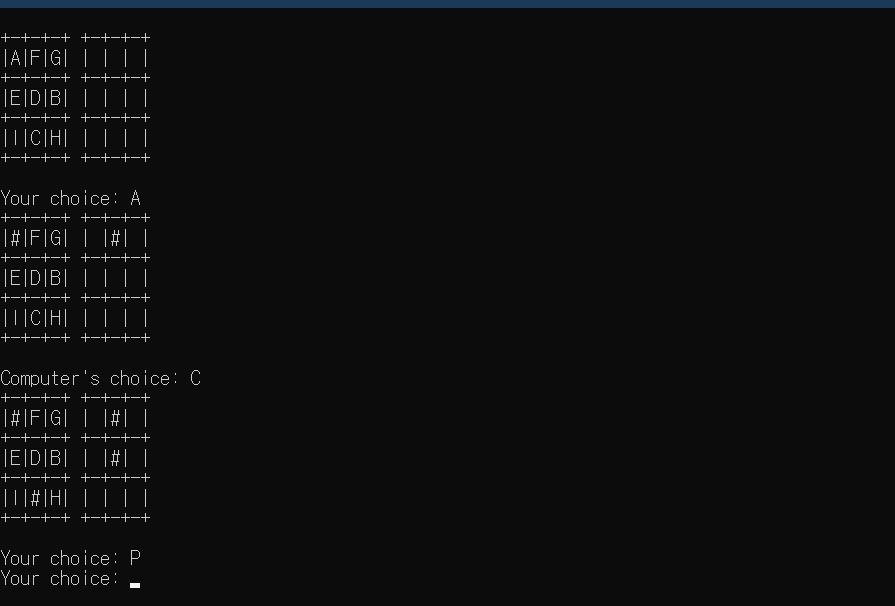
1. 프로그램 실행방법 및 예제

<초기 화면>

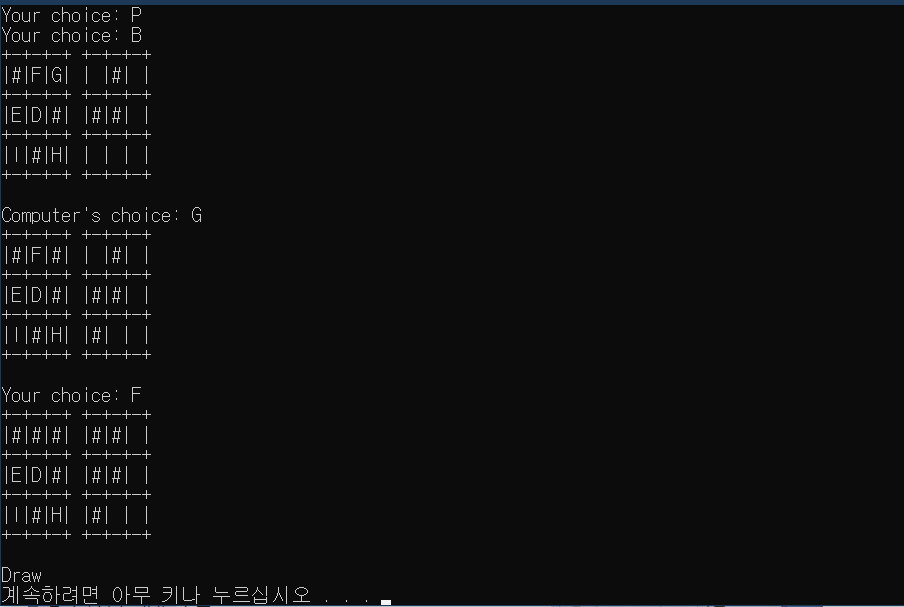
* 1, 2외 다른 숫자가 나오면 다시 입력 받기.

<빙고 판 출력>

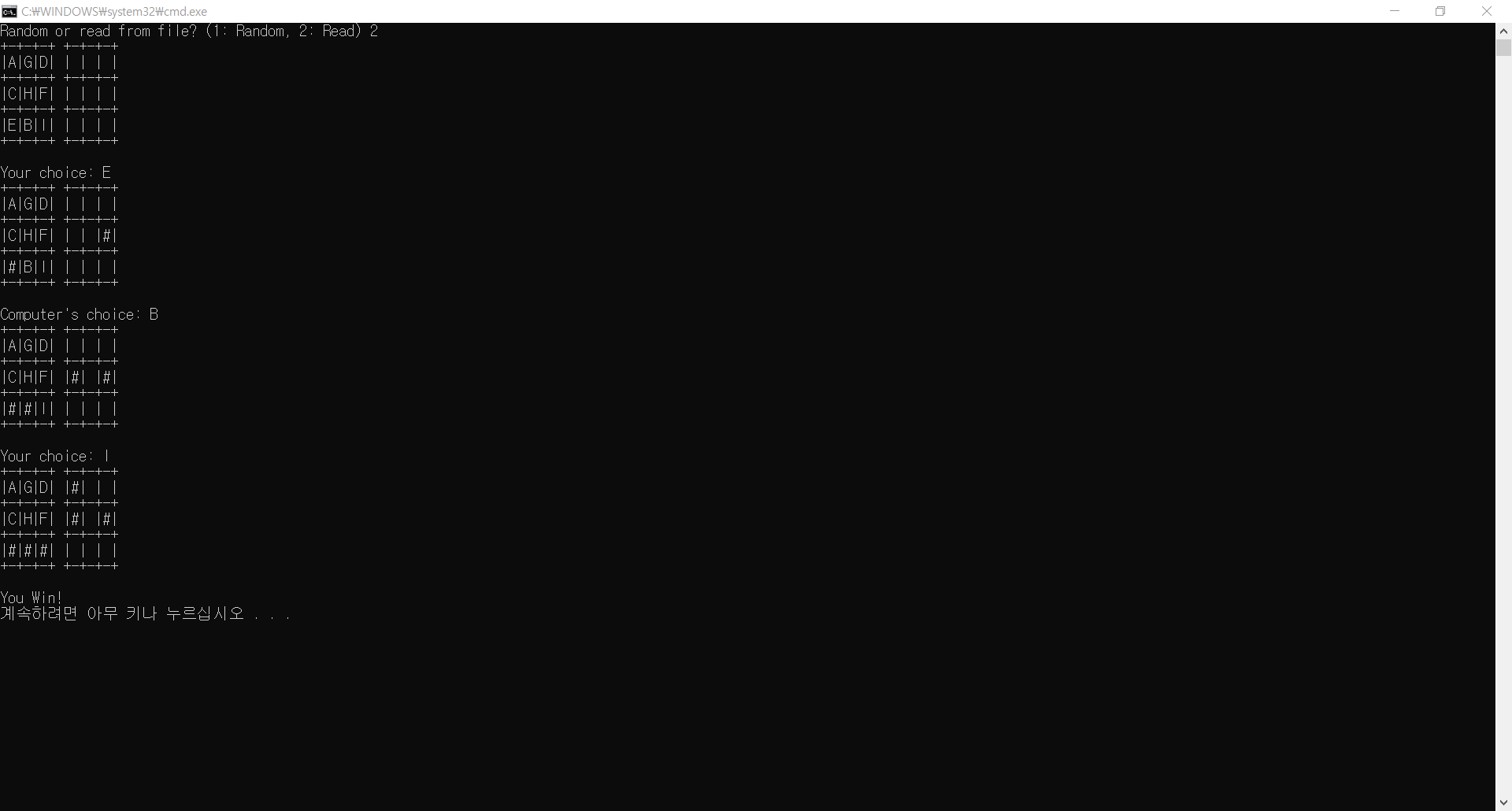
* 플레이어의 빙고 판 알파벳만 출력된다.
* 플레이어부터 알파벳을 입력한다.

<빙고 판 현황 및 예외 처리>

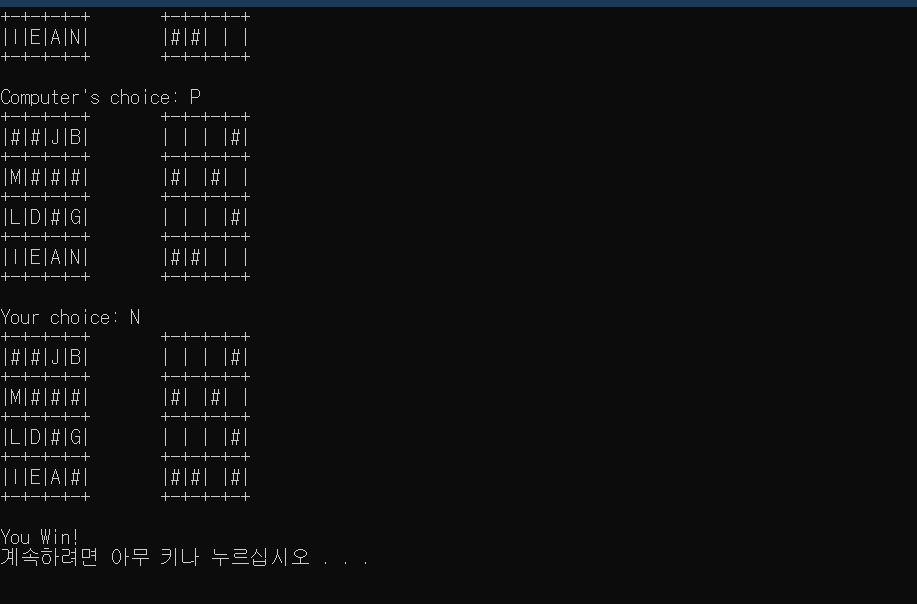
* 플레이어와 컴퓨터의 입력 값이 #으로 출력된다.
* 잘못 입력했을 경우 다시 입력 받는다.

<결과 출력>

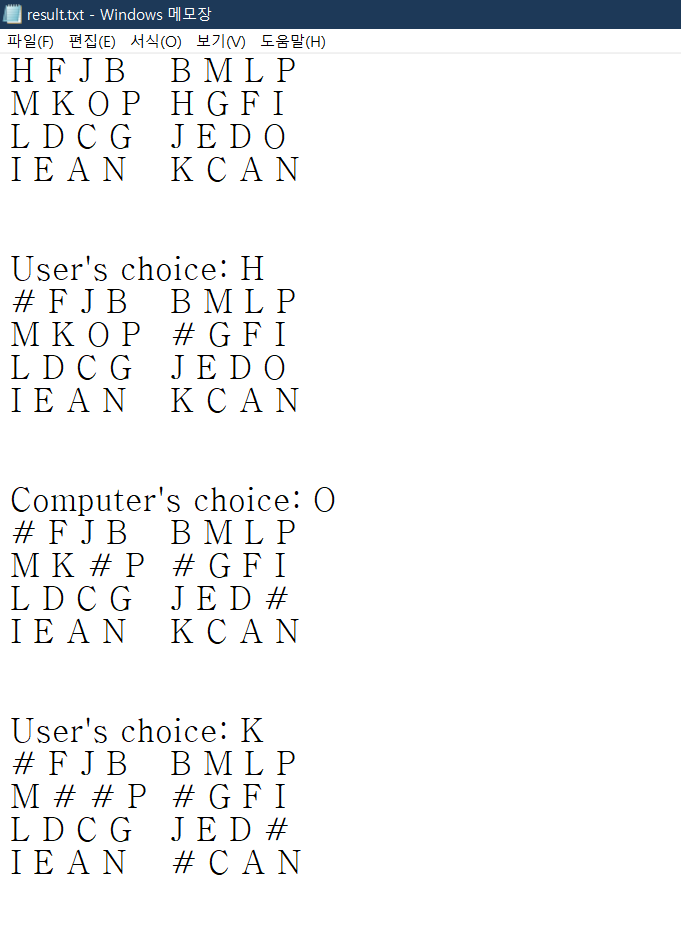
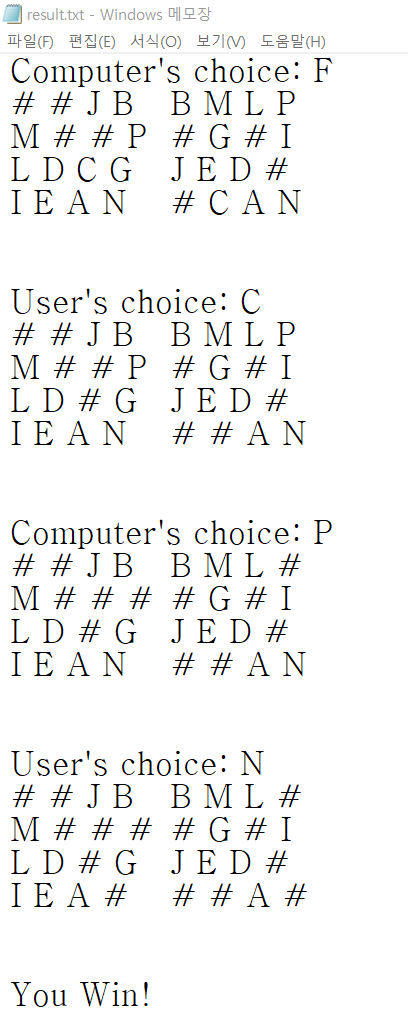
* 한 줄이라도 완성되면 결과를 출력하고 프로그램이 종료된다.
* 동시에 완성되었기 때문에 ‘Draw’가 출력된 것을 확인할 수 있다.

<결과 출력>

* 플레이어만 빙고가 완성되었기 때문에 ‘You Win!’메시지가 출력된 것을 확인할 수 있다.

<빙고 파일 출력>

* 4 \* 4 크기의 빙고 판이다. 다음 게임의 결과가 파일로 출력된 것을 밑에서 확인할 수 있다.



1. 토론

* rand()함수를 이용하여 배열 원소 하나하나에 난수를 생성하려 했으나, 중복 여부를 검사하는 과정에서 반복 횟수가 지나치게 많아져 너무 복잡해진다는 문제점이 발생하였다.
* 우선 배열에 알파벳을 차례대로 입력한 후, 인덱스를 섞는 방식을 이용함으로써 코드를 단순화할 수 있었다.

1. 결론

* 본 과제는 rand()함수부터 파일 입출력, 배열 등을 적절히 활용하는 것을 요구한다. 빙고 판을 무작위로 생성하는 과정에서 배열과 rand()의 기초를 다지고 적절하게 활용하는 방법을 익힐 수 있었다. 또한 결과를 파일로 출력함으로써 파일 입출력의 응용을 익힐 수 있었다.

1. 개선방향

* 본 과제에서 요구하는 사항들은 지금까지 배운 수업 내용을 적절히 활용하면 충분히 해결할 수 있었다. 다만 본인이 배열에 대한 기초가 부족하고 반복문의 응용에 미숙하기 때문에 프로그램 코딩을 하는 데에 있어서 오류가 많이 발생하였다. 이번 과제를 통해 배열의 기초를 다지고 응용을 수월하게 할 수 있을 것이라 기대한다. 배열과 반복문의 활용에 대해 실력을 키운다면 비슷한 프로그램을 좀 더 수월하게 코딩할 수 있을 것이다.