4190.103A-001: Programming Practice Lab. 15

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

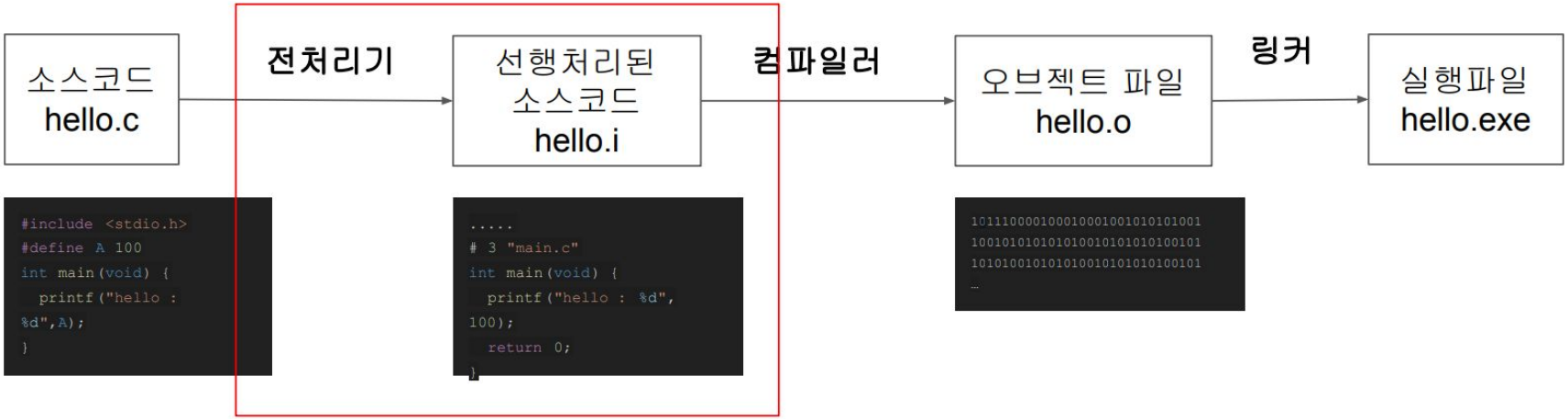
Systems Software & Architecture Lab.

Seoul National University

Spring 2023

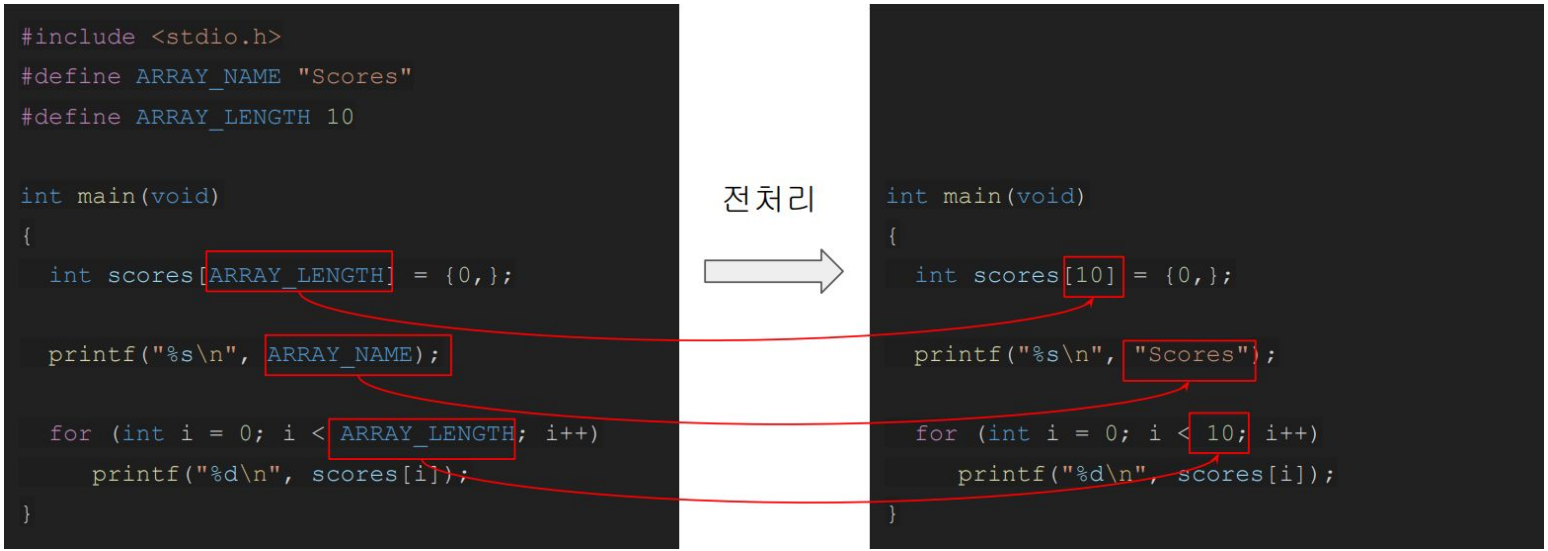
전처리기 (#define)

- 전처리기는 컴파일러 이전에 실행되어 텍스트 치환과 같은 소스코드의 텍스트를 조작하는 일을 진행함



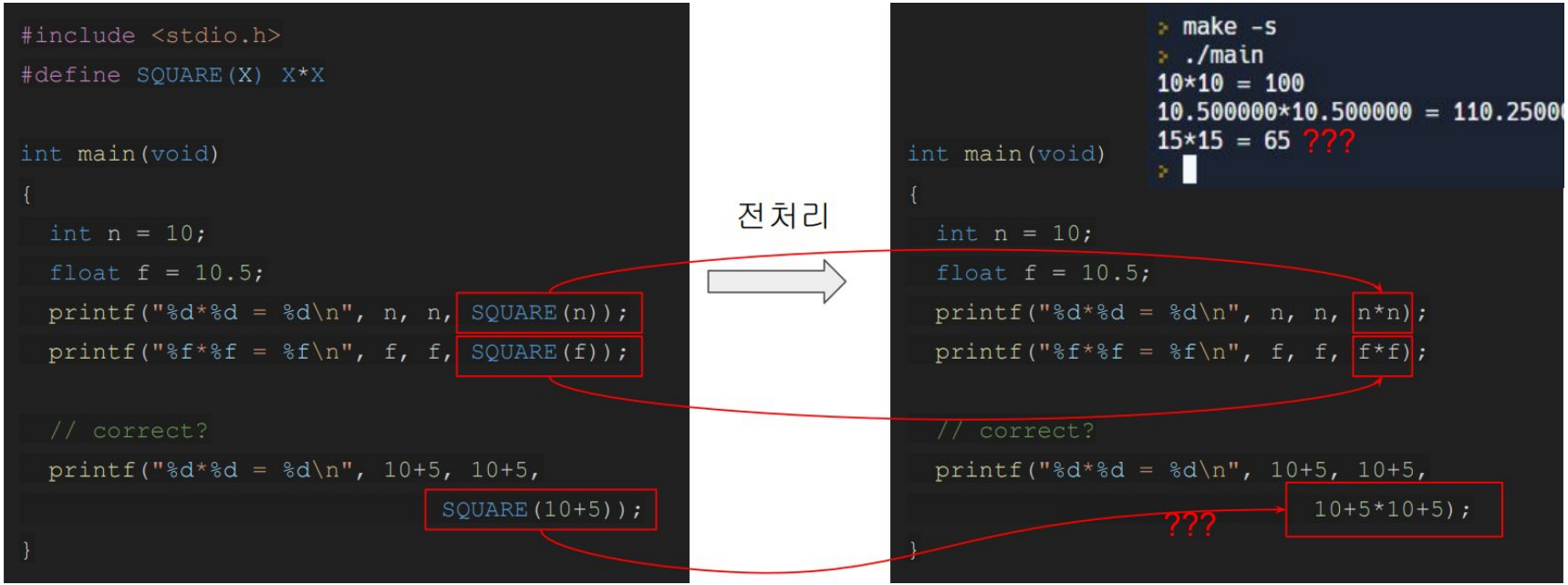
Object-like Macro (#define)

- 단순한 치환 과정을 진행함
- #define ARRAY_NAME “Scores” → ARRAY_NAME 를 “Scores” 으로 치환



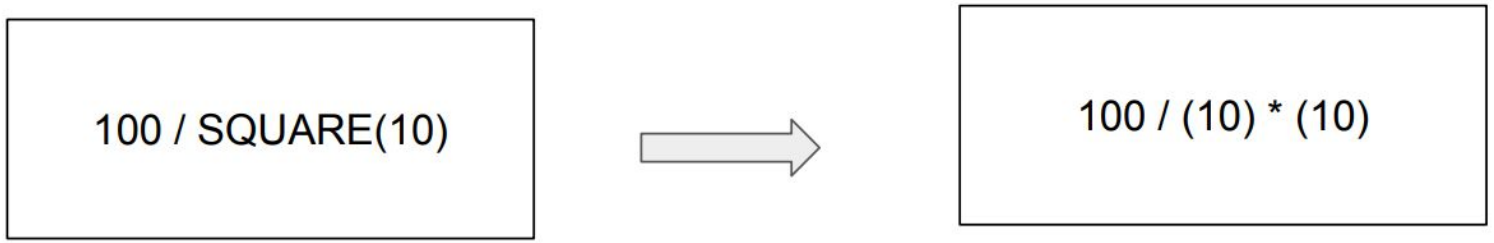
Function-like Macro (#define)

- #define SQUARE (X) X*X → SQUARE(X) 패턴을 X*X 로 치환



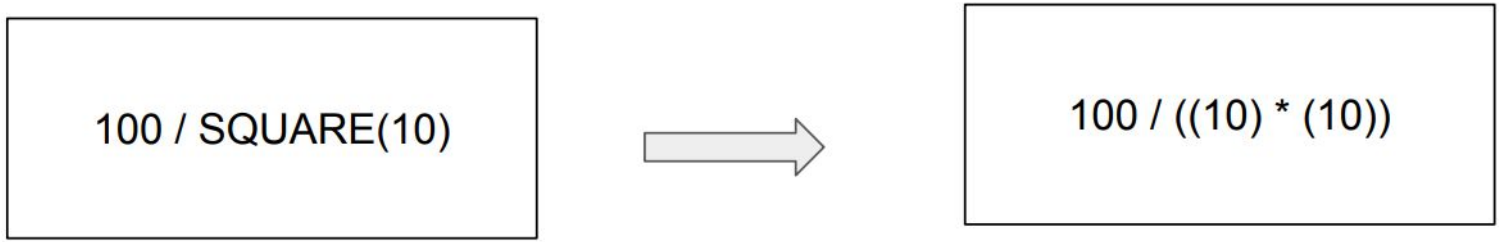
Function-like Macro (#define)

- #define SQUARE(X) (X)*(X)
  - 괜찮을까요?



Function-like Macro (#define)

- #define SQUARE(X) ((X)*(X))



Function-like Macro (#define)

- 매크로 사용시 주의사항
  - 매크로 사용시 소괄호를 올바르게 사용하지 않으면 연산자 우선순위 때때문에 의도한대로 프로그램이 동작하지 않을 수 있습니다.
  - 매크로를 잘 못 사용했을시 디버깅이 쉽지 않습니다.
  - 따라서 매크로를 정의할 때 소괄호를 많이 이용하여 의도한 순서대로 연산이 되도록 하는 것이 좋습니다.

Function-like Macro (#define)

- 이미 정의된 매크로를 이용한 매크로
  - 앞서 정의된 매크로를 사용해서 새로운 매크로를 생성할 수 있습니다.

```c
#include <stdio.h>
#define KB(X) ((X) * 1024)
#define MB(X) (KB(X) * 1024)
#define GB(X) (MB(X) * 1024)
#define TB(X) (GB(X) * 1024)

int main(void) {
  long n = 1;
  printf("%ldKB : %ldByte\n", n, KB(n));
  printf("%ldMB : %ldByte\n", n, MB(n));
  printf("%ldGB : %ldByte\n", n, GB(n));
  printf("%ldTB : %ldByte\n", n, TB(n));
  return 0;
}
```

전처리기 (#if)

- 전처리과정에서 #if 조건이 참이라면 #endif 까지 컴파일 대상에 포함, 거짓이라면 컴파일 대상에 포함하지 않음

```
#define DEBUG 1

int main(void) {
  int a,b;
  scanf("%d %d",&a, &b);

  #if DEBUG
  printf("a: %d b: %d\n", a, b);
  #endif

  return 0;
}
```

전처리기 (#elif, #else)

- #elif, #else 를 통해 분기를 만들어낼 수 있습니다.

```c
#define DEBUG_LEVEL 0
int main(void) {
  int a = 10, b = 20;

  #if DEBUG_LEVEL == 2
  printf("a: %d b: %d \n", a, b);
  #elif DEBUG_LEVEL == 1
  printf("a: %d\n", a);
  #else
  printf("nothing\n");
  #endif

  return 0;
}
```

실습

실습 1 - MIN(), MAX(), DIV_ROUND_UP()

- 아래 3개의 매크로를 작성해 보세요
  - #define MIN(A, B)
  - #define MAX(A, B)
  - #define DIV_ROUND_UP(A, B)
    - A, B 중 큰 수에서 작은 수로 나누어서 올림함
    - DIV_ROUND_UP(10, 100) => 10
    - DIV_ROUND_UP(10, 109) => 11
- Tip: 여러 줄에 걸쳐서 매크로를 정의하려면 줄 끝에 ₩(역 슬래시)를 삽입하면 됩니다.

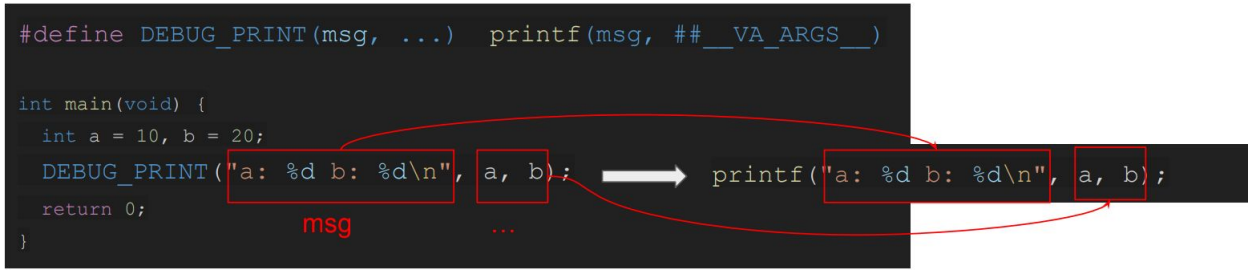
```
#define ADD(A, B) ((A) \
                   + (B))
```

실습 3 - DEBUG_PRINT()

- 디버깅을 하기 위해서 여러가지 로그들을 프로그램 중간 중간에 삽입합니다.
- 삽입 한 디버깅 로그들은 실제 프로그램이 동작하는데는 필요가 없지만 추후 문제가 발생했을시 해당 코드들을 다시 사용할 수도 있습니다.
- 즉, 디버깅 로그들을 쉽게 끄고 킬 수 있는 기능을 이용하여, 작성된 디버깅로그들을 관리하는 것은 개발의 효율을 높여줍니다.
- 이러한 기능은 매크로를 통해 구현할 수 있습니다.

실습 3 - DEBUG_PRINT()

- 가변인자를 이용한 디버깅 출력 매크로를 만들어봅시다.
  - … : 가변인자
  - ##__VA_ARGS__ 는 …(가변인자)에 대응되어 치환됩니다.



- 연속된 문자열 상수는 접합된다는 점을 이용해주세요
  - printf(“abc””def”) → printf(“abcdef”)

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음주 수요일 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용