시스템 프로그래밍을 위한 C언어

Preprocessing/Compiling/Assembling/Linking

현대자동차 입문교육
박대진 교수

What is Programming ?

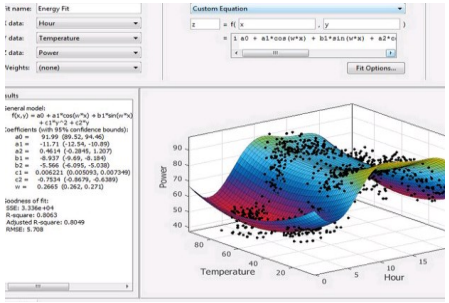
기계가 알아들을 수 있게 할일을 번역하는 작업

The art of
making a computer do
what you want it to do

Design (DS) + Coding (Algorithm)

Efficiency in
time/space/power

How to communicate
between human and
computer?

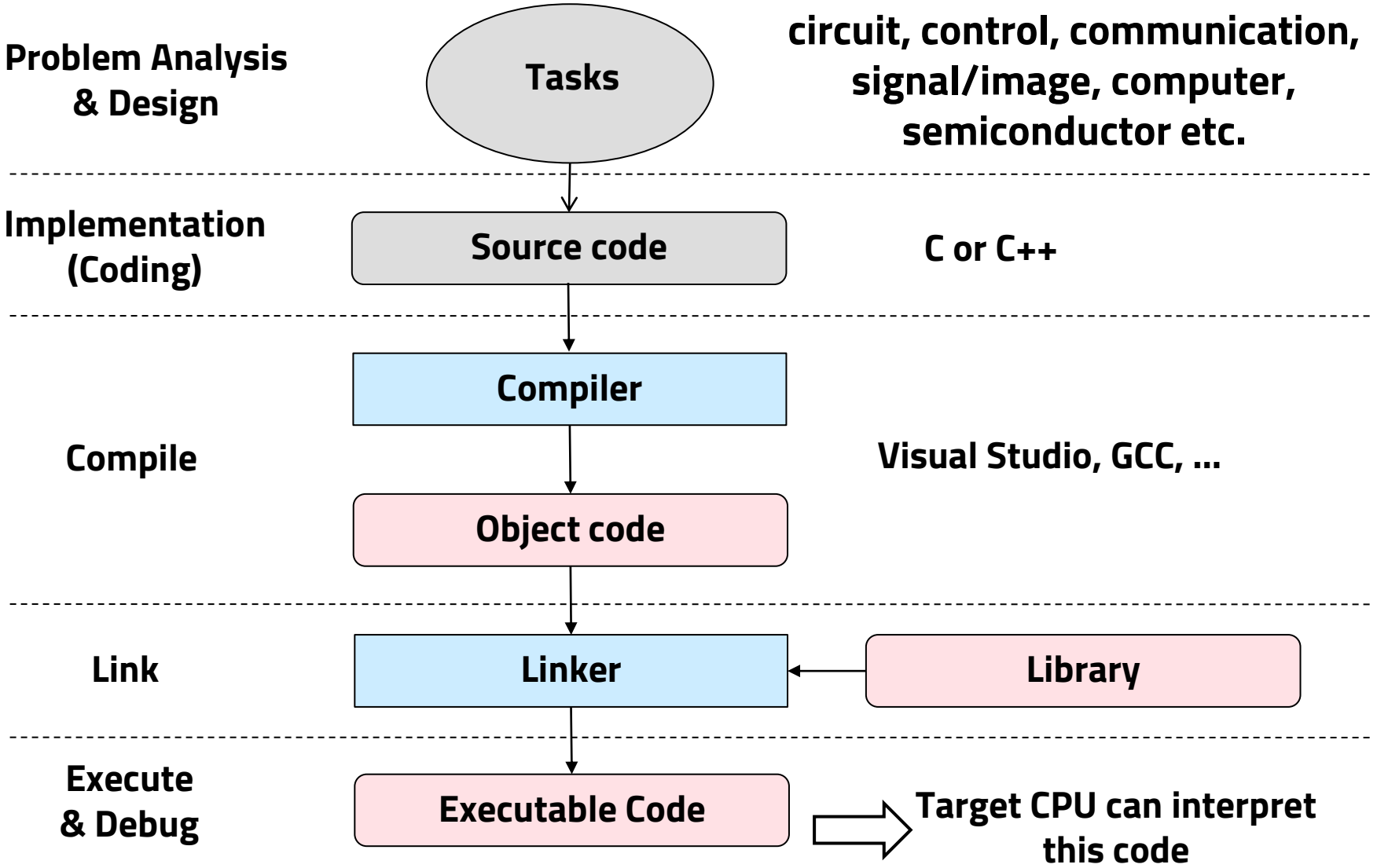


Machine language

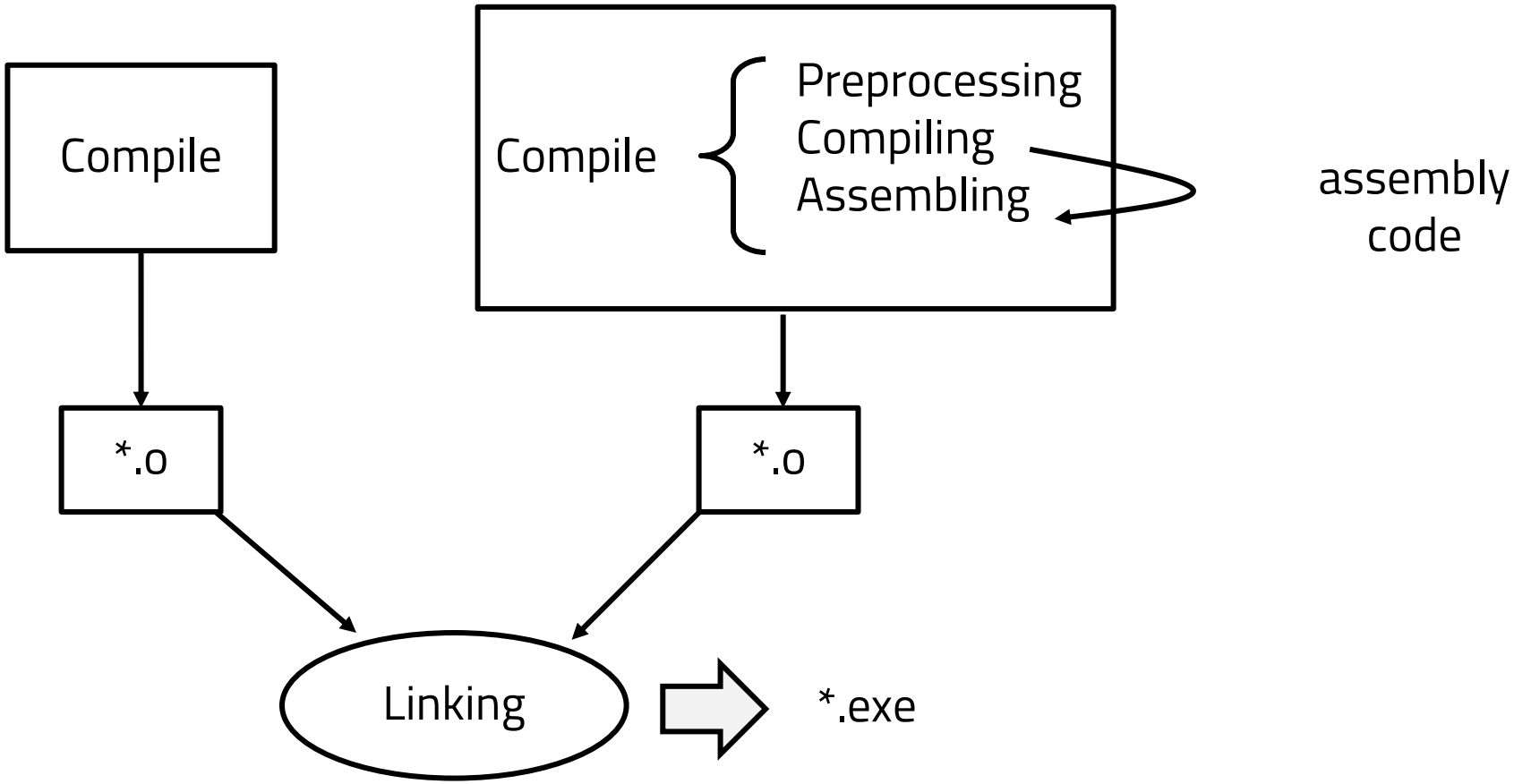
```
00401028  C7 45 FC 01 00 00 00 C7
00401030  45 F8 64 00 00 00 8B 45
00401038  FC 03 45 F8 89 45 FC F0
00401040  5E 5B 8B E5 5D C3 CC CC
```

Compilation

Tasks → Executable Binary Code



Detailed View of Compilation



Compiler & Compilation
명령어 기반

// Pre-processing only

% cpp hello.c > hello.i → Concatenated C code 출력

// Compilation only

% gcc –S hello.i → Assembly code 출력

// Assembling only

% as –o hello.o hello.s → Object code 출력

// Linking only

% ld –o hello.exe hello.o …. additional library …

→ Executable code 출력

% gcc –o hello.exe hello.c

Compiler & Compilation

// All (preprocessing, compiling, assembling) except linking

```
% gcc –c hello.c
```

→ Object code hello.o 출력

// linking multiple objects

```
% gcc –o hello.exe hello1.o hello2.o
```

→ Executable code 출력

// Compile 과정 보여주기

```
% gcc –-verbose –o hello.exe hello.c
```

// header path 지정

```
% gcc –o hello.exe hello.c –IC:\foo\include
```

#include <XXXX.h>의 파일 경로 지정가능

Makefile – Batch Automation

Makefile


```
task1:
	execution list #1…
task2:
	execution list #2…
```

% make task1
➔ Perform execution list #1
% make task2
➔ Perform execution list #2

```
task1: task2 task3
	execution list #1…
task2:
	execution list #2…
task3:
	execution list #3…
```

% make task1
➔ After task2, task3 are performed if necessary, task1 will be done.

Makefile Example

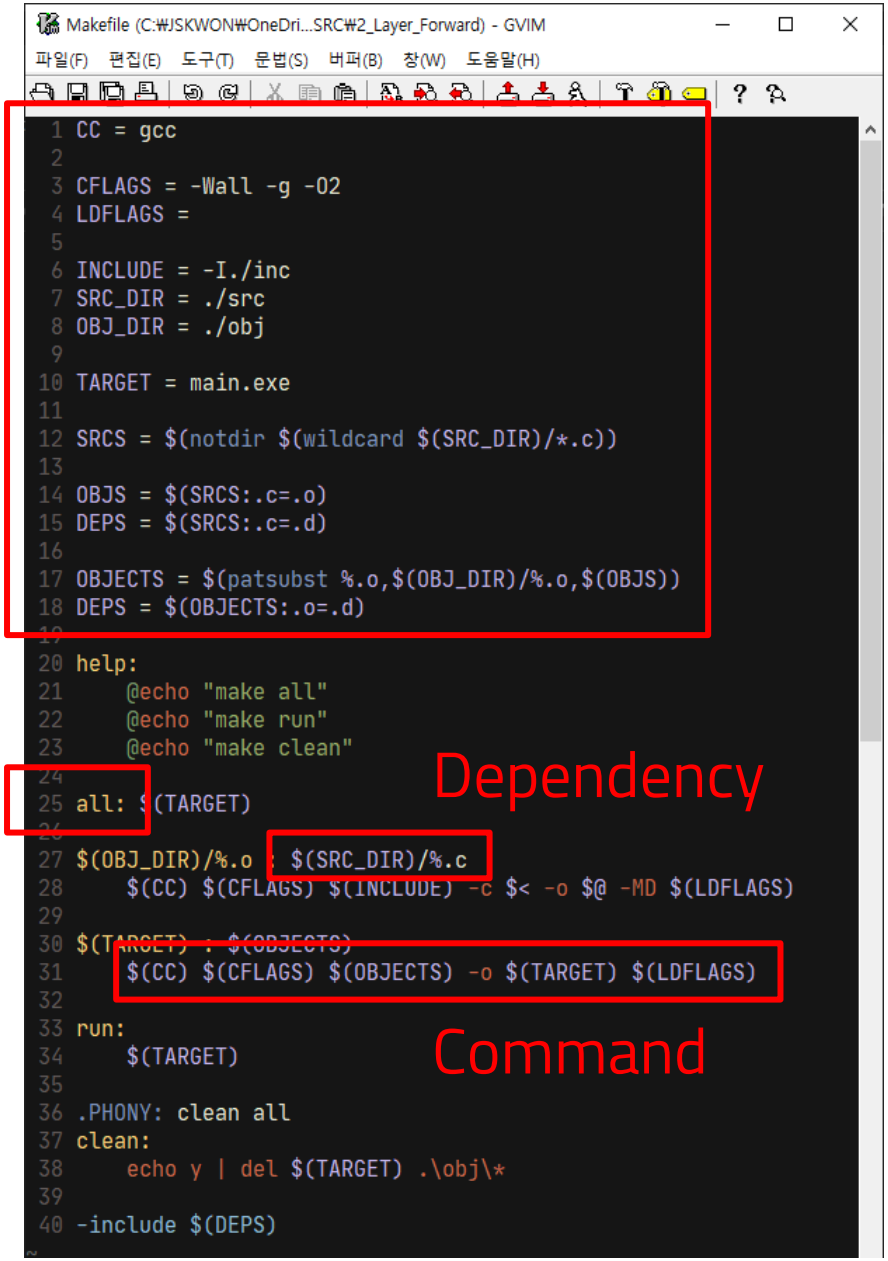
Macro

Target

Dependency

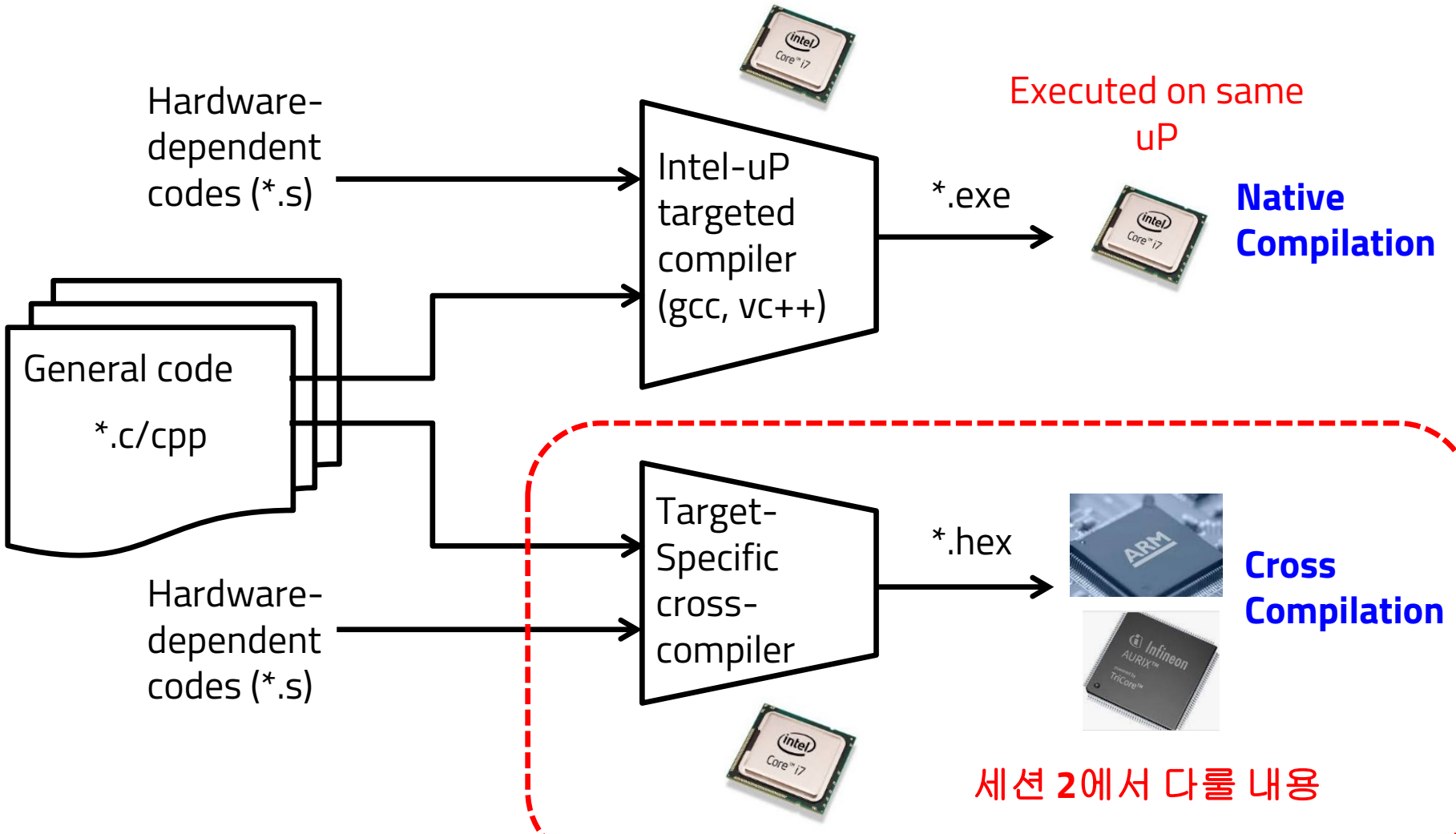
Command

조건에 따라서 commands를 실행할 **Rule**



Native & Cross Compilation

단일 코드로부터 복수의 타겟프로세서용 실행 코드 생성



잊지 말자!

- 결국은 1과 0으로 채워진 바이너리 코드로 만들어져야 온칩에 적재된다.
- CPU도 1과 0으로 된 바이너리 코드를 읽어서 실행하기 때문이다.
- 1과 0로 직접 코드를 표현하기 어렵기 때문에 컴파일러를 사용해야 한다.
- 컴파일러를 실행할 운영체제에 따라 다양한 컴파일러가 존재한다.
- 타겟 프로세서에 맞는 다양한 컴파일러가 존재한다.
- 코드를 작성하기 위해서는 에디터가 필요하다. 잘 다룰 수 있어야 한다.
- GUI로 구현된 컴파일러도 결국은 컴파일러 명령어를 호출할 뿐이다.
- 세부적인 조작을 위해서는 명령어 기반 컴파일 과정도 잘 이해하면 좋다.
- 컴파일한 환경에서 바로 코드를 실행해볼 수 있다면 Native Compilation이다.
- 컴파일한 코드를 타겟 프로세서로 옮겨야만 실행할 수 있다면 Cross Compilation이다.

시스템 프로그래밍을 위한 C언어

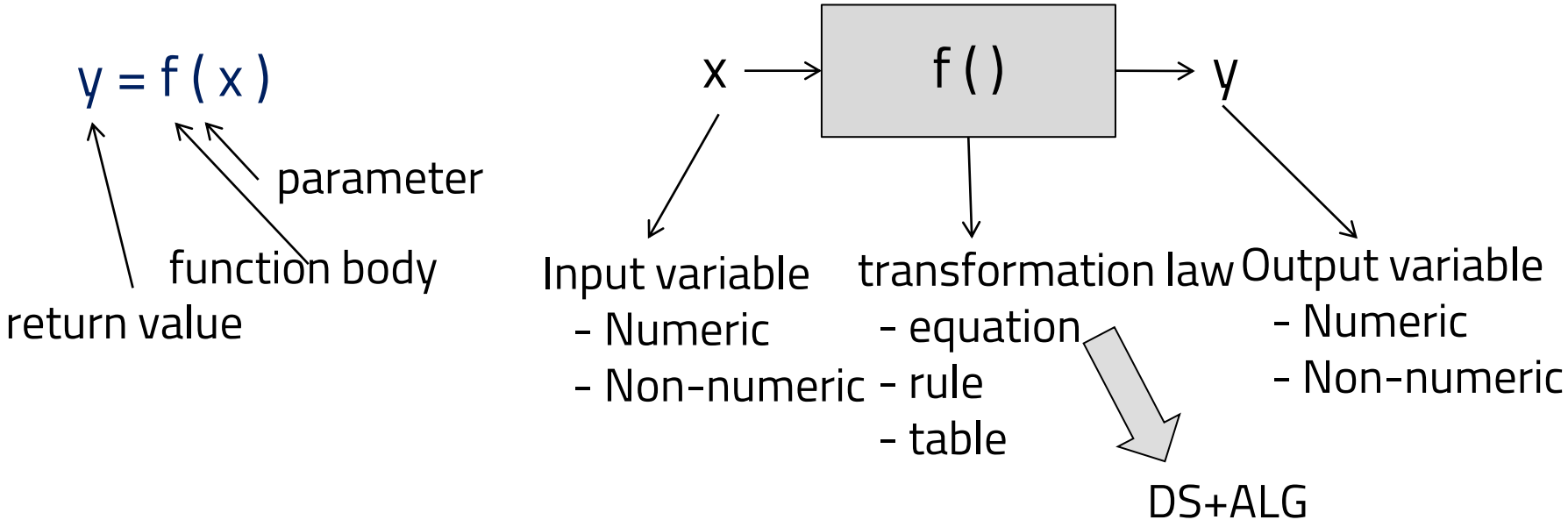
C프로그래밍 Hello World 기초 복습

현대자동차 입문교육
박대진 교수

Lecture Lessoned

- C언어 구조 소개
  - 파일 분리
  - 변수
  - 함수

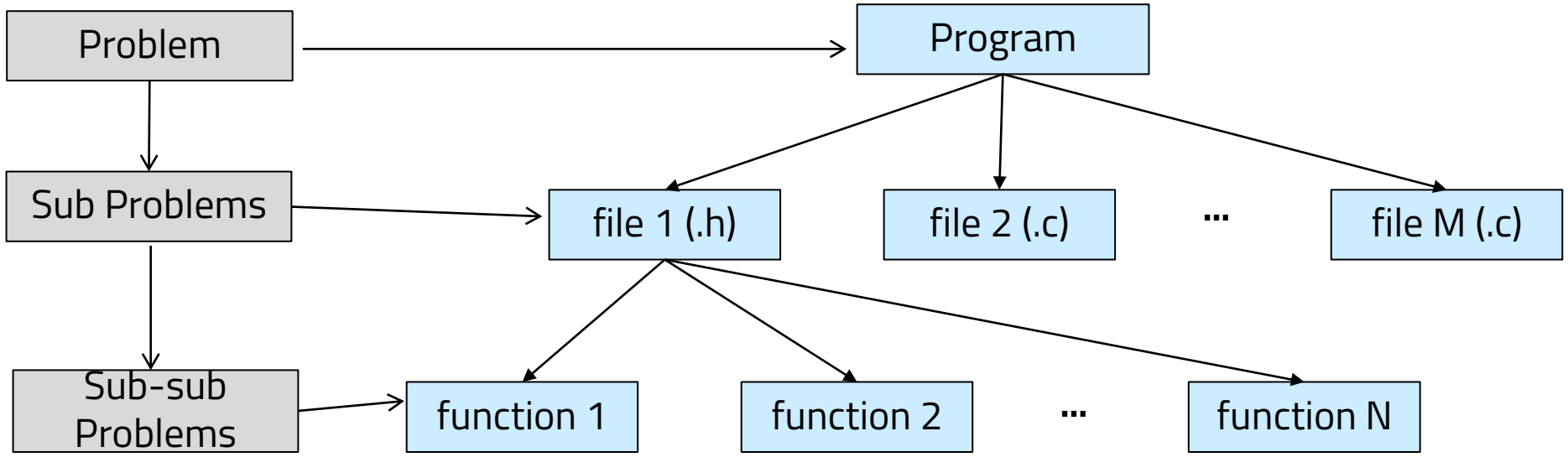
Everything is Function in C World



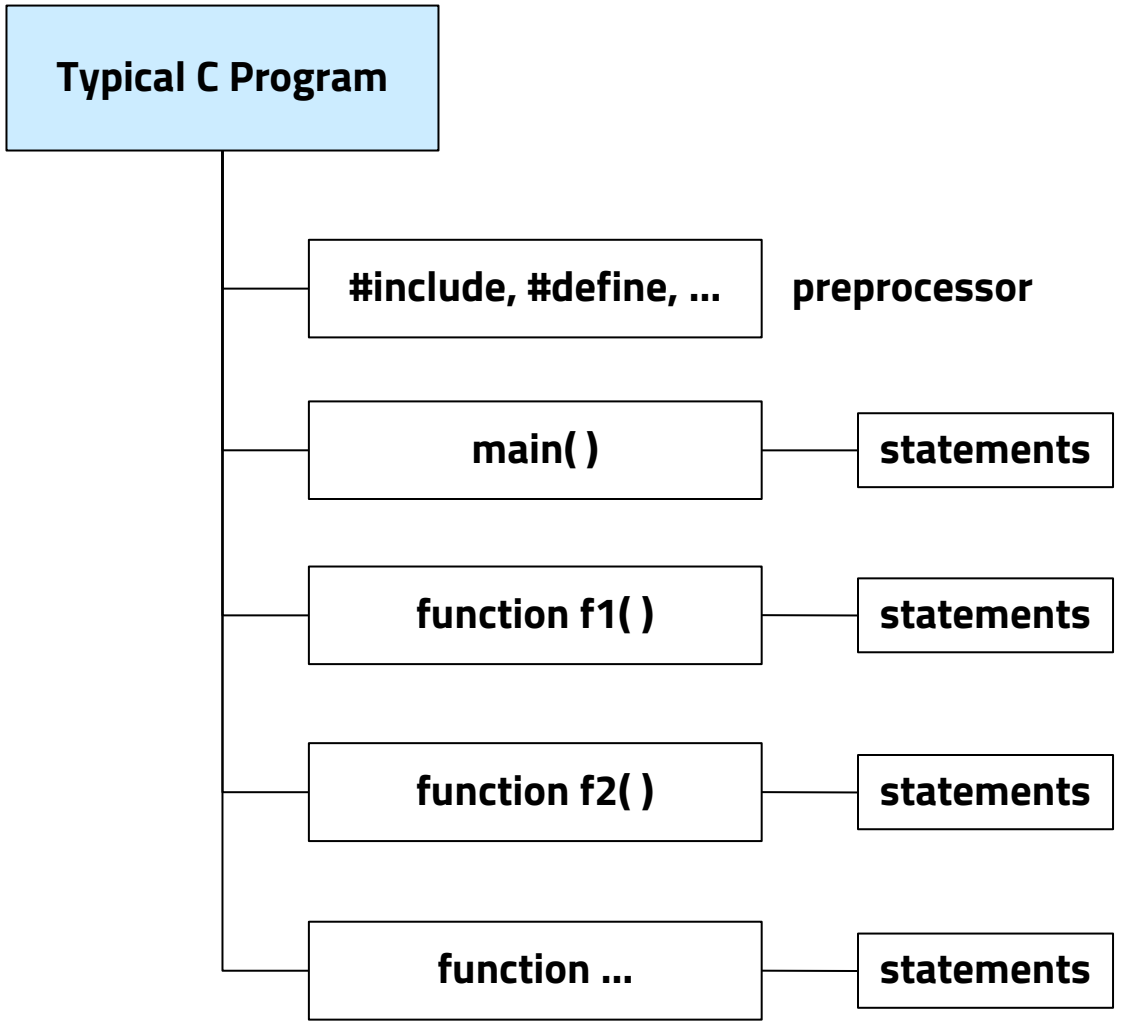
Problem-solving is done in function

Problem Analysis → Dividing Problem into Functions

- Problem solving with C language
  - A problem is divided into sub problems
  - Each sub problem is solved by function



Structure of Typical C Program

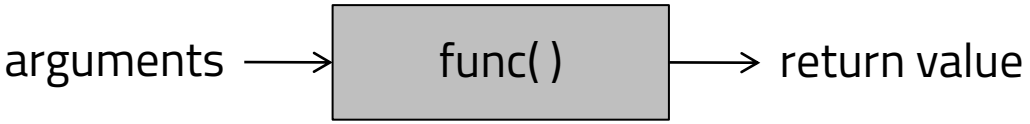



```
#include   <stdio.h>
int factorial(int a);
main() {
   int a, result;

   printf("input: ");
   scanf("%d", &a);

   if (a>=0) {
      result = factorial(a);
      printf ("%d! = %d\n", a, result);
   } else {
       printf ("ERROR: negative integer\n");
   }
}
```

```
/* calculate n! */
int factorial(int a) {
   int i;
   int result = 1;
   for (i=1; i<=a; i++)
         result = result * i;
   return result;
}
```

Function in C

- Black box as function representation



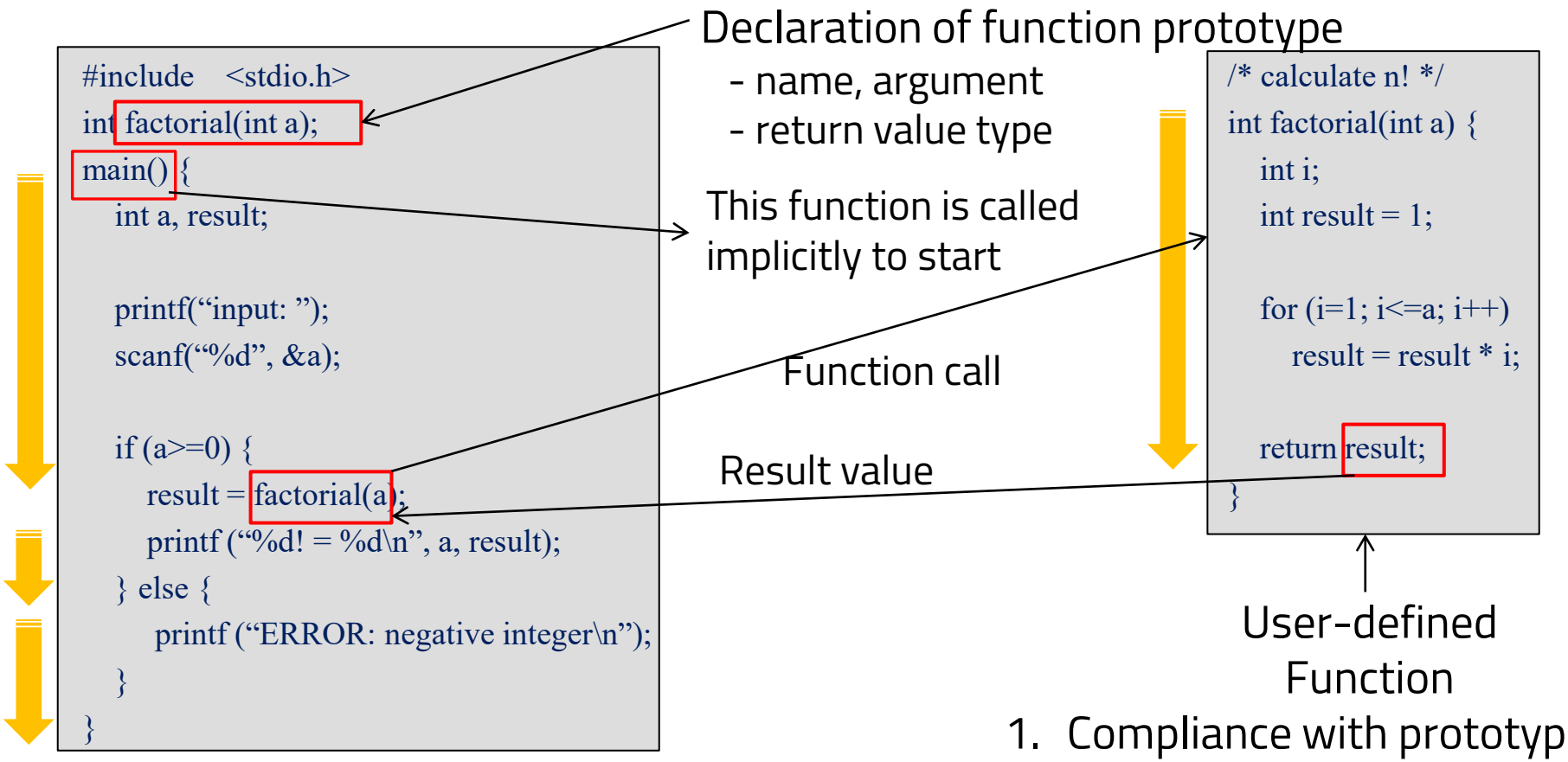
- Modular Programming

  → No variables within a function is shared between functions.

  - Easy to program by concentration of each function.
  - Easy to modify and maintain within each function.

- Types of functions
  - Standard library functions
    - printf(), scanf(), getchar(), putchar(), …
    - Standard library header files (ex. stdio.h)
  - User-defined functions
    - main()
      - Special function to be called implicitly to start execution of a C program
    - Prototype must be declared or function body must be defined before use (function call)

Function: Declaration vs. Call



1. Compliance with prototype
2. Body of function as a sequence of statements

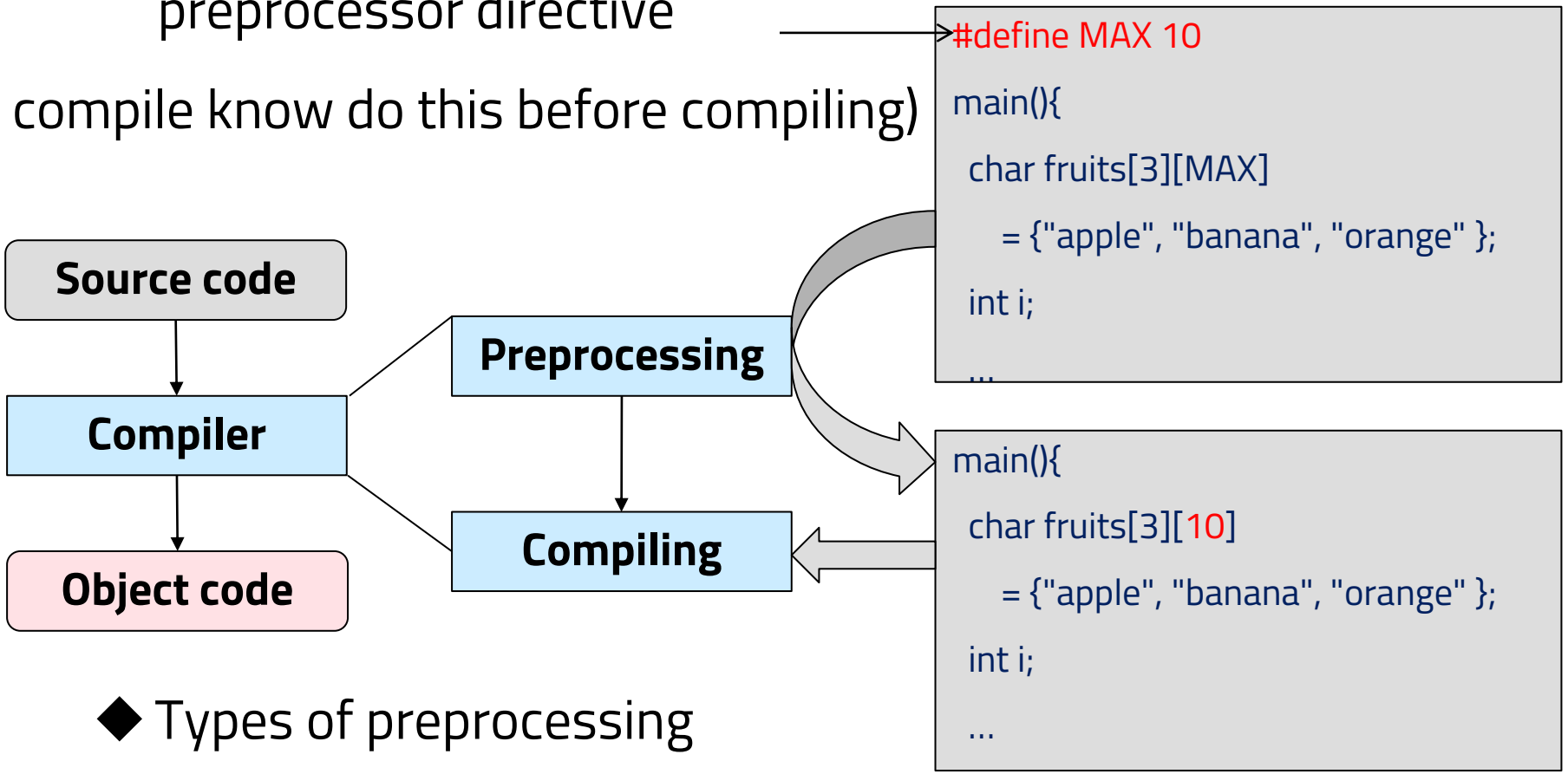
◆Function

- ❖ main() : caller ↔ factorial() : callee
- ❖ Functions should be modular.

C Preprocessor

preprocessor directive

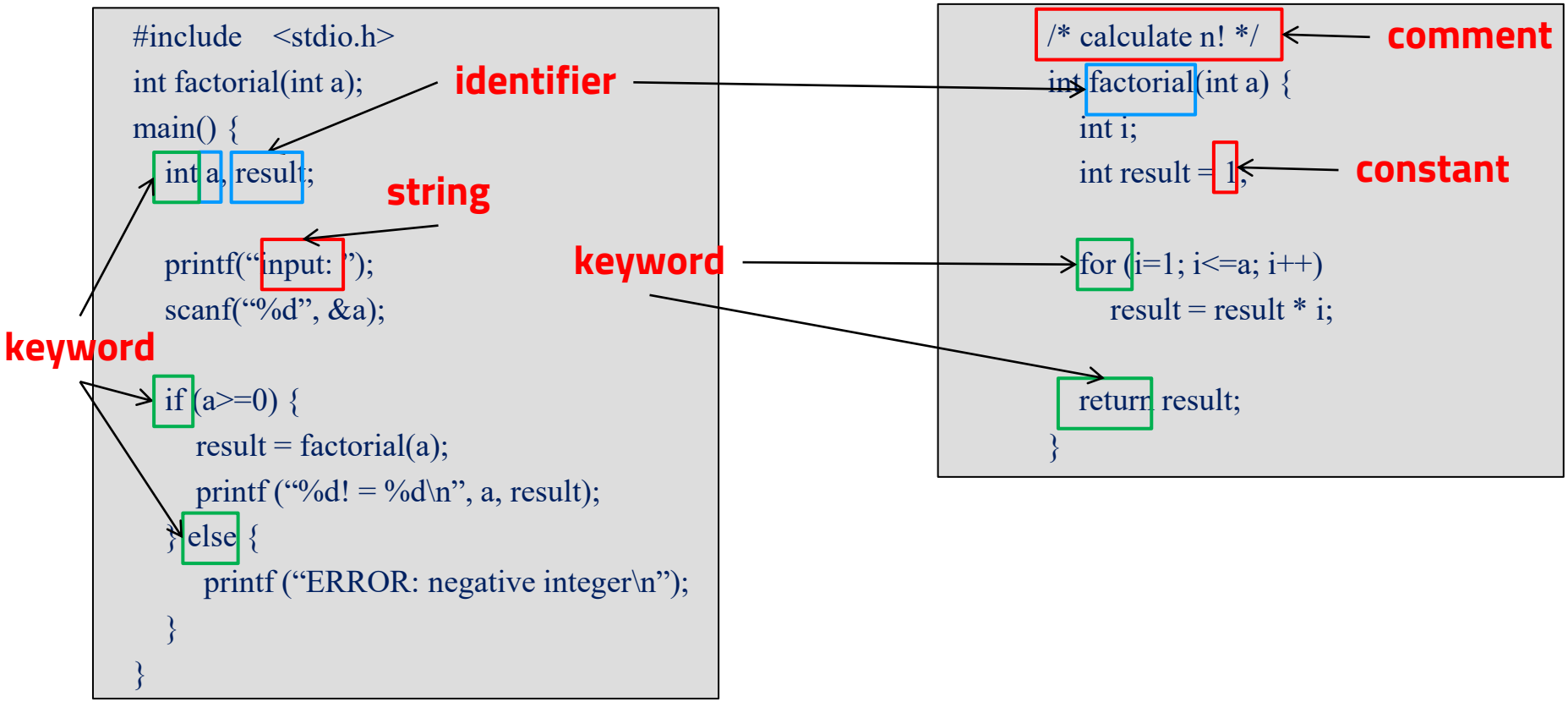
(let compile know do this before compiling)



- Types of preprocessing
  - Macro substitution
  - Conditional compilation
  - inclusion of named files

C Lexical Conventions

◆Words used in C



◆Comments

- ❖The characters /* introduce a comment, which terminates with the characters */.

Reserved Keywords: You should know

(Cannot be used as identifiers)

		keyword
data type		**char, enum, float, int**, long, short, **void**, union, unsigned, **typedef, struct**, double
control flow	loop	**for, while, do**
	decision	**if, else, switch, default, case**
	branch	goto, break, continue, **return**
storage class		auto, **extern**, **static**, register
etc		const, sizeof

잊지 말자!

- C언어는 함수들의 집합이다.
- 운영체제 혹은 시스템 리셋에서 최초로 main()함수를 실행하기로 약속했을 뿐이다. (태초에 하나님이...)
- 일단 main함수로 진입이후에는 프로그래머가 다음 함수 call을 결정한다.
- 코드를 여러 파일에 나누어서 코딩후 컴파일하고 그것을 링킹을 통해 합친다.
- 코드를 함수를 넘나들면서 순차적인 수행을 한다 (변수 전달, 리턴값 받기)
- C언어에 포함된 기본적인 문법을 잘 복습해야 한다.