4190.103A-001: Programming Practice Lab. 14

Euidong Lee
Junseok Lee
Minhyo Jung
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

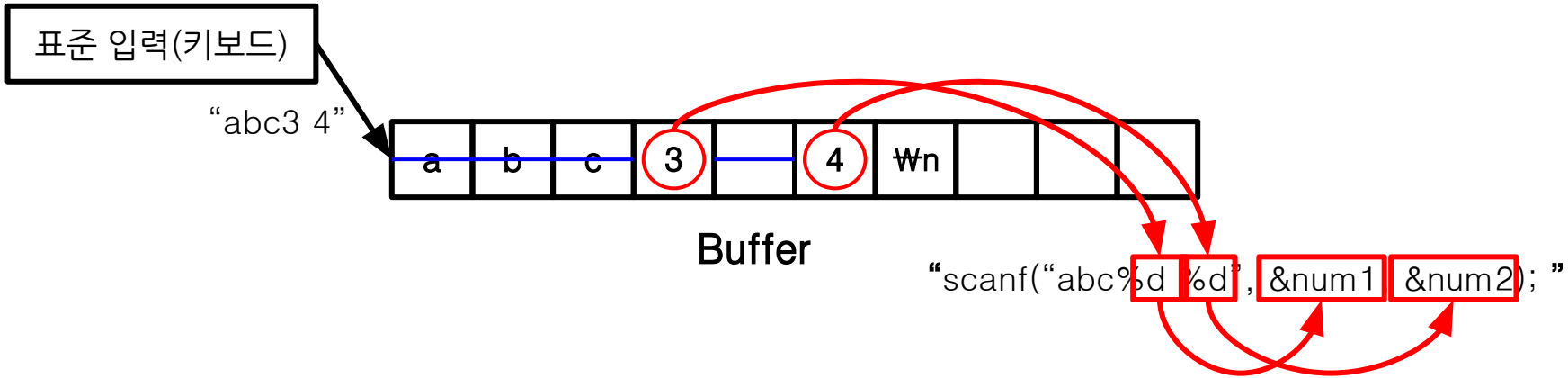
Seoul National University

Spring 2022

실습 1

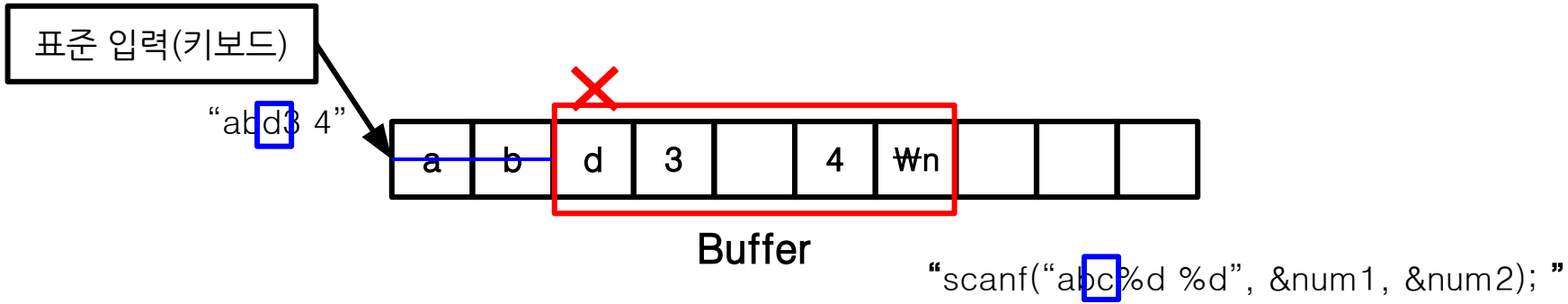
scanf

- scanf(“abc%d %d”, &num1, &num2);
  - 서식지정자 이외의 문자들은 비교 후 버림
  - 다르면 함수 종료
  - 서식지정자와 대응되는 값은 알맞는 type으로 변환 후 변수에 저장



scanf

- scanf 주의점
  - 버퍼에 데이터가 남는 경우가 존재(예: 입력의 형태가 달라 scanf 함수가 종료된 경우)
  - fflush, getchar와 같은 함수로 버퍼를 비울 수 있음



실습 2

File IO

- 실습을 통해 file I/O를 연습해봅시다!
- File I/O는 다음과 같은 과정을 통해 동작됩니다

File IO

- File I/O와 관련된 함수는 여러가지가 있습니다
  - fputc / fgetc
  - fputs / fgets
  - fprintf / fscanf
- printf / scanf와 사용법이 비슷한 fprintf / fscanf를 이용해봅시다

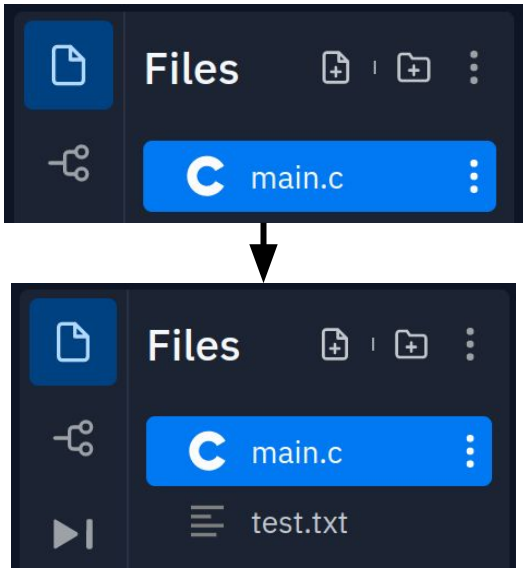
File IO

- 먼저, fopen을 이용해서 test.txt 파일을 만들어봅시다
  - 새로운 내용을 적기 위해서 “w”(write) 모드로 열어줍니다
  - 만약, 해당 파일이 없다면, 해당 이름의 새로운 파일을 만들어 줍니다
  - 파일이 있다면, 내용을 전부 비우고 새롭게 작성합니다

File IO

```c
#include <stdio.h>

int main(){
  FILE *fp;

  fp = fopen("test.txt", "w");

  if(!fp){
    printf("File open faild\n");
    return -1;
  }
  printf("File open success\n");

  fclose(fp);

  return 0;
}
```

```
make -s
./main
File open success
```



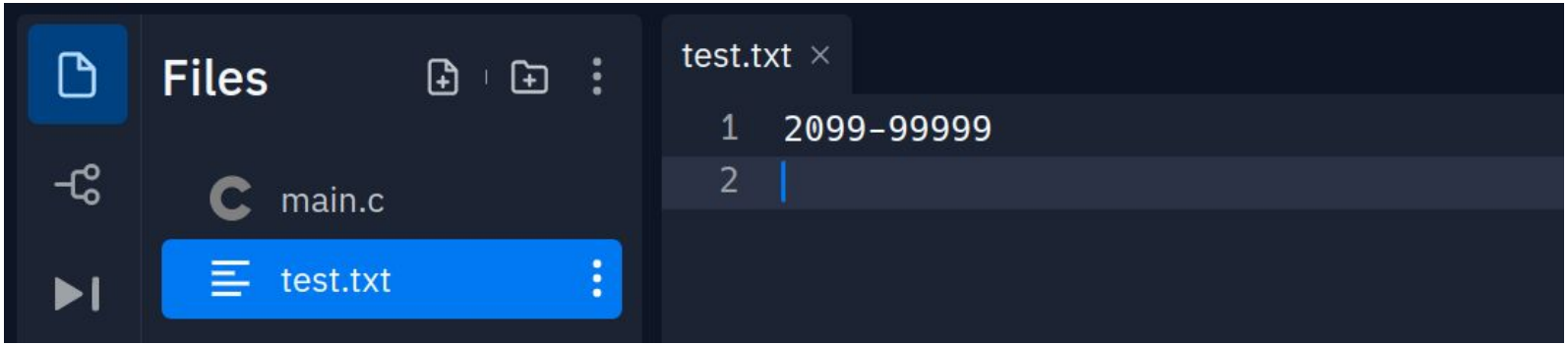
File IO

- test.txt 파일에 내용을 작성해봅시다
- fprintf는 printf와 사용법이 매우 유사합니다
- 차이점은 fprintf의 첫 인자로 File 포인터를 넘겨주어야 합니다

```
fprintf(fp, “hello, %d ₩n”, num);
```

File IO

- fprintf를 이용해서 test.txt 파일에 본인의 학번을 작성해보세요!



File IO

- 코드 예시

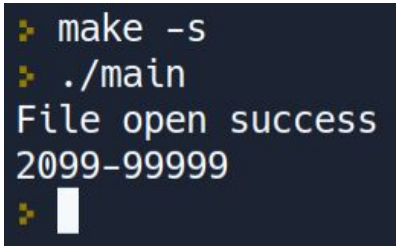
```c
#include <stdio.h>

int main(){
  FILE *fp;

  fp = fopen("test.txt", "w");

  if(!fp){
    printf("File open faild\n");
    return -1;
  }
  printf("File open success\n");

  fprintf(fp, "2099-99999\n");

  fclose(fp);

  return 0;
}
```

File IO

- 파일에 적은 내용을 읽어봅시다
- fscanf도 scanf와 사용법이 매우 유사합니다
- fprintf와 마찬가지로, fscanf를 사용하기 위해서는 File 포인터를 넘겨주어야 합니다

File IO

- 문자열을 받아올 char 배열을 만들어 줍니다
- test.txt 파일을 “r”(read) 모드로 열어줍니다
- 미리 만들어둔 char 배열에 파일의 문자열을 읽어옵니다
- char 배열에 담겨있는 내용을 printf를 이용해서 출력해보세요!



File IO

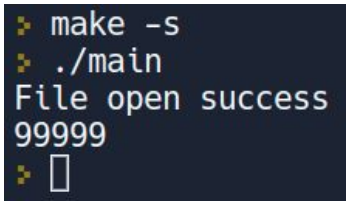
- 코드 예시

```c
#include <stdio.h>

int main(){
  FILE *fp;
  char result[100];

  fp = fopen("test.txt", "r");

  if(!fp){
    printf("File open faild\n");
    return -1;
  }
  printf("File open success\n");

  fscanf(fp, "%s", result);

  printf("%s\n", result);

  fclose(fp);

  return 0;
}
```

실습 3

File IO

- 학번의 앞자리(년도)를 제외하고, 뒷자리만을 정수로 읽어봅시다
  - 다양한 방법을 사용할 수 있습니다
- 먼저, fscanf로 읽어봅시다
  - scanf를 사용할 때와 비슷한 방법을 활용해봅시다



File IO

- 코드 예시

```c
int main(){
  FILE *fp;
  int _, num;

  fp = fopen("test.txt", "r");

  if(!fp){
    printf("File open faild\n");
    return -1;
  }
  printf("File open success\n");

  fscanf(fp, "%d-%d", &_, &num);

  printf("%d\n", num);

  fclose(fp);

  return 0;
}
```

File IO

- fseek 함수를 사용해봅시다
  - 파일 포인터의 위치를 조정할 수 있습니다

기준에서 얼만큼 이동할 것인지

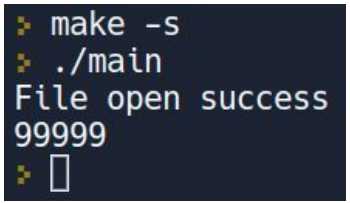
기준을 어디로 할 것 인가

```
fseek(fp, 10, SEEK_SET);
```

SEEK_SET(처음)
SEEK_CUR(현재 위치)
SEEK_END(끝)

File IO

- fseek을 이용해서 파일 포인터의 위치를 학번 뒷자리로 이동합니다
- 이후 fscanf를 이용해서 5자리를 int형으로 읽고, 읽은 내용을 출력해보세요



File IO

- 코드 예시

```c
int main(){
  FILE *fp;
  int num;

  fp = fopen("test.txt", "r");

  if(!fp){
    printf("File open faild\n");
    return -1;
  }
  printf("File open success\n");

  if(fseek(fp, 5, SEEK_SET)){
    printf("fseek failed\n");
    return -1;
  }

  fscanf(fp, "%d", &num);

  printf("%d\n", num);

  fclose(fp);

  return 0;
}
```

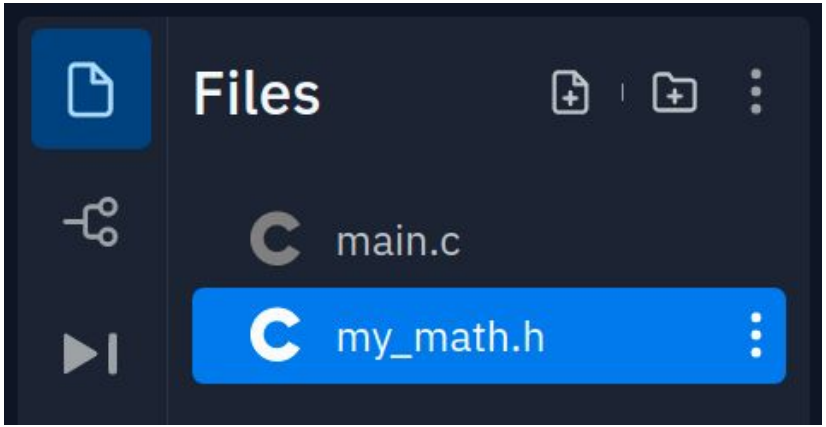
실습 4

Header File

- 헤더 파일이란?
  - .h 확장자 파일
  - 여러 소스 파일에서 공유되는 함수들, macro 등을 정의
- 예제를 통해서 이해해봅시다!
  - 직접 헤더 파일을 만들어 이용해봅시다

Header File

- 간단한 수학 계산 기능을 만들어 사용해봅시다
- 헤더 파일을 추가합니다
  - my_math.h 파일을 추가해보세요!



Header File

- 헤더 파일에는 헤더 가드가 필요합니다
  - 헤더 파일은 다수의 소스 파일에 include 되기 때문에, 별도의 조치가 없다면 중복해서 복사됩니다
  - 헤더 가드를 통해 이러한 경우를 방지할 수 있습니다

```
1  #ifndef MY_MATH_H
2  #define MY_MATH_H
3
4
5
6  #endif
```

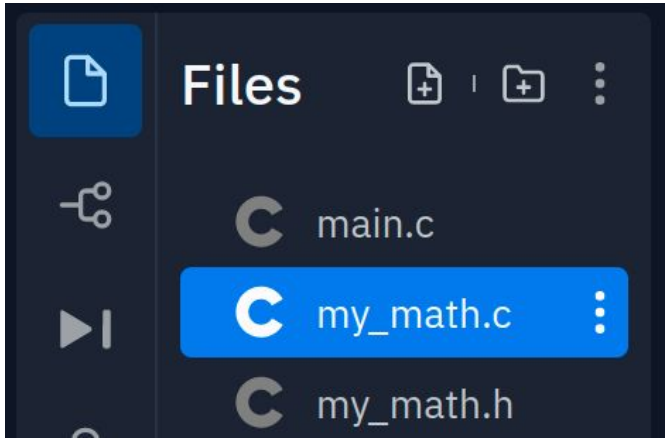
Header File

- 두 정수를 받아 덧셈을 계산해주는 my_add 함수를 만들어봅시다!
  - 헤더 파일에는 함수의 원형만 적습니다 (함수 정의)
  - 함수의 구현은 다른 소스 파일에서 이루어집니다

```
1  #ifndef MY_MATH_H
2  #define MY_MATH_H
3
4  int my_add(int num1, int num2);
5  //int my_add(int, int);
6
7  #endif
```

Header File

- 함수의 구현을 위해서 새로운 파일을 추가해봅시다
  - my_math.c 파일을 추가해보세요!



Header File

- my_math.c 파일에서 my_add 함수를 구현해봅시다
  - my_math.h 파일을 include 해야합니다
  - <>가 아닌, “”를 이용해야 합니다!

```c
#include "my_math.h"

int my_add(int num1, int num2){
    return num1 + num2;
}
```

Header File

- 이제 main.c 파일 안의 main 함수에서 my_add 함수를 사용해봅시다
  - 마찬가지로, my_math.h 파일을 include 해야합니다

```c
#include "my_math.h"

#include <stdio.h>

int main(){
    printf("%d\n", my_add(3, 7));

    return 0;
}
```

```
> make -s
> ./main
10
>
```

Header File

- 두 정수를 받아서 곱한 값을 구해주는 my_mul 함수를 만들어보세요!

Header File

my_math.h

```h
#ifndef MY_MATH_H
#define MY_MATH_H

int my_add(int num1, int num2);
//int my_add(int, int);

int my_mul(int num1, int num2);

#endif
```

my_math.c

```c
#include "my_math.h"

int my_add(int num1, int num2){
    return num1 + num2;
}

int my_mul(int num1, int num2){
    return num1 * num2;
}
```

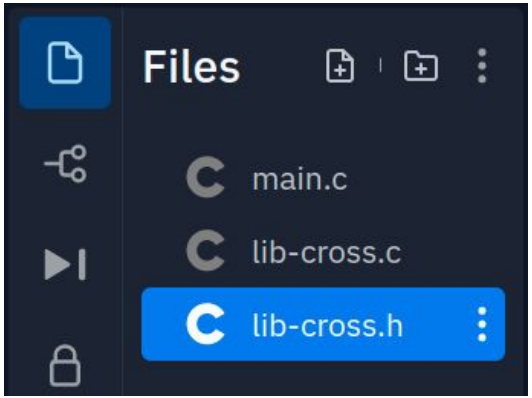
실습 5

Visualization

- Visualization을 위해서 다음을 위한 연습을 해봅시다!
  - 반짝이는 별을 그리는 프로그램을 만들어봅시다

Visualization

- 간단한 라이브러리를 이용해봅시다
  - 프로젝트에 lib-cross.c와 lib-cross.h 파일이 추가되어 있습니다
  - lib-cross.h 파일을 통해 사용할 수 있는 함수들을 찾아보실 수 있습니다



Visualization

- 한 단계씩 진행하면서 라이브러리를 연습해봅시다
  - 원하는 위치에 '*' 문자를 출력해봅시다

Visualization

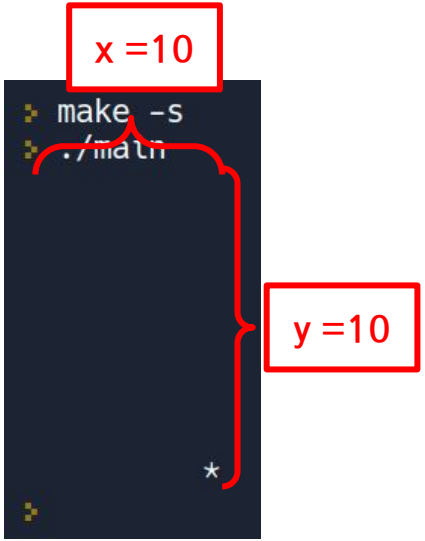
- 라이브러리를 include 해줍니다

```c
#include "lib-cross.h"

#include <stdio.h>

int main(){
    gotoxy(10, 10);
    printf("*");
    printf("\n");

    return 0;
}
```

Visualization

- 출력하고 싶은 위치로 커서를 이동시킵니다
  - 원래 위치에서 (x, y)의 위치
  - x : 10
  - y : 10

```c
#include "lib-cross.h"

#include <stdio.h>

int main(){
    gotoxy(10, 10);
    printf("*");
    printf("\n");

    return 0;
}
```



Visualization

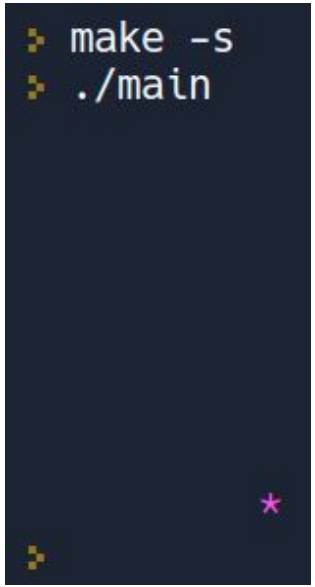
- ‘*’을 출력합니다

```c
#include "lib-cross.h"

#include <stdio.h>

int main(){
    gotoxy(10, 10);
    printf("*");
    printf("\n");

    return 0;
}
```

```
> make -s
> ./main


          *
>
```

Visualization

- 문자의 색을 변경해봅시다



Visualization

- 사용하고 싶은 색을 정해줍니다
  - 사용할 수 있는 색의 리스트는 lib-cross.h 파일에 정리되어 있습니다

```c
#include "lib-cross.h"

#include <stdio.h>

int main(){
    change_text_color(VIOLET);
    gotoxy(10, 10);
    printf("*");
    printf("\n");

    return 0;
}
```

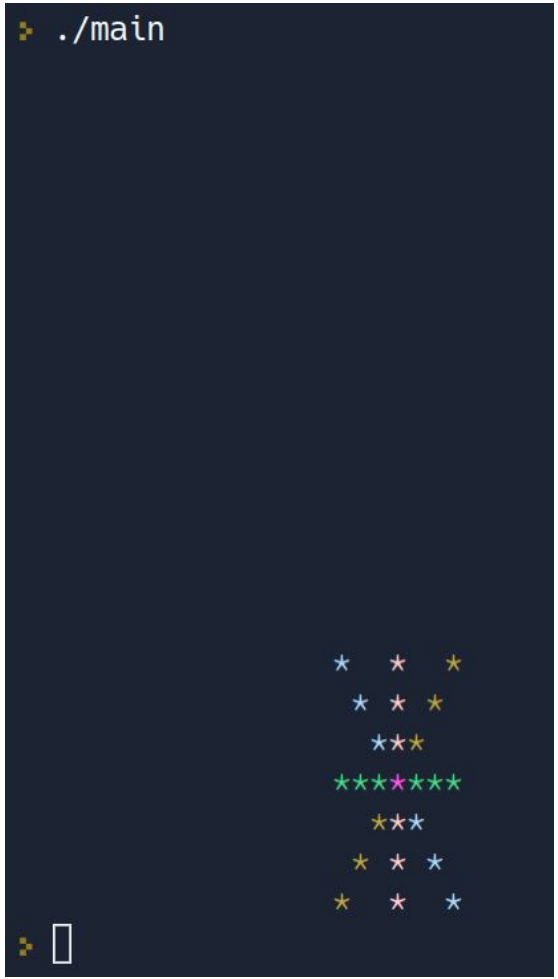
```c
enum Code {
    BLACK       = 0,
    DARK_BLUE   = 1,
    DARK_GREEN  = 2,
    DARK_SKY_BLUE = 3,
    DARK_RED    = 4,
    DARK_VIOLET = 5,
    DARK_YELLOW = 6,
    GRAY        = 7,
    DARK_GRAY   = 8,
    BLUE        = 9,
    GREEN       = 10,
    SKY_BLUE    = 11,
    RED         = 12,
    VIOLET      = 13,
    YELLOW      = 14,
    WHITE       = 15,
};
```

Visualization

- 배운 함수들을 활용하여 별을 그리는 draw_star 함수를 만들어보세요!
  - draw_star 함수는 3가지 parameter를 받습니다
  - int size : 별의 크기(중심을 기준으로 한 쪽 방향으로 뻗어나간 길이)
  - int center_x : 중앙 별의 x 좌표
  - int center_y : 중앙 별의 y 좌표

Visualization

- 결과 예시




```
51 ▼ int main(){
52       int size = 3;
53
54       draw_star(size, 20, 20);
55   }
```

Visualization

- 코드 예시

```
5 ▼ void draw_star(int size, int center_x, int center_y){
6       int start_pos_x = center_x - size;
7       int start_pos_y = center_y - size;
8       int pos_x, pos_y;
9       int len = size*2+1;
10
```

Visualization

- 코드 예시

```
12  change_text_color(GREEN);
13  pos_x = start_pos_x;
14  pos_y = center_y;
15  for(int i = 0; i < len; i++){
16    gotoxy(pos_x++, pos_y);
17    printf("*");
18  }
19
20  change_text_color(BLUE);
21  pos_x = start_pos_x;
22  pos_y = start_pos_y;
23  for(int i = 0; i < len; i++){
24    gotoxy(pos_x++, pos_y++);
25    printf("*");
26  }
```

Visualization

- 코드 예시

```
28    change_text_color(RED);
29    pos_x = center_x;
30    pos_y = start_pos_y;
31 ▼  for(int i = 0; i < len; i++){
32      gotoxy(pos_x, pos_y++);
33      printf("*");
34    }
35
36    change_text_color(YELLOW);
37    pos_x = center_x + size;
38    pos_y = start_pos_y;
39 ▼  for(int i = 0; i < size*2+1; i++){
40      gotoxy(pos_x--, pos_y++);
41      printf("*");
42    }
43
44    change_text_color(VIOLET);
45    gotoxy(center_x, center_y);
46    printf("*");
47
48    gotoxy(0, center_y + size + 1);
49  }
```

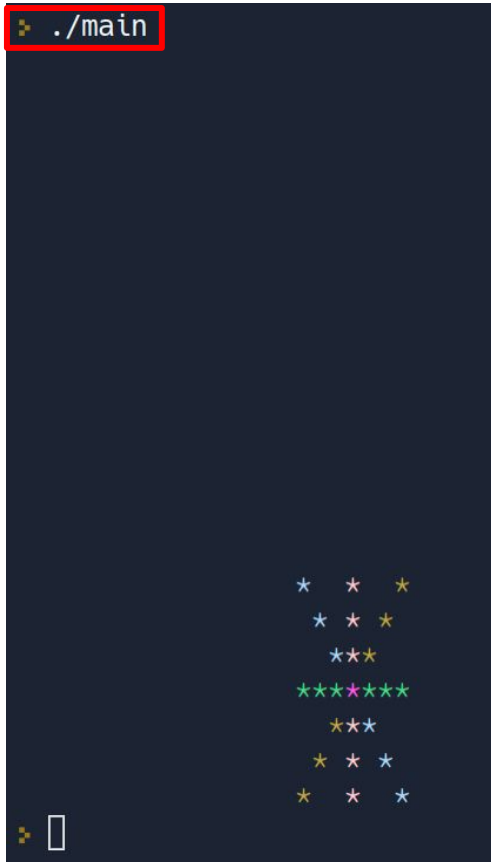
실습 6

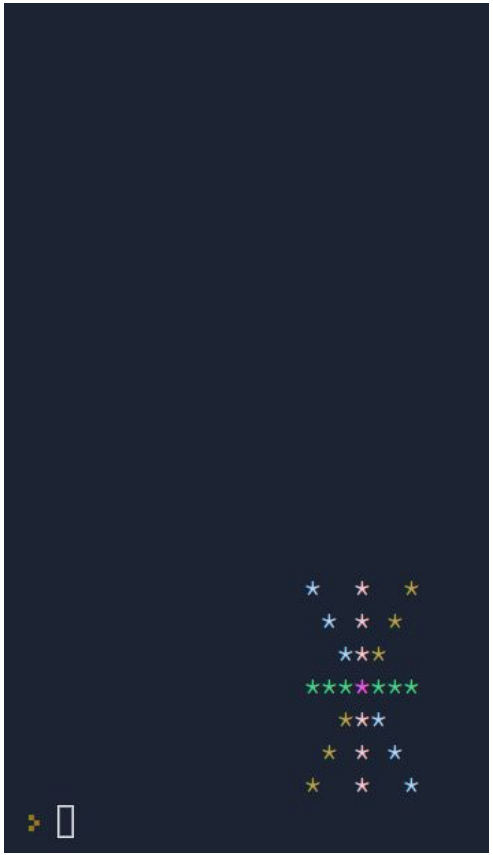
Visualization

- 별이 반짝 반짝 움직일 수 있도록 코드를 개선해봅시다!
  - 먼저 화면에 방해되는 친구들을 지워봅시다

Visualization

- 결과 예시





Visualization

- 화면을 비우고, 커서를 왼쪽 위에 위치시킵니다

```
5  void draw_star(int size, int center_x, int center_y){
6      int start_pos_x = center_x - size;
7      int start_pos_y = center_y - size;
8      int pos_x, pos_y;
9      int len = size*2+1;
10
11     screen_clear();
12
13     change_text_color(GREEN);
14     pos_x = start_pos_x;
15     pos_y = center_y;
```

Visualization

- 별이 반짝이는 효과를 넣어봅시다!
  - 작은 별과, 큰 별을 번갈아 가면서 화면에 그려봅니다!
  - while문 조건에 1을 주어서 무한히 반복시킵니다
  - 별의 위치는 (20, 20)으로 고정합니다
  - 작은 별의 크기는 3, 큰 별의 크기는 7로 합니다

Visualization

- 코드 예시

```
52 ▼ int main(){
53     int size[2] = {3, 7};
54     int i = 0;
55
56 ▼   while(1){
57       draw_star(size[i], 20, 20);
58       i = (i+1)%2;
59     }
60   }
```

- 실행해보면, 커서가 거슬립니다

Visualization

- 커서를 지워봅시다

```
52 ▼ int main(){
53     int size[2] = {3, 7};
54     int i = 0;
55
56     hide_cursor();
57
58 ▼   while(1){
59       draw_star(size[i], 20, 20);
60       i = (i+1)%2;
61     }
62   }
```

- 다시 실행해보면, 별이 그려지는 속도가 너무 빠릅니다

Visualization

- 별이 그려지는 속도를 조절해봅시다
  - 별과 별 사이에 sleep(delay)를 넣어봅시다
  - 라이브러리에 구현되어 있는 msleep은 넘겨주는 ms 만큼 프로그램의 동작을 쉽니다
  - 0.1초(100ms) 간격으로 별을 그릴 수 있게 delay를 추가해봅시다

```
52 ▼ int main(){
53     int size[2] = {3, 7};
54     int i = 0;
55
56     hide_cursor();
57
58 ▼   while(1){
59       draw_star(size[i], 20, 20);
60       i = (i+1)%2;
61
62       msleep(100);
63     }
64   }
```

Visualization

- 실행해보면, 정상적으로 별이 찍히지 않는 것을 확인할 수 있습니다
  - 프로그램이 쉬러가기 전에, standard out buffer에 쌓인 내용을 비워야합니다
  - 해당 기능을 수행하는 함수를 사용해봅시다

```
52 ▼ int main(){
53     int size[2] = {3, 7};
54     int i = 0;
55
56     hide_cursor();
57
58 ▼   while(1){
59       draw_star(size[i], 20, 20);
60       fflush(stdout);
61       i = (i+1)%2;
62
63       msleep(100);
64     }
65   }
```

Visualization

- 완성된 프로그램을 실행해서 별이 반짝이는 것을 확인해보세요!
  - draw_star에 추가했던, screen_clear 함수를 제거하고도 실행해보세요
    - 프로그램이 멈춘 것 처럼 보이지만, 실제로는 동작하고 있습니다
    - 이전에 그렸던 큰 별이 남아있는 상태에서, 그 위에 작은 별을 덮어씌워 그리기 때문입니다
    - 따라서, 화면을 새로 그리기 위해, 별을 그리기 전에 screen_clear 함수를 활용할 수 있습니다

Visualization

- kbhit 함수를 이용해서 키보드가 눌렸는지 확인할 수 있습니다
  - scanf와 다르게, 키를 누른 후 엔터를 치지 않아도 감지할 수 있습니다
- getch 함수를 이용해서 눌린 키의 정보도 알아낼 수 있습니다
- ‘c’ 키가 눌리면, 반복을 멈출 수 있도록 작성해봅시다

```
52 ▼ int main(){
53     int size[2] = {3, 7};
54     int i = 0;
55
56     hide_cursor();
57
58 ▼   while(1){
59       draw_star(size[i], 20, 20);
60       fflush(stdout);
61       i = (i+1)%2;
62
63       if(kbhit() && getch() == 'c')
64         break;
65
66       msleep(100);
67     }
68 }
```

Wordle: Final Project

채점 코드 Update

- 기존의 special case를 정상적으로 판단하지 못하던 부분을 변경하였습니다

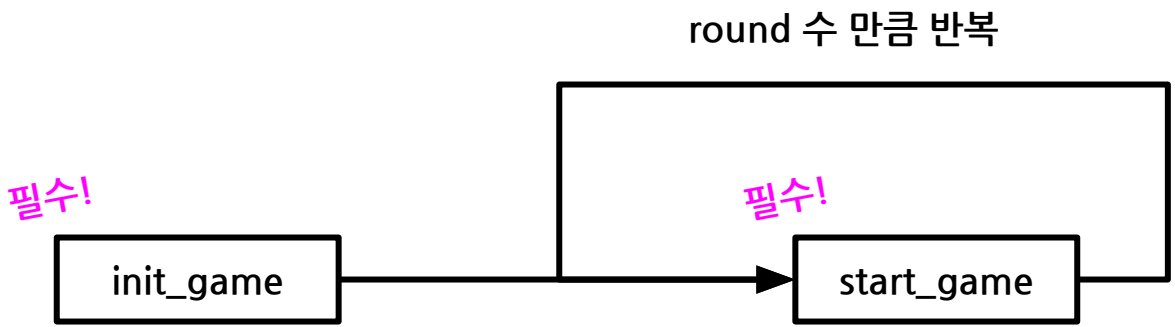
```c
void check_answer(const char *answer, char *result) {
  int checked[5] = {
      0,
  };

  // fill black
  for (int i = 0; i < 5; i++)
    result[i] = 'b';

  // check green
  for (int i = 0; i < 5; i++) {
    if (answer[i] == question[i]) {
      result[i] = 'g';
      checked[i] = 1;
    }
  }

  // check yellow
  for (int i = 0; i < 5; i++) { // for iterating answer
    if (result[i] == 'b') {
      for (int j = 0; j < 5; j++) { // for iterating checked
        if (!checked[j] && answer[i] == question[j]) {
          result[i] = 'y';
          checked[j] = 1;
        }
      }
    }
  }
}
```

문제 설정

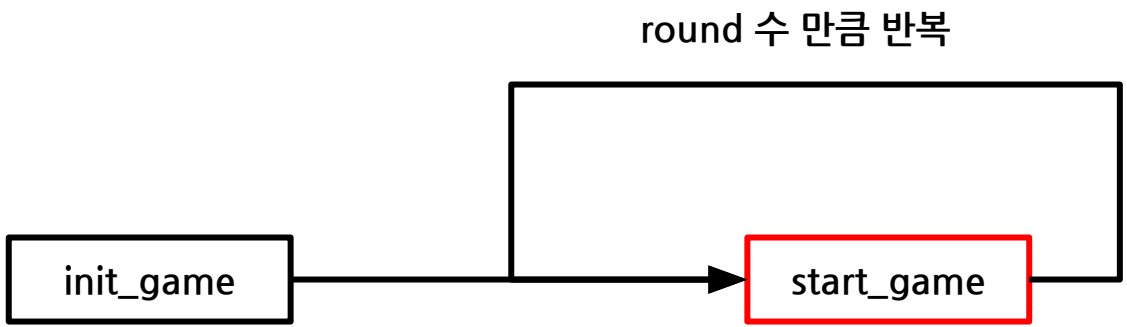
- wordle_ta.c 파일의 set_question 함수를 통해 문제를 설정할 수 있습니다
  - question 배열에 원하는 단어를 넣는 것으로 문제를 설정할 수 있습니다
  - 그림을 예시로, “askew” 부분을 원하는 단어로 변경 해보세요

```
24 ▼ void set_question(){
25 ▼     /* for random generation
26       for(int i = 0; i < WORD_LEN_MAX; i++)
27           question[i] = rand()%26 + 97;
28       question[WORD_LEN_MAX] = '\0';
29       */
30       strcpy(question, "askew");
31   }
```

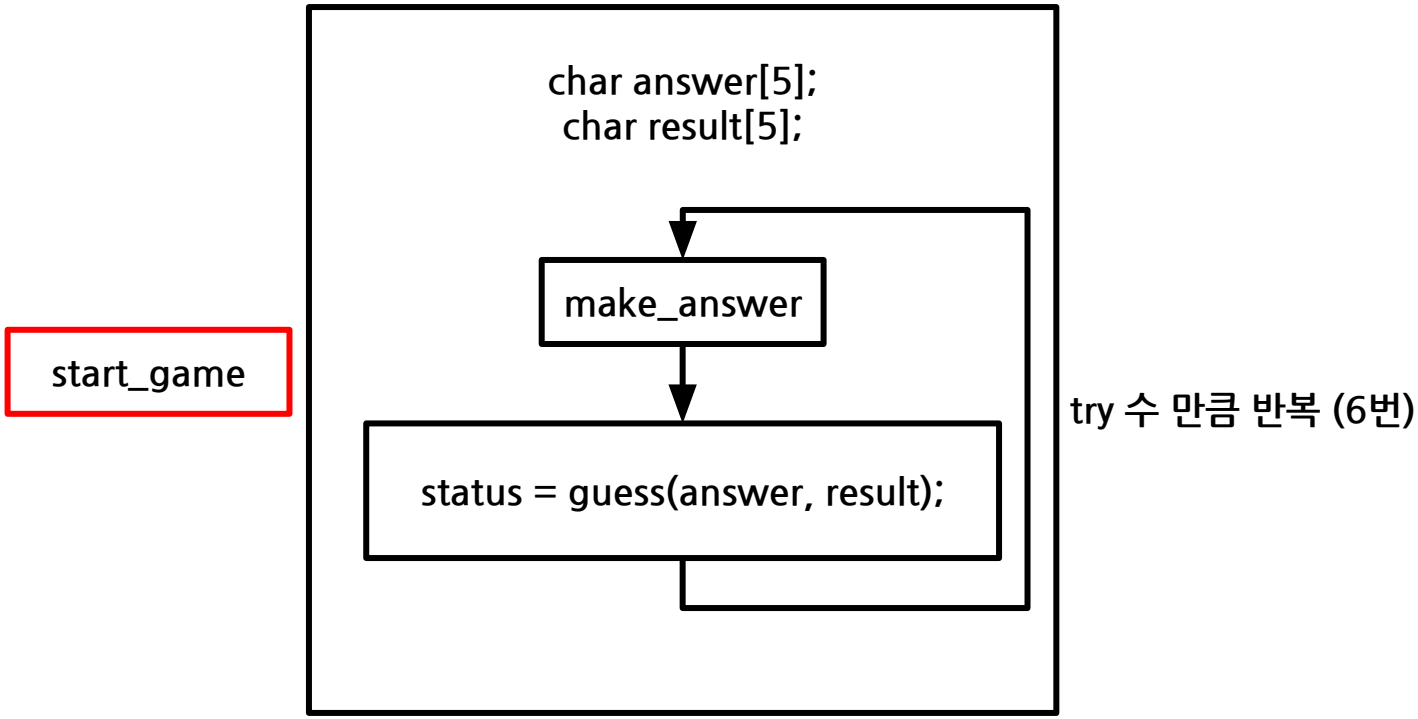
Flow Chart



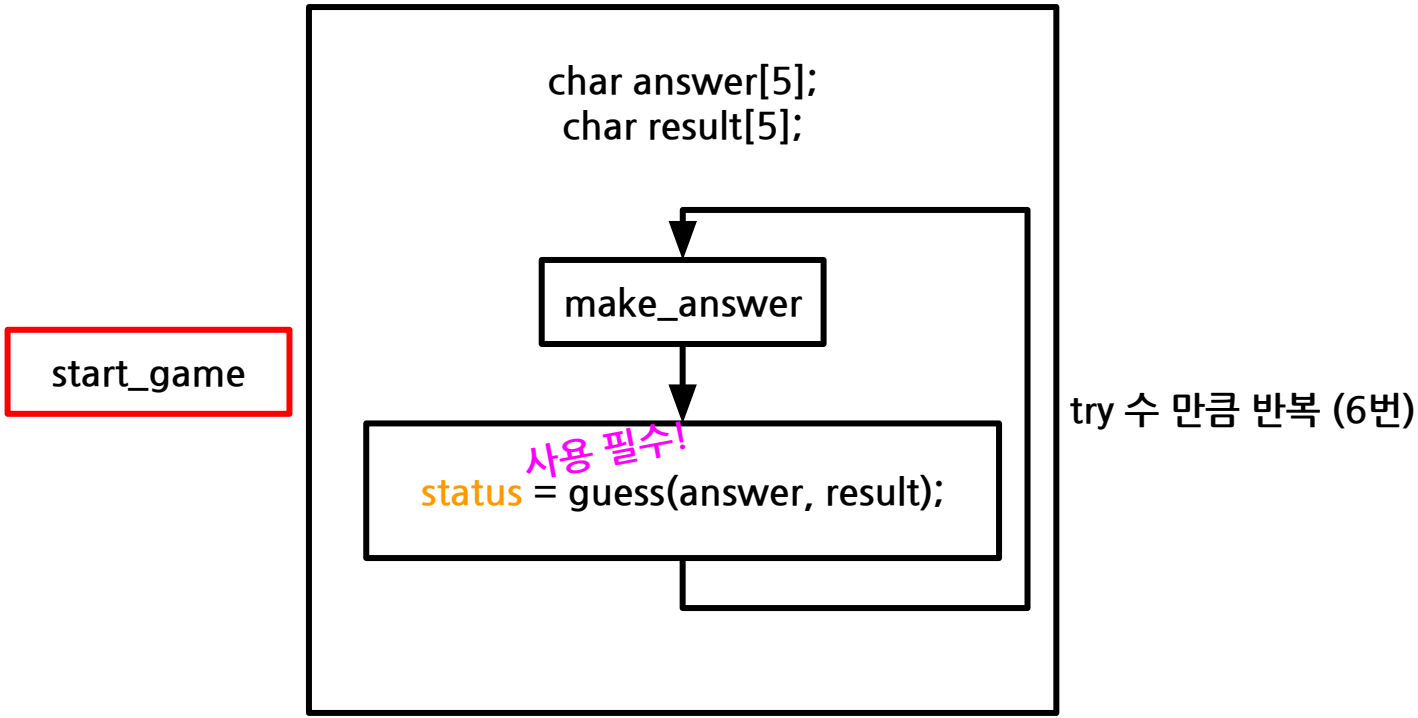
Flow Chart



Flow Chart


start_game
char answer[5];
char result[5];
make_answer
status = guess(answer, result);
try 수 만큼 반복 (6번)

Flow Chart


start_game
char answer[5];
char result[5];
make_answer
사용 필수!
status = guess(answer, result);
try 수 만큼 반복 (6번)

Status Information

wordle-st.h

```
6 ▼ enum error_list{
7   GOOD,
8   ERR_IMPROPER_WORD,
9   ERR_TRY_EXCEEDED,
10  };
```

result 배열에 정상적인 결과가 들어있음

answer가 단어 리스트에 없음

try 횟수가 6번을 초과함

Result 형태

- result 배열의 각 칸에 3가지 중 하나의 결과를 채워줍니다
  - ‘g’ : green - 해당 알파벳이 해당 위치에 있음
  - ‘y’ : yellow - 해당 알파벳이 다른 위치에 있음
  - ‘b’ : black - 해당 알파벳이 단어에 없음
- 따라서, 넘겨주는 result 배열은 최소 5칸이 있어야 합니다

Grading (**100점 만점**)

- Report (20점 + Extra 5점) - 최대 3장
  - 작성 항목 (총 20점 : 각 5점)
    - 주요 함수 설명
    - 간단한 Flow chart
    - 동작 화면
    - 팀 역할 분담
  - 기타 추가하고 싶은 내용들(Extra 5점)
    - 프로그램에서 자랑하고 싶은 부분
    - 참고 자료

Grading (**100점 만점**)

- Implementation (20점)
  - 필수 구현 함수가 정상적으로 구현되어 있는가? (각 10점)
    - init_game()
    - startGame()

Grading (100점 만점)

- Performance (60점)
  - TC를 수행하여 득점과 수행시간을 고려해 순위에 따라 차등 부여
    - ~ 5등 : 60점
    - 6 ~ 10등 : 55점
    - 11 ~ 15등 : 50점
    - 16 ~ 20등 : 45점
    - 21 ~ : 40점
    - 미동작 : 0점
- LeaderBoard 운영(실시간 순위 확인 가능)
  - 코드 제출은 Replit이 아니라 LeaderBoard로 해주세요!

Grading (**100점 만점**)

- Visualization (Extra +15점)
  - 기존에 제공된 예시 코드가 동작하는 수준이 Baseline
  - 다양한 색 활용 여부, 애니메이션 활용 여부, 다양한 Scene 구현 등을 고려

- Extra 점수 반영 방법
  - 만점은 100점 고정
  - Extra 점수로 100점을 초과할 수는 없음
    - 예 : 나머지 점수 90 + extra 점수 20점 = 100점(110점X)

주의사항

- 제출하는 프로그램의 모든 print는 반드시 render 관련 함수 안에만 있어야 합니다
  - printf, putchar와 같이 콘솔에 출력이 이루어지는 함수들은 반드시, rendering과 관련된 함수 내부에서만 사용되어야 합니다
  - 따라서, RENDER_OFF 옵션을 주었을 때, 프로그램은 어떠한 출력도 이루어져서는 안됩니다!

Q&A