CSED101. Programming & Problem solving
Fall 2021

Programming Assignment #2
(75 points)

이명지 (mjlee7@postech.ac.kr)

■ **제출 마감일: 2021.11.16 23:59**

■ **개발 환경: Windows Visual Studio 2019**

■ **제출물**

- C 소스 코드 (assn2.c)
  - ➢ 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn2.docx, assn2.hwp 또는 assn2.pdf)
  - ➢ 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
  - ➢ 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
  - ➢ 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ **주의 사항**

- 컴파일이나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 "POSTECH 전자컴퓨터공학부 부정행위 정의"를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조하세요).
- 이번 과제는 추가 기능 구현과 관련된 추가 점수가 따로 없습니다.

■ Problem: 매치3 게임

(문제)
이번 과제에서는 2차원 배열을 이용하여 매치3 게임을 구현합니다. 매치3 게임의 대표적인 예시로, ‘캔디크러시사가’, ‘애니팡’이 있습니다.

(목적)
- 2차원 배열의 선언과 사용을 익힙니다.
- 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힙니다.
- 텍스트 파일을 통한 입력을 익힙니다.

(주의사항)
- 이번 과제는 2차원 배열을 선언하고 사용하는 과제이므로, 게임 보드는 반드시 2차원 배열로 선언 후 사용해야 합니다.
- 전역 변수, goto 문, string 관련 함수, 구조체는 사용할 수 없으며, 포인터의 경우 수업시간에 다룬 내용에 한해서 사용이 가능합니다.
- 모든 기능을 main() 함수 내에 구현 시 감점합니다. 기능적으로 독립됐거나 반복적으로 사용되는 기능은 사용자 정의 함수를 정의해서 구현하여야 합니다.
- 프로그램 구현 시, main() 함수를 호출하여 사용하지 않습니다. 즉, 소스 코드 내에 main(); 이라고 호출하지 않습니다.
- 문제의 출력 형식은 채점을 위해 실행 예시와 최대한 비슷하게 작성 바랍니다.
- 프로그램에서 랜덤 시드는 프로그램 시작 시 main() 에서 srand(time(NULL)); 함수를 한 번만 호출하도록 하여 한번만 초기화 합니다.
- 명시한 에러 외에는 고려할 필요가 없습니다.

(게임 규칙)
- 플레이어는 6×6 보드에서 제한 턴 수(5턴) 동안 같은 블록 3개 이상을 가로 세로로 맞춰 제거해야 합니다.
- 블록은 숫자 1~5로 구성되는 숫자 블록입니다.
- 상하좌우로 인접한 블록끼리만 교환 가능합니다.
- 없어진 블록에 해당하는 숫자만큼 점수를 획득하게 됩니다. 예를 들어, 숫자 블록 ‘2’를 3개를 제거했을 경우 6점을 획득하게 됩니다.
- 빈 자리는 임의의 숫자 블록으로 채워집니다.

(함수 요구 사항)
반드시 작성해야 하는 함수는 다음과 같습니다. 이 함수들 외에 필요한 함수들을 적절하게 구현하도록 하세요. 아래 명시된 함수 이름은 변경하지 말아주세요. (매개변수는 자유롭게 구성하여 구현하면 됩니다.)
- **main()**
  게임 보드를 나타내는 적절한 2차원 배열을 main() 함수에 선언 후 사용
- **print_board()**
  게임 보드(2 차원 배열) 등을 전달받아 점수, 보드, 보드의 행과 열 번호 출력하는 함수
- **check_position()**
  사용자 입력한 두 블록의 위치가 유효한지 검사하는 함수

(설명)

(1) **[게임 시작]**

프로그램을 실행하면 아래와 같이 게임 시작 화면을 출력합니다.

```
-=-=-=-=-=-=-=-=-=-=-=
CSED 101
Assignment 2

[[ Match-3 Game ]]

Press [Enter] to start
-=-=-=-=-=-=-=-=-=-=-=
```

(2) 시작 화면에서 [Enter]키가 입력되면 시작 화면을 지우고(문서 말미 "Tip-(a)" 참고) 아래와 같이 현재 점수, 6×6 게임 보드, 보드의 행과 열 번호를 출력합니다. 이 때, [Enter]키 입력 외에는 고려할 필요가 없습니다.

점수의 초기값은 0으로 설정하고, 보드는 'board.txt'로부터 읽어온 것으로 출력하며, 행과 열은 좌측상단부터 0부터 시작하도록 합니다. 게임 보드는 반드시 2차원 배열로 선언 후 사용해야 합니다.

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 5 3 2 ¦ 3
1 1 5 4 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

board.txt 내용

```
4 2 3 4 4 2
1 3 4 3 4 1
1 4 3 2 1 2
5 4 2 5 3 2
1 1 5 4 5 5
3 3 2 4 1 5
```

예외처리)

'board.txt' 파일이 없는 경우, 적절한 에러 메시지 출력 후 프로그램을 종료합니다.

'board.txt' 파일 내용이 틀린(게임 보드를 채울 숫자가 부족하거나 또는 많은 등) 경우에 대해서는 고려하지 않습니다.

(3) 남은 턴 수를 출력하고 사용자 입력을 기다립니다. 남은 턴 수의 초기값은 5 입니다.

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 5 3 2 ¦ 3
1 1 5 4 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================

* Remaining turns: 5
Swap...
Block 1:
```

(4) 사용자는 교환할 두 블록의 위치 값을 각각 차례로 입력합니다. 위치 선택은 행, 열의 순으로 입력합니다. 즉 4행 3열의 위치를 선택하고 싶으면 4 3 의 순서로 입력합니다.
(실행 예시의 밑줄은 사용자 입력을 나타냄)

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 5 3 2 ¦ 3
1 1 5 4 5 5 ¦ 4   <-- 4행
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================
      ^
      3열
* Remaining turns: 5
Swap...
Block 1: 4 3
Block 2: 3 3
```

(5) 유효한 입력인지 확인합니다. 유효한 입력이 아니라면 에러 메시지를 출력하고 다시 입력을 받습니다. 유효 입력 확인은 Block 1과 Block 2를 모두 입력 받은 후에 진행하도록 합니다.

가. 아래 왼쪽 예시는 보드의 행과 열 범위를 벗어난 경우의 출력 예시입니다.
입력 예) Block 1: 6 0, Block 2: 5 0

나. 아래 오른쪽 예시는 인접한 두 블록이 아닌 경우(입력한 두 위치가 같은 경우 포함)의 출력 예시입니다.

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 5 3 2 ¦ 3
1 1 5 4 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================

*** Out of range! ***
* Remaining turns: 5
Swap...
Block 1:
```

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 5 3 2 ¦ 3
1 1 5 4 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================

*** They are not adjacent! ***
* Remaining turns: 5
Swap...
Block 1:
```

(6) 유효한 입력인 경우라면, 화면을 지운 후 입력 받은 두 블록을 교환한 후 보드를 출력합니다. 아래는 "Block 1: 4 3, Block 2: 3 3"을 입력 받아 교환한 예시입니다. 쉽게 확인할 수 있도록 교환한 블록은 빨간 색깔로 처리하고(문서 말미 "Tip-(b)" 참고) 잠시 현재 출력 상태를 1초간 유지합니다(문서 말미 "Tip-(c)" 참고).

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 4 3 2 ¦ 3
1 1 5 5 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

(7) **[매치3 검사]**

게임 보드에서 가로 또는 세로로 연속된 3개 이상의 블록이 같은지 확인하고, 다음과 같이 해당 블록들을 빨간 색깔로 처리하여 보드를 출력합니다.

```
Score: 0
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 4 3 2 ¦ 3
1 1 5 5 5 5 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

(8) 매치된 블록들을 제거한 보드와 업데이트된 점수를 출력합니다.

```
Score: 20
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 4 3 2 ¦ 3
1 1         ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

(9) 빈 자리를 임의의 숫자(1~5) 블록으로 채웁니다.

```
Score: 20
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 4 3 2 ¦ 3
1 1 2 4 4 4 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

(10) 만약 채워진 보드에 또 연속된 블록이 3개 이상 존재한다면 (6)~(8)에서처럼 그 블록들을 색깔 처리 후 제거하고 점수를 업데이트 후 빈 자리를 채웁니다. 3개 이상 연속된 블록이 없

을 때까지 매치3 색깔 처리, 제거, 점수 업데이트, 보드 채우기를 반복합니다. 일련의 과정을 눈으로 확인할 수 있도록 화면을 지운 후, 다음 상태를 출력하기 전에 잠시 현재 출력 상태를 1초간 유지합니다.

```
Score: 20
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 2 4 3 2 ¦ 3
1 1 2 4 4 4 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

```
Score: 46
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4     3 2 ¦ 3
1 1         ¦ 4
3 3     1 5 ¦ 5
------------+
0 1 2 3 4 5
```

```
Score: 46
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 1 5 3 2 ¦ 3
1 1 1 3 4 5 ¦ 4
3 3 4 2 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

```
Score: 46
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 1 5 3 2 ¦ 3
1 1 1 3 4 5 ¦ 4
3 3 4 2 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

```
Score: 49
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 1 5 3 2 ¦ 3
      3 4 5 ¦ 4
3 3 4 2 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

```
Score: 49
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 1 5 3 2 ¦ 3
2 5 2 3 4 5 ¦ 4
3 3 4 2 1 5 ¦ 5
------------+
0 1 2 3 4 5
```

(11) 연속된 블록이 없다면, 다음 턴이 진행되며 (3)에서처럼 사용자 입력을 기다립니다.

```
Score: 49
4 2 3 4 4 2 ¦ 0
1 3 4 3 4 1 ¦ 1
1 4 3 2 1 2 ¦ 2
5 4 1 5 3 2 ¦ 3
2 5 2 3 4 5 ¦ 4
3 3 4 2 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================

* Remaining turns: 4
Swap...
Block 1:
```

(12) 남은 턴 수가 0이 될 때까지 (3)~(11)을 반복합니다.

(13) **[게임 종료]**

남은 턴 수가 0이 되면 다음과 같이 점수, 보드, 최종 점수를 알리는 메시지를 출력하고 프로그램을 종료합니다.

```
Score: 108
4 3 2 1 5 1 ¦ 0
1 2 1 5 4 2 ¦ 1
1 4 3 3 2 1 ¦ 2
5 4 2 2 1 2 ¦ 3
1 1 2 4 3 2 ¦ 4
3 3 2 4 1 5 ¦ 5
------------+
0 1 2 3 4 5
====================

** You've earned 108 points! **

계속하려면 아무 키나 누르십시오 . . .
```

[Tip]

(a) **화면 지우기**

system("cls") 함수를 호출하여 콘솔 화면에 출력된 내용을 지울 수 있습니다. 이를 이용하여 콘솔창에 있는 출력 화면을 지우고 다른 화면을 출력하는 작업을 반복할 수 있습니다.

```c
#include <stdio.h>
#include <stdlib.h> // system() 사용을 위해 포함

int main()
{
    printf("이 메시지는 곧 지워집니다.\n");
    system("cls"); // 리눅스 또는 맥의 경우 system("clear");
    return 0;
}
```

(b) **글자 색상 설정**

ANSI escape sequence("\033")와 색상 코드를 조합한 텍스트를 출력함으로써 콘솔의 출력 글자 색상을 설정할 수 있습니다. 아래는 사용 예시입니다. (맥에서 사용 가능합니다.)

```c
#include <stdio.h>

void set_color_red() {
    printf("\033[1;31m");
}

void reset_color() {
    printf("\033[0m");
}

int main() {
    set_color_red();
    printf("Hello ");
    reset_color();
    printf("world\n");
    return 0;
}
```

참고) 윈도우에서 위 코드로 안 되는 경우

```c
#include <stdio.h>
#include <windows.h> // SetConsoleTextAttribute() 사용을 위해 포함

void set_color_red() {
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 12); // RED
}

void reset_color() {
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 7); // GRAY
}
```

```c
int main() {
    set_color_red();
    printf("Hello ");
    reset_color();
    printf("world\n");
    return 0;
}
```

(c) **지연 효과**

지연 효과는 Sleep() 함수를 통해 구현 가능합니다. 해당 함수는 일정 시간 동안 작업을 대기시키며, 인자로 지연시간을 millisecond 단위로 받습니다. 사용하기 위해서는 <windows.h> 헤더파일을 포함해야 합니다. 아래는 사용 예시입니다.

```c
#include <stdio.h>
#include <windows.h> // Sleep() 사용을 위해 포함

int main()
{
    printf("첫번째 메시지입니다.\n");
    Sleep(1000);
    printf("이 메시지는 1초 뒤에 출력됩니다.\n");
    return 0;
}
```

참고) 리눅스 또는 맥에서 프로그래밍을 하는 경우

```c
#include <stdio.h>
#include <unistd.h> // sleep() 사용을 위해 포함

int main()
{
    printf("첫번째 메시지입니다.\n");
    sleep(1); // 매개변수는 초 단위
    printf("이 메시지는 1초 뒤에 출력됩니다.\n");
    return 0;
}
```

(d) 프로그램 작성 중 무한 루프에 의해 프로그램이 끝나지 않을 경우, [Ctrl]+[c]를 누르면 강제 종료됩니다.

(e) 다음은 과제 작성을 위해 구현하면 좋을 사항입니다. 필수 사항은 아닙니다.

- #define 지시자를 이용하여 보드 크기를 정의
- int match_block[BOARD_SIZE][BOARD_SIZE]: 매치 3 에 해당하는 블록을 체크한 변수