

프로그래밍언어

# 1. C 프로그래밍의 기초



교수 김 영 탁

영남대학교 정보통신공학과

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

# Outline

- ◆ 컴퓨팅사고와 소프트웨어 개발
- ◆ 알고리즘, 유사코드 (pseudo code)
- ◆ C 프로그램 작성 절차(Visual Studio)
- ◆ C 프로그램 구성요소
- ◆ 디버깅 (debugging)
- ◆ Visual Studio Community 2019 설치 및 사용
- ◆ C/C++ 예제 프로그램 작성 및 실행

# 컴퓨팅사고 (Computational Thinking)와 소프트웨어 설계

# 컴퓨팅사고와 프로그래밍

## ◆ 인류문명의 진화

- 새로운 도구와 재질, 동력장치를 사용하여 **생산성** (*productivities*)을 향상:
  - 석기, 청동기, 철기, 증기(steam) 엔진, 전기, 산업용 로봇, **컴퓨터**, 인터넷, 분산처리, 인공지능 (AI), 빅데이터, 사물인터넷 (IoT)
- 다수 인원의 효율적인 **협동/협력** (*collaborations*)
- 부가가치 창출의 **성능** (*performance*) 개선
- **에너지 효율성** (*energy efficiencies*)



(a) 석기 망치, 창



(b) 철기 농기구



(c) 증기 (steam) 엔진



(d) 전기, 산업용로봇



(e) 컴퓨터, 인터넷, 분산처리



(f) 인공지능 (AI), 빅데이터, 사물인터넷(IoT)

# 제4차 산업혁명과 프로그래밍

## ◆ 산업혁명

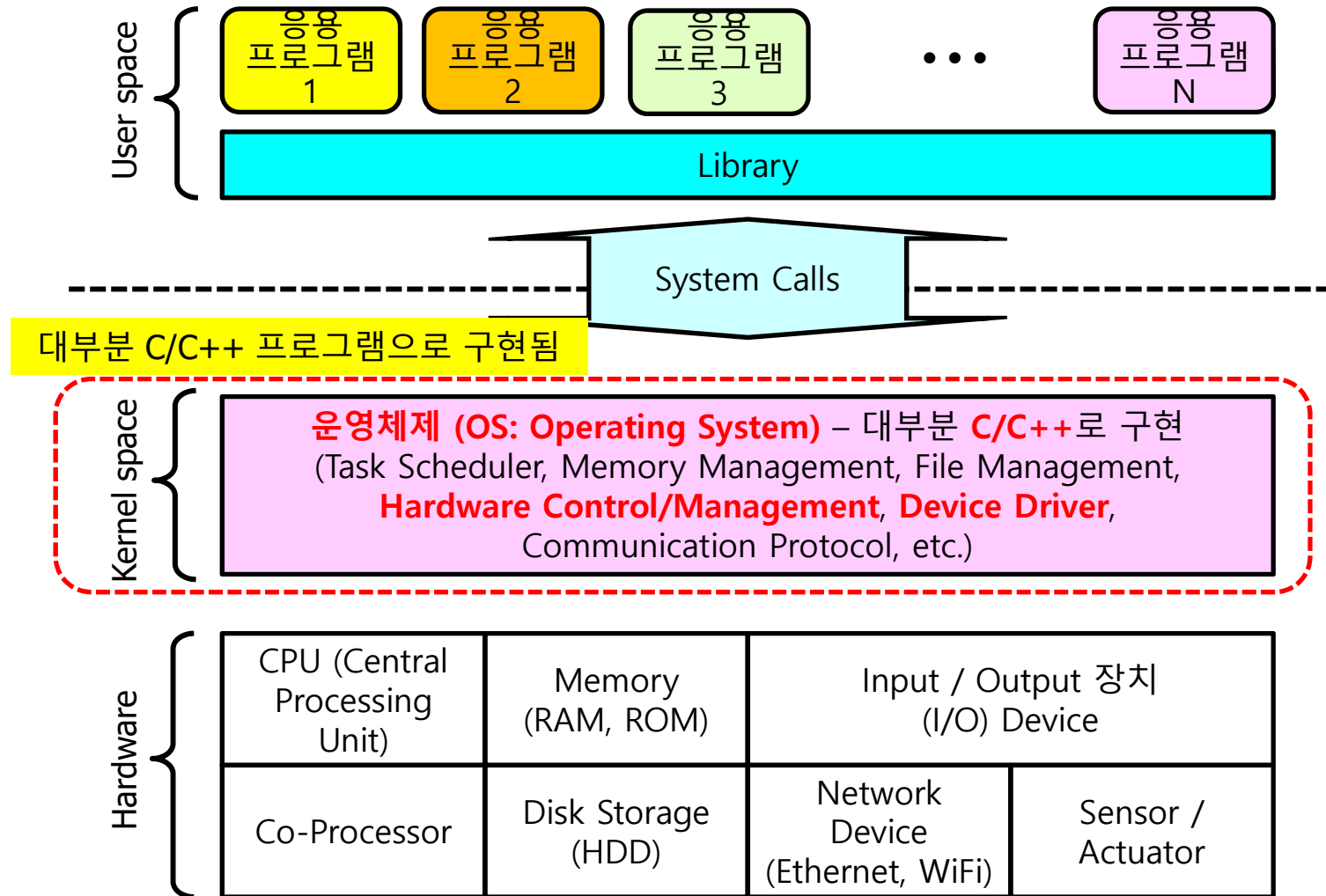
산업혁명	시기	주요 변화
1차	18세기	증기 기관 (steam engine)의 발명으로 사람과 동물의 힘보다 더욱 더 큰 힘을 효율적으로 발생시킬 수 있었으며, 영국에서는 이를 활용하여 섬유생산을 획기적으로 증대시킴
2차	19세기 ~ 20세기 초반	전기 에너지 생산 및 공급 기술이 개발되고, 표준화된 모듈을 컨베이어 벨트에서 조립하는 대량 생산 기술이 개발됨
3차	20세기 후반	컴퓨터와 인터넷 기술의 개발로 과학 기술 분야의 대용량 데이터 처리가 가능하게 되었으며, 서로 다른 지역에 떨어져 있는 사람과 장치간의 협동 작업이 가능하게 되어 기술 발전이 더욱 가속화 됨
4차	2015년 ~ 현재	사물인터넷, 빅데이터, 인공지능, CPS (Cyber Physical System) 기술의 개발로 사람, 사물, 공간을 초연결, 초지능화 시킴

## ◆ 제4차 산업혁명의 핵심 기술

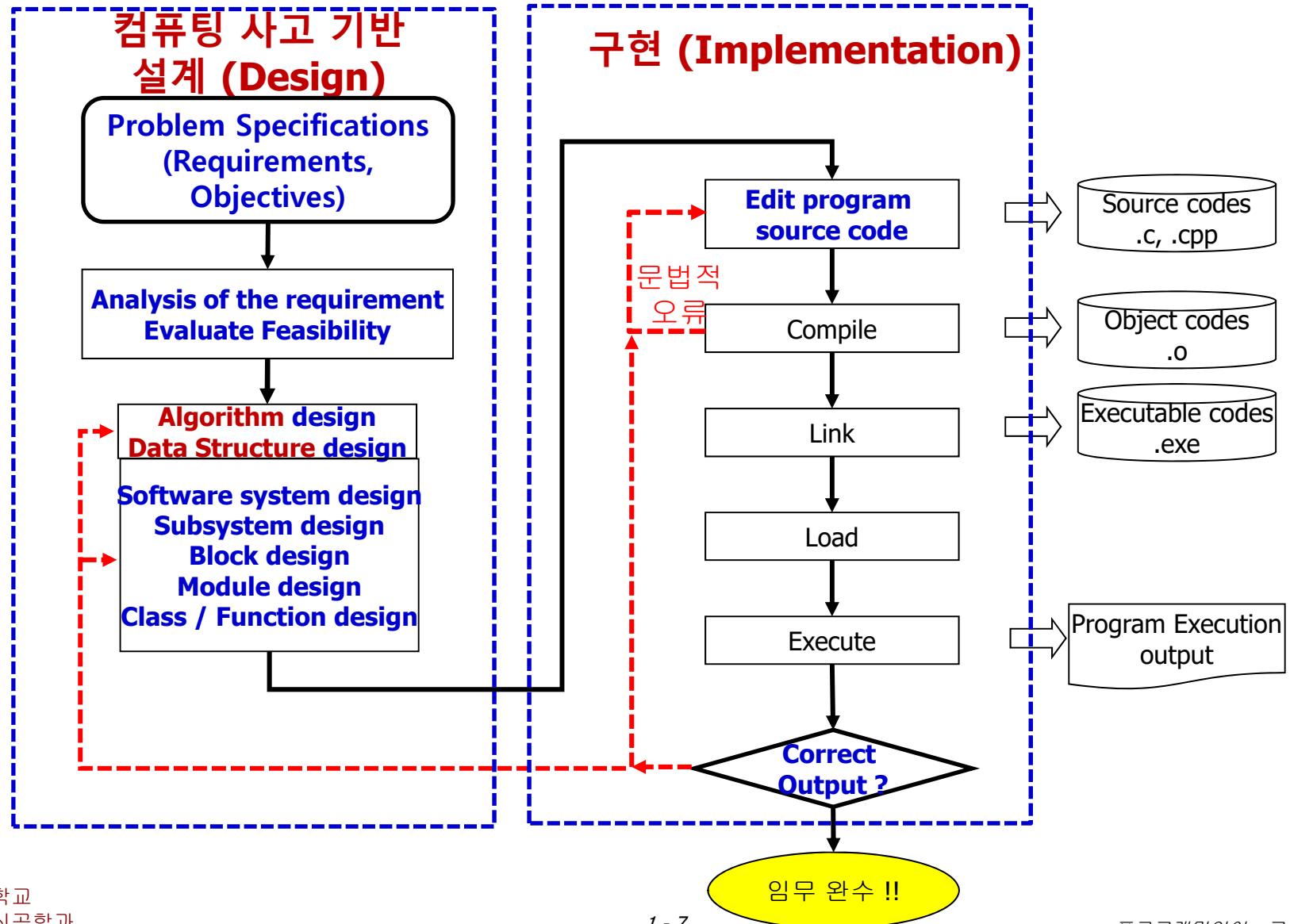
- 사물인터넷 (Internet of Things, IoT)
- 빅데이터 (Big Data)
- 인공지능 (Artificial Intelligence, AI)

➔ 모두 소프트웨어가 핵심이 되는 기반 기술임 !

# 컴퓨터 구조 – 하드웨어, 운영 체제 및 응용 소프트웨어



# 소프트웨어 개발 과정



# 알고리즘과 자료구조

- ◆ 모든 소프트웨어 개발에서는 효율적인 알고리즘 (algorithm)과 이 알고리즘에 적합한 자료구조 (data structure)를 사용하여야 우수한 성능을 발휘할 수 있음
- ◆ 알고리즘의 예
  - sorting (정렬): selection sorting, quick sorting
  - search (탐색): depth-first search, breadth-first search
  - shortest path, minimum spanning tree
- ◆ 자료구조의 예
  - 단순 배열 (simple array)
  - 구조체 (structure)
  - 구조체 배열 (array of structure)
  - 자기참조 구조체 (self-referential structure)
  - Linked List, Binary Tree
  - Heap, priority queue
  - Hash Map, Skip list
  - Graph

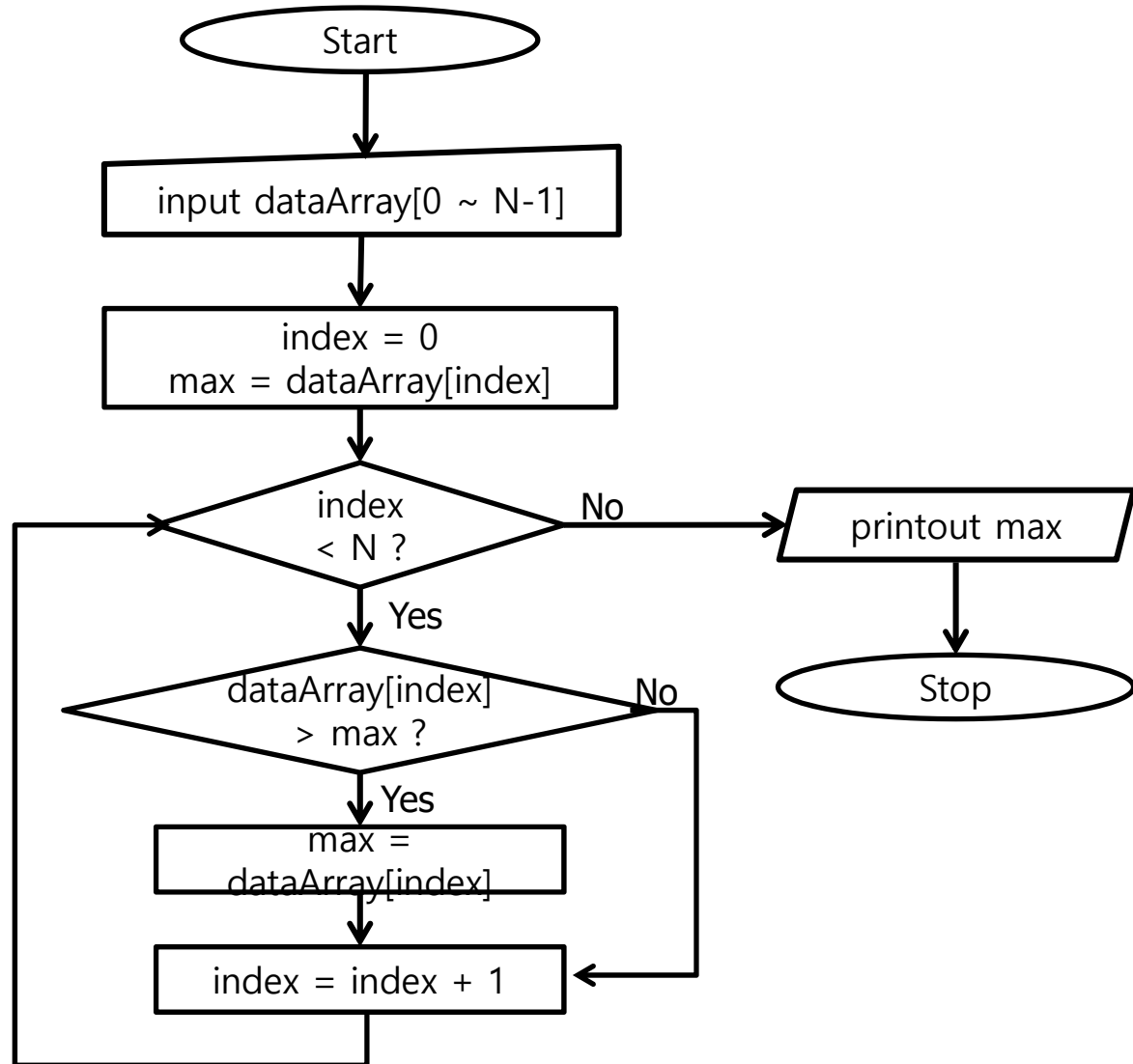


# 알고리즘의 개발

- ◆ 프로그램/소프트웨어 개발에서 가장 핵심적인 부분
- ◆ 어떤 절차와 단계를 밟아서 어떤 순서로 작업을 처리할 것인지를 설계
- ◆ 알고리즘은 프로그래밍 언어와는 독립적임
- ◆ 알고리즘은 원하는 결과를 얻기 위하여 밟아야 하는 절차와 단계에 집중적으로 초점을 맞추는 것
- ◆ 알고리즘의 표현
  - 순서도 (flow chart)
  - 유사 코드 (pseudo code)



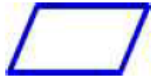







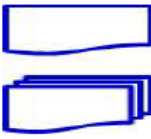

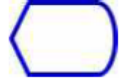

# 순서도 (flow chart)

- ◆ 프로그램에서의 논리순서 또는 작업순서를 그래픽으로 표현하기 위한 형식
- ◆ 알고리즘이 복잡하면 그래픽으로 표현하기가 힘들어진다.



# 순서도에서 사용되는 기호와 의미

## ◆ Flow Chart Symbols

순서도 기호	의미	순서도 기호	의미
	단말(단자): 알고리즘의 시작과 끝을 표시		순서 흐름선 (flow line)
	데이터의 입력과 출력		판단, 분기
	데이터의 수동입력		지연 (delay)
	준비		논리집합, 가산집합
	데이터 처리		정렬, 분류
	미리 준비된 처리 (서브 루틴)		대조
	문서, 서류		다른 페이지 연결자
	디스플레이		페이지 내 연결자

# 유사코드 (Pseudo Code)

## ◆ 유사코드 (Pseudo code)

- 프로그래밍언어(C, C++, Java, Python 등)의 모든 문법을 정확하게 지키지 않고,  
전체 알고리즘의 기본 구조와 기능을 간략화된 표현으로 설명
- Simplified list of instructions to show the overall process of algorithm
- not following the details of programming language
- the skeleton of the algorithm should be shown

## ◆ Pseudo code (1)

### Procedure FindMax(dataArray, N)

```
1:  //input argument: int dataArray[0..N-1];  
    // array of data with N elements  
    // output result: maximum value of the given data array  
2:  int max;  
3:  int index;  
4:  index = 0; //initialize index  
5:  max = dataArray[index];  
6:  while (index < N)  
7:  {  
8:      if (max < dataArray[index])  
9:          max = dataArray[index];  
10:     index = index + 1;  
11: } // end while  
12:  
13: printout max;  
14: end // end of Algorithm FindMax()
```

## ◆ Pseudo code (2)

### Procedure FindAvg(dataArray, N)

```
1:  //input argument: int dataArray[0..N-1];  
    // array of data with N elements  
    // output result: maximum value of the given data array  
2:  int count;  
3:  double sum, avg;  
4:  count = 0; //initialize index  
5:  sum = 0.0;  
6:  while (count < N)  
7:  {  
8:      sum = sum + dataArray[count];  
9:      count = count + 1;  
10: } // end while  
11: avg = sum / count;  
12: printout avg;  
13:  
14: end // end of Algorithm FindAvg()
```

# 소스프로그램 작성 (Source Program Coding)

- ◆ 알고리즘의 각 단계를 프로그래밍 언어를 이용하여 기술
- ◆ 어떤 프로그래밍 언어로도 가능
- ◆ 알고리즘을 프로그래밍 언어의 문법에 맞추어 기술한 것을 *소스 프로그램(source program)*
- ◆ 소스 프로그램은 주로 텍스트 에디터나 통합 개발 환경을 이용하여 작성

(Q) 알고리즘 개발과 코딩 중 어떤 것이 더 어려울까?

(A) 알고리즘 개발이 더 창의적인 작업이고 더 어렵다

# 주석 (Comment)

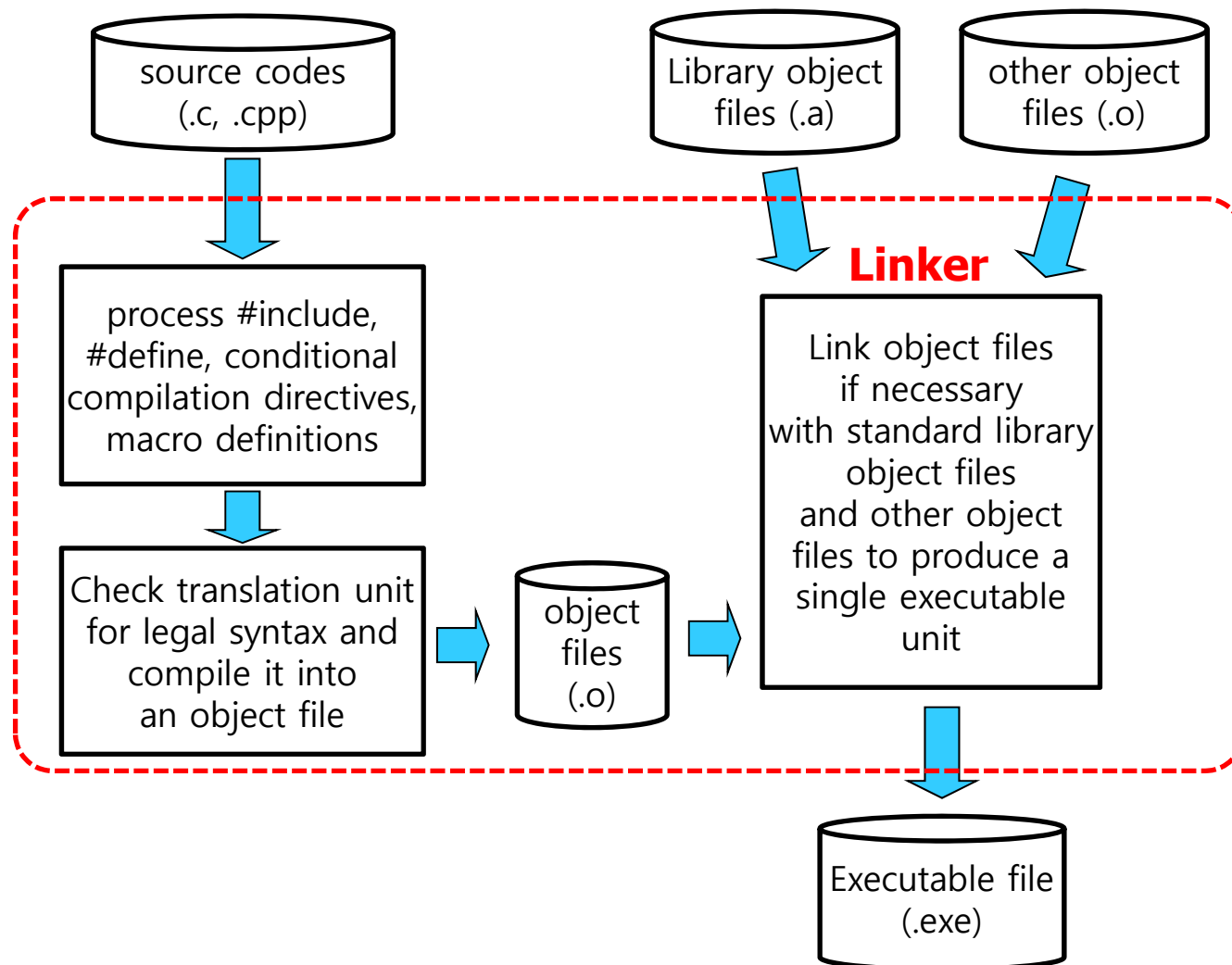
## ◆ 프로그램 소스 코드 작성에서의 기본 주석 표기내용

```
/**
 * 파일명: "automaticTempControl.c" or "xxx.h", or "yyy.cpp"
 * 프로그램의 목적 및 기본 기능:
 *   - 이 프로그램은 사물인터넷의 온도 센서를 읽고, 사전에 설정된 온도가 유지될 수 있도록
 *     히터/에어컨을 동작시키는 .....
 *
 * 프로그램 작성자: 홍 ○○ (2021년 3월 3일),
 * 최종 Update : Version 2.0, 2021년 3월 5일 (김○○).
 *
 * =====
 * 프로그램 수정/보완 이력
 * =====
 * 프로그램 수정/보완작업자   일자   수정/보완 내용
 * 홍 ○○      2021/03/03   v1.0   온도센서 읽기, 히터 동작, 에어컨 동작 모듈 완성
 * 정 ○○      2021/03/04   v1.1   GUI 기능 보완
 * 김 ○○      2021/03/05   v2.0   온도 센서 읽기의 데이터 오류 발생 여부 확인 기능 추가
 * =====
 */
```



# Compile과 Link, 실행코드 생성

## ◆ 소스 프로그램을 목적 프로그램으로 변환하는 작업



# 실행 (execution)과 오류 디버깅 (Debugging)

◆ 실행 파일: 실행 파일은 윈도우즈에서는 .exe라는 확장자

◆ 실행 시간 오류(run time error):

- 0으로 나누는 것
- 잘못된 메모리 주소에 접근하는 것

◆ 논리 오류(logical error):

- 문법은 틀리지 않았으나 논리적으로 정확하지 않는 것
- (예)

```
int age;

input age from console input;
if ( age == 13)
{
    print ("Your age is 13!
    Congratulate to be
    a teen ager");
}
```

(a) 정상적인 경우

```
int age;

input age from console input;
if ( age = 13)
{
    print ("Your age is 13!
    Congratulate to be
    a teen ager");
}
```

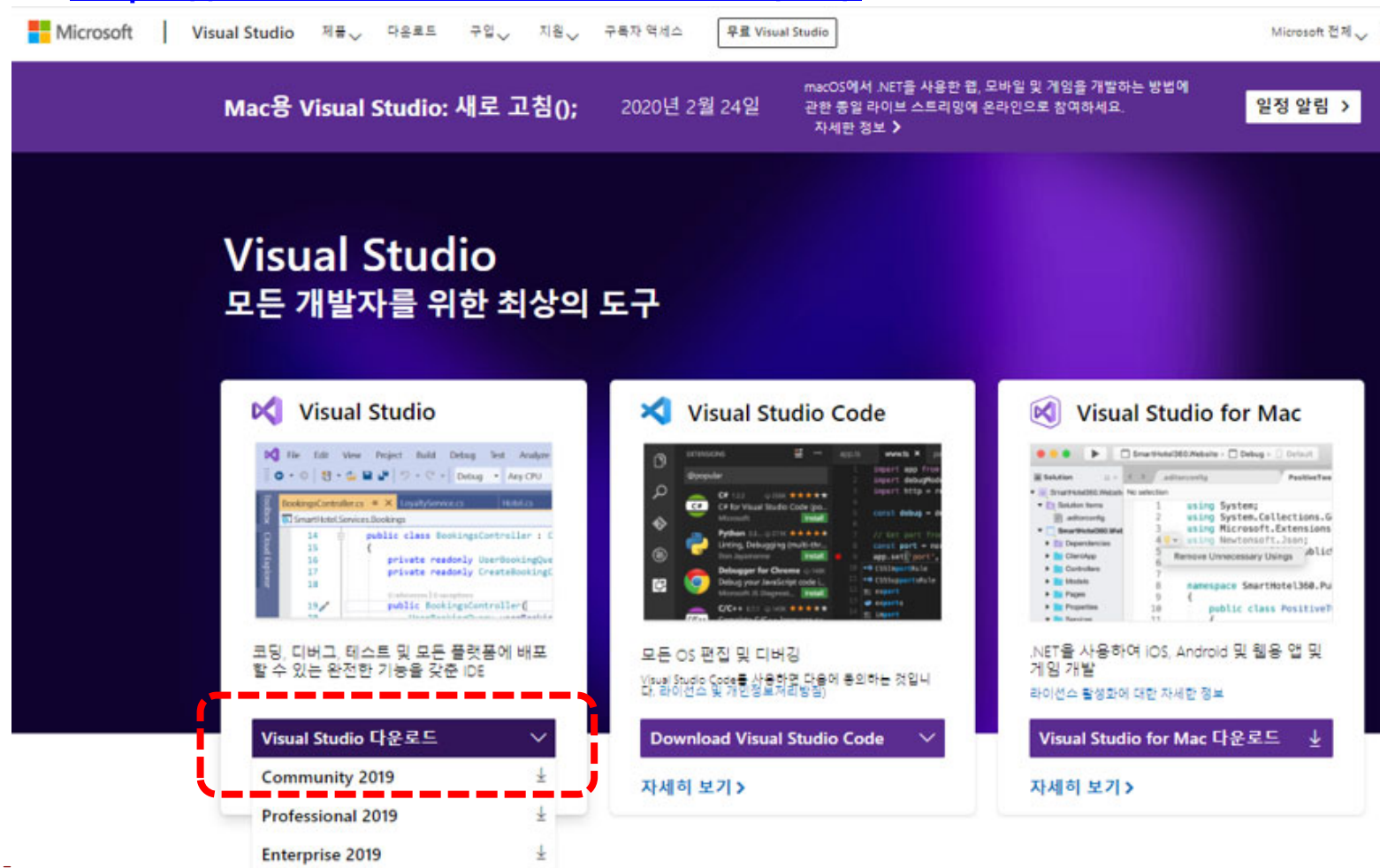
(b) 논리적 오류가 발생하는 경우

# **프로그램 통합 개발환경 준비와 C 프로그램 기본 구조**

# 프로그램 통합 개발 환경 설치

## ◆ Visual Studio Community 2019 (무료다운로드)

- <https://visualstudio.microsoft.com/ko/>



# C/C++ 프로젝트 만들기

## ◆ Visual Studio Community 2019에서의 C/C++ 새 프로젝트 만들기



# C/C++ 프로젝트 구성

## 새 프로젝트 구성

빈 프로젝트   C++   Windows   콘솔

프로젝트 이름(N)

MyFirstCprog

위치(L)

C:\MyC\_Cpp\_Progs\2020-2 Book (C-Prog Visual Studio 2019)\ch 1\

...

솔루션 이름(M) ⓘ

MyFirstCprog

☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

뒤로(B)   만들기(C)



# 원의 면적 계산 프로그램

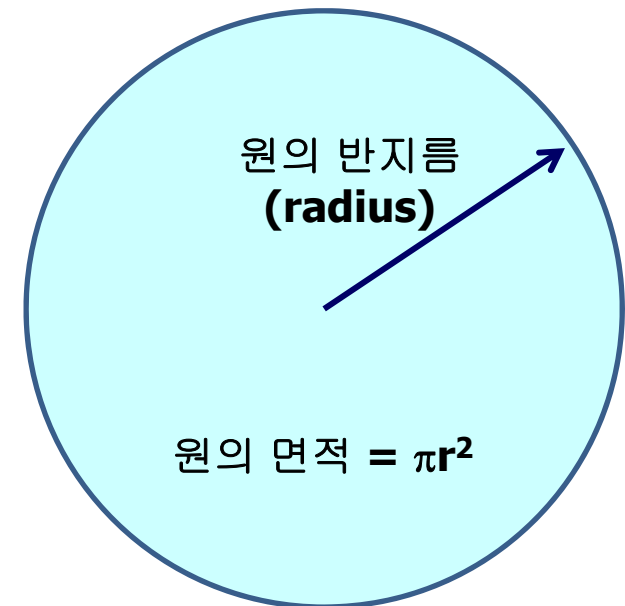
## ◆ 원의 반지름이 주어졌을 때, 면적 계산하기

- 원의 면적 =  $3.14159 \times \text{반지름} \times \text{반지름}$

### Algorithm AreaOfCircle()

```
1: // input arguments: none
2: double radius; // local variable
3: double area; // local variable

4: input radius; // input data
5: area = 3.14159 * radius * radius;
6: printout radius and area;
7: end
```



# 원의 면적 프로그램 소스코딩

```
/* myFirstCprog.cpp */
/*
 * Programmed on Jan. 2, 2021
 * Author : YTKim
 * Sample C program to demonstrate basic C programming
 */
#include <stdio.h>
#define PI 3.141592

int main()
{
    double radius, area, circum; // 반지름, 면적, 둘레

    printf("원의 반지름을 실수형으로 입력하세요 : ");
    scanf("%lf", &radius);
    area = PI * radius * radius;
    circum = 2.0 * PI * radius;

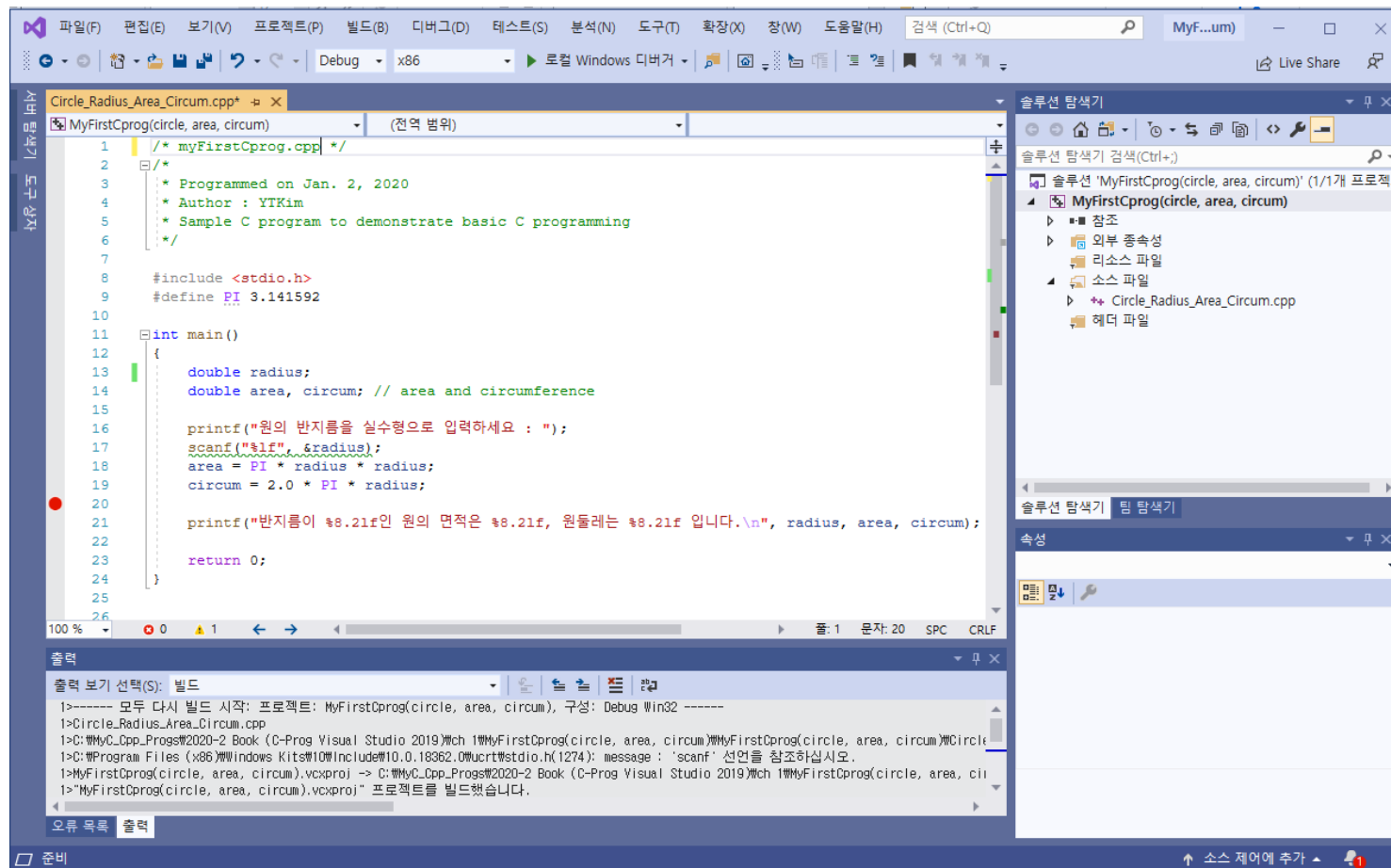
    printf("반지름이 %8.2lf인 원의 면적은 %8.2lf, 원둘레는 %8.2lf 입니다.\n", radius, area, circum);

    return 0;
}
```



# C/C++ 프로그램 소스코드 작성

## ◆ Visual Studio 2019 Community 환경에서의 C/C++ 소스코드 작성



# 주석(comment)

```
/* 원의 면적을 계산하는 프로그램 */  
#include <stdio.h>  
  
int main(void)  
{  
    ...  
    ...  
    ...  
}
```

주석은 코드를 설명  
하는 글입니다.



주석 (comment)

## 3가지 방법의 주석 (comment) 표시

### ◆ 한 줄로 표시되는 주석

```
/* 한 줄로 된 주석 */
```

### ◆ 여러 줄로 표시되는 주석

```
/* 여러  
   줄로  
   된 주석  
*/
```

### ◆ 줄의 중간부터 끝까지의 주석

```
// 여기서부터 이 줄의 끝까지 주석
```

# 들여쓰기 (indentation)

## ◆ 들여쓰기(indentation)

- 같은 수준에 있는 문장들을 왼쪽 끝에서 몇 자 안으로 들여 쓰기를 하여 쉽게 이해 할 수 있게 하는 것
- 들여쓰기를 하여도 블록이 구분되지는 않음

프로그램의  
주요기능과  
작성일자,  
작성자,  
수정 내용  
등을  
주석으로  
설명

```
/* myFirstCprog.cpp */  
/*  
 * Programmed on Jan. 2, 2020  
 * Author : YTKim  
 * Sample C program to demonstrate basic C programming  
 */
```

```
#include <stdio.h>  
#define PI 3.141592
```

```
int main()  
{
```

```
    double radius;  
    double area, circum; // area and circumference
```

```
    printf("원의 반지름을 실수형으로 입력하세요 : ");
```

```
    scanf("%lf", &radius);
```

```
    area = PI * radius * radius;
```

```
    circum = 2.0 * PI * radius;
```

```
    printf("반지름이 %8.21f인 원의 면적은 %8.21f, 원둘레는 %8.21f 입니다.\n", radius, area, circum);
```

```
    return 0;  
}
```

프로그램  
모듈의  
블록을  
들여쓰기로  
구분

프로그램의  
각 블록별 주요  
기능을 간단하게 설명



# 주석과 들여 쓰기가 없다면..

```
#include <stdio.h>
int main(void) { int x; int y; int sum;
x = 100; y = 200; sum = x + y;
printf("두수의 합: %d", sum); return 0;
}
```

실행은 되지만 무슨 처리를 하고 있는 프로그램인지 알기가 힘들고 또한 들여쓰기가 안되어 있어서 같은 수준에 있는 문장들을 구분하기 힘듭니다.

## 프로그램 소스코드의 **가독성 (readability)**

- 프로그램 소스코드를 읽고 쉽게 이해할 수 있도록 작성하여야 함
- 프로그램 소스코드의 가독성을 높이기 위하여 주석 (comment)과 들여쓰기를 잘 사용하여야 함



# 전처리기 (preprocessor)

```
#include <stdio.h>
```

- ◆ **#**기호로 시작
- ◆ 헤더 파일 **stdio.h**를 소스 코드 안에 포함
- ◆ **stdio.h**는 표준 입출력에 대한 라이브러리 함수의 정의가 들어 있다.

# 전처리기

```
/* 첫번째 프로그램 */
```

```
#include <stdio.h>
```

```
int main(void)
{
    printf("Hello World!");
    return 0;
}
```

hello.c

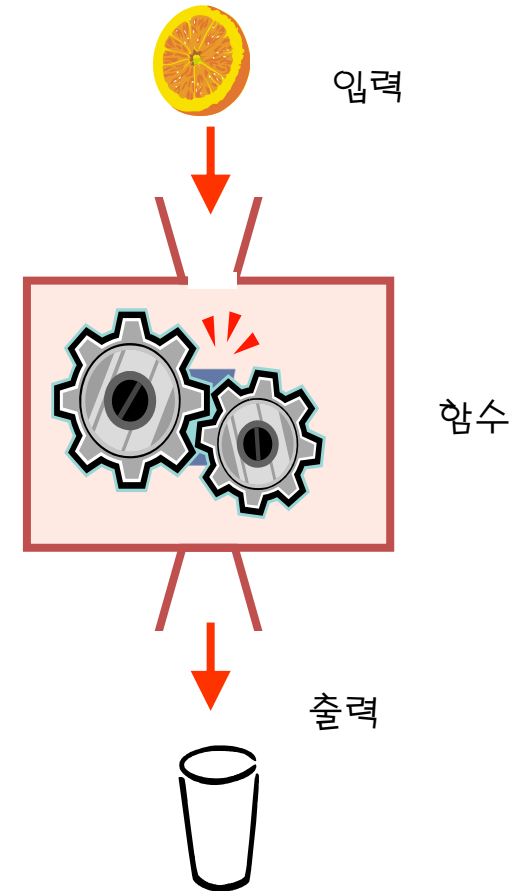
```
// stdio.h
...
int printf(char *,...);
...
```

stdio.h



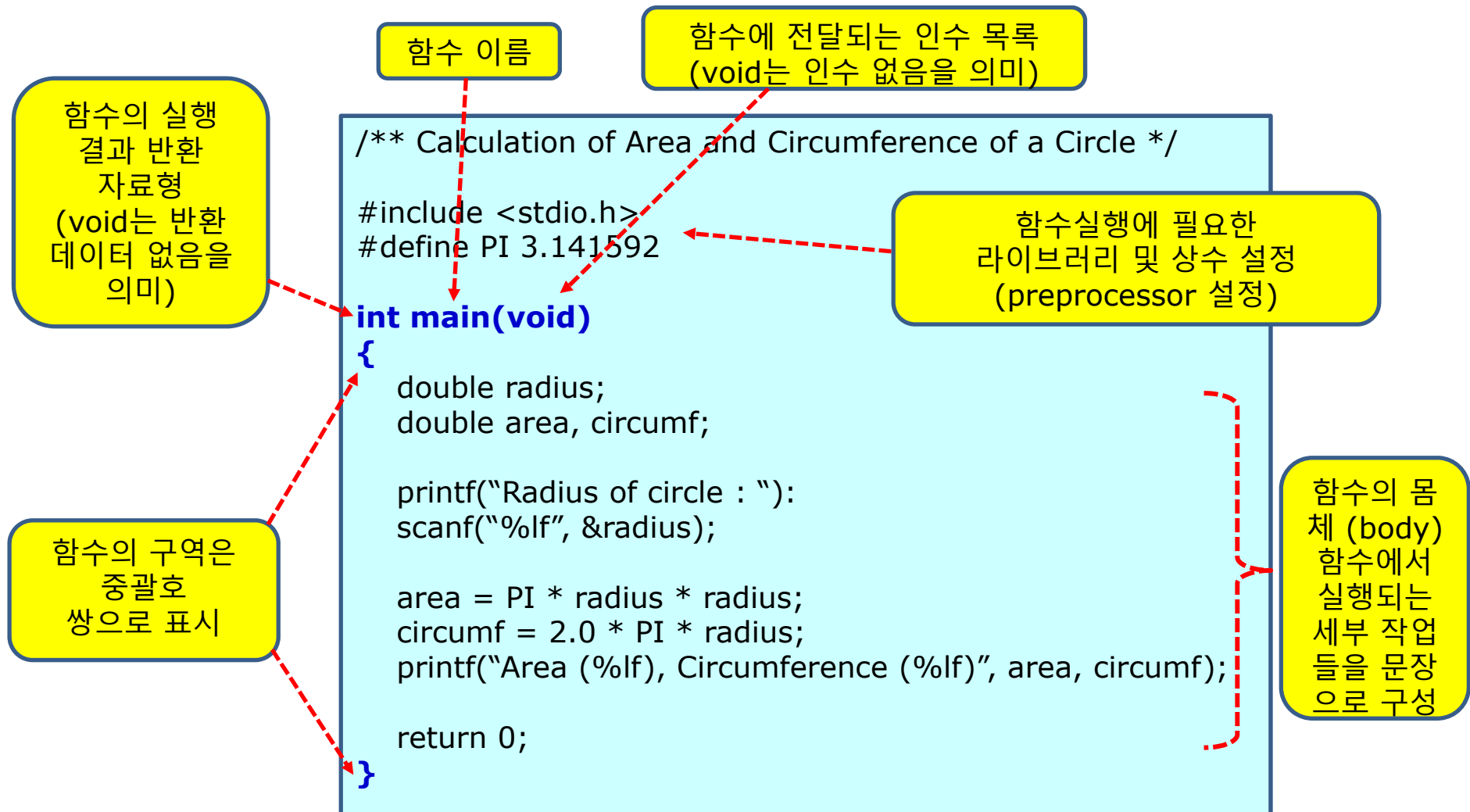
# 함수 (function)

- ◆ **함수(function)**: 특정 기능을 수행하는 처리 단계들을 중괄호({})로 묶어서 이름을 붙인 것
- ◆ 함수는 프로그램을 구성하는 기본적인 단위(부품)
- ◆ 각 함수에는 전달되는 인수(argument)들의 자료형이 명시됨
- ◆ 각 함수의 반환 자료형도 명시됨
- ◆ 전달되는 인수가 없거나, 반환되는 데이터가 없는 경우 **void**로 표시
- ◆ **main()** 도 함수 중 하나





# 함수의 구조



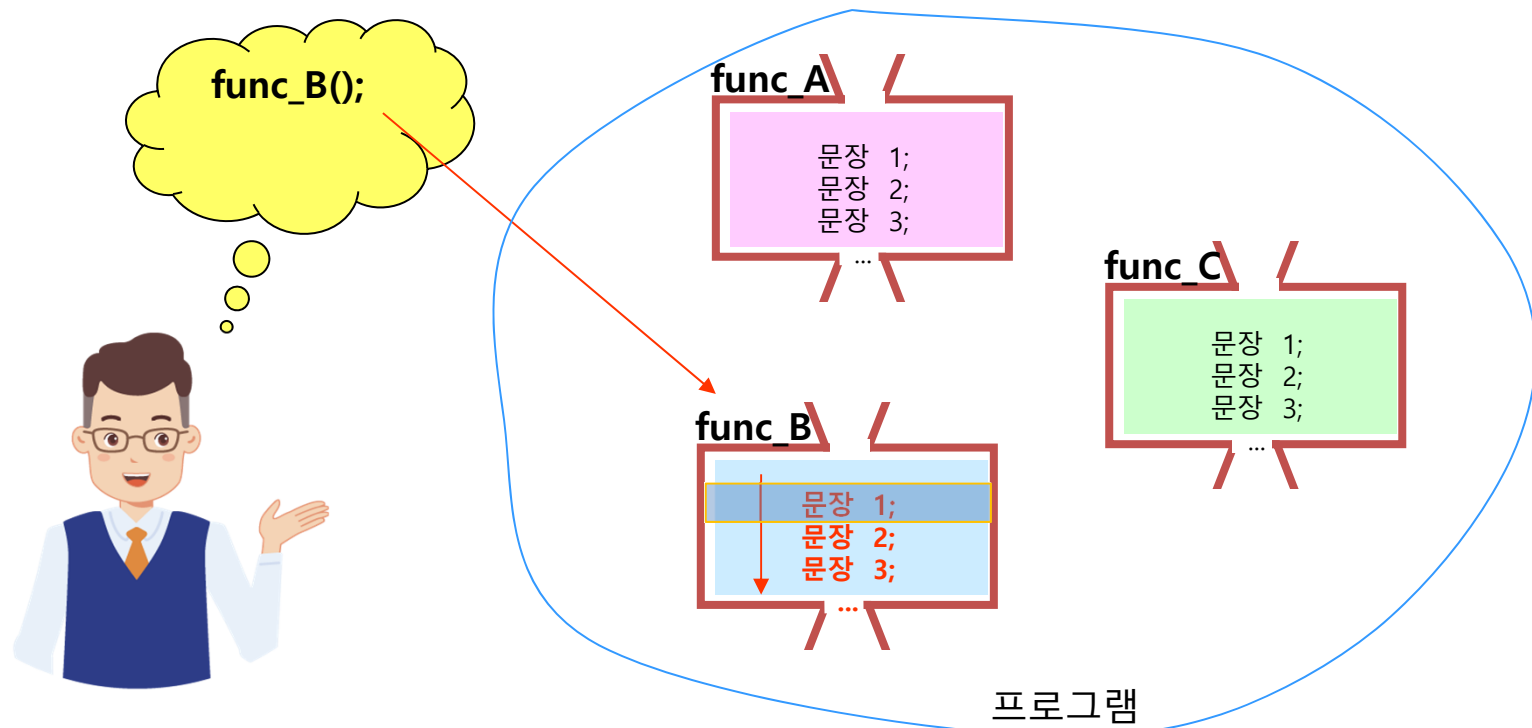
# 함수 호출 (function call)

**Q) 함수 안에 있는 문장들은 언제 실행되는가?**

A) 함수가 호출되면 실행된다.

**Q) 함수 호출은 어떻게 하는가?**

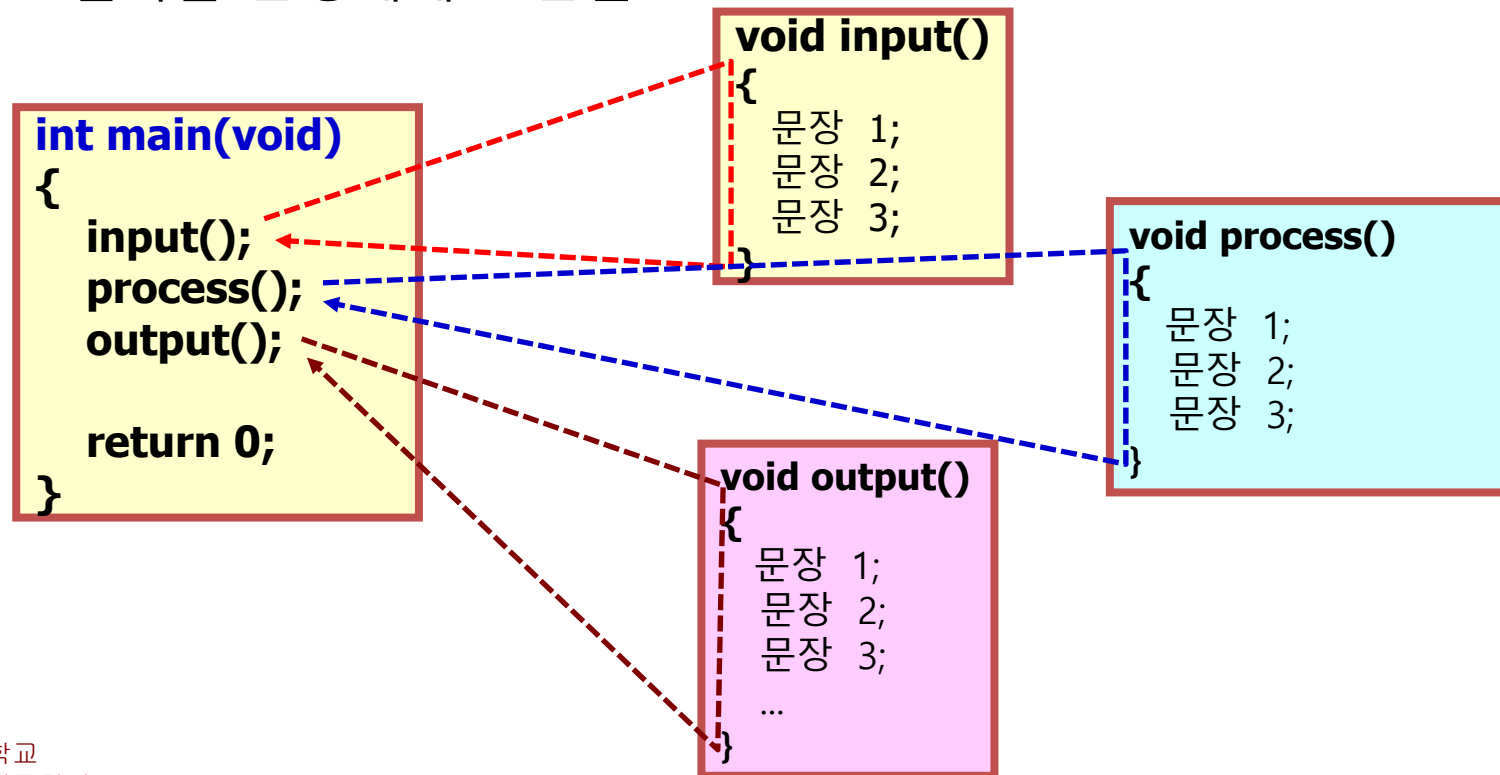
A) 함수의 이름을 적어주면 된다.



# 함수 호출

## ◆ C 프로그램의 많은 함수 중에서 가장 먼저 실행되는 것은?

- 운영체제 (Operating System)이 main() 함수를 호출하여 프로그램 실행이 시작됨
- 다른 함수들은 main()으로부터 직간접적으로 호출된다.
- main() 함수가 종료되면 return 명령어를 사용하여 프로그램 실행 결과를 운영체제로 반환



# C 프로그램 문장 (statement)

## ◆C/C++ 프로그램의 문장

- 함수는 여러 개의 문장으로 이루어진다.
- 문장들은 순차적으로 실행된다.
- 각 문장은 ; (세미콜론)으로 끝나야 한다.

```
/** Calculation of Area and Circumference of a Circle */  
#include <stdio.h>  
#define PI 3.141592  
  
int main(void)  
{  
    double radius, area, circumf;  
  
    printf("Radius of circle : ");  
    scanf("%lf", &radius);  
  
    area = PI * radius * radius;  
    circumf = 2.0 * PI * radius;  
    printf("Area (%lf), Circumference (%lf)", area, circumf);  
  
    return 0;  
}
```

프로그램 소스코드의 문장들은 차례대로 실행된다. 함수 호출이 있는 경우, 그 호출된 함수를 실행하고 난 후, 호출한 함수로 돌아와서 나머지 문장을 계속 실행한다.



# return 문장

## ◆ return statement

- 함수의 실행 결과를 반환
- 실행 결과의 반환이 없는 경우 return이 생략 될 수 있음
- main() 함수의 return은 그 프로그램의 실행 종료 상태를 운영체제 (OS) 에게 알려 주는 기능을 수행함
  - 0 또는 1 : 정상적인 종료
  - -1 : 오류 발생에 따른 비정상적인 종료

**변수 (variable), 상수 (constant)와  
연산 (operation)**

# 변수 (variable)

## ◆ 변수 (variable)란 ?

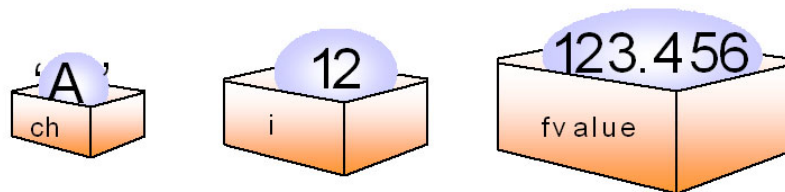
- 프로그램이 사용하는 데이터를 일시적으로 저장할 목적으로 사용하는 메모리 공간

## ◆ 변수의 자료형 (data type)

- 변수에 담기는 데이터의 종류에 따라 여러가지 자료형이 존재
- char (문자)
- int (정수, integer)
- double (실수, 부동 소숫점, double precision floating point)

## ◆ 변수의 예

- `int radius;` // 원의 반지름을 저장하는 변수
- `double area;` // 원의 둘레를 저장하는 변수



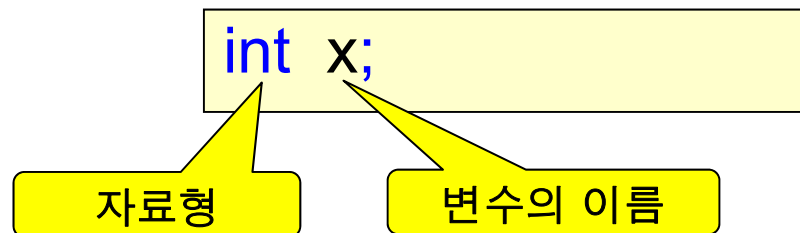
# 변수 선언

## ◆ 변수 선언

- 컴파일러에게 어떤 타입의 변수가 사용되는지를 미리 알리는 것
- 변수의 자료형과 변수의 이름 (식별자)를 선언

## ◆ 메모리 할당

- 프로그램 실행 단계에서 변수를 위한 메모리 할당
- 자료형에 따라 할당되는 메모리의 양이 달라짐

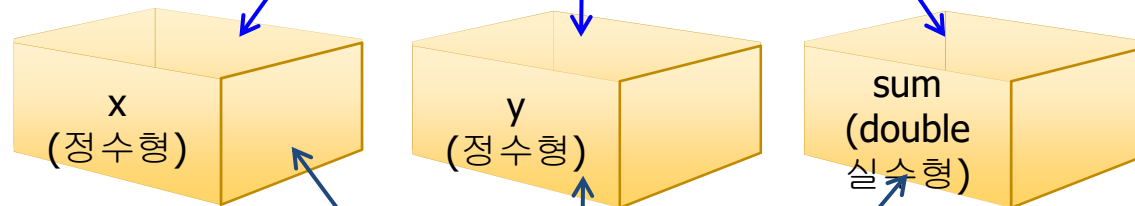




# 변수 선언

```
int x; // 첫번째 정수를 저장하는 변수  
int y; // 두번째 정수를 저장하는 변수  
double sum; // 두 정수의 합을 저장하는 변수
```

각 변수는 지정된 자료 유형 (type)에 따라  
정수 또는 실수 를 저장 할 수 있다.



메모리 공간에 변수 가 만들어지고  
이름이 붙여진다.

# 변수의 이름

## ◆ 식별자(identifier)

- 변수나 함수의 이름

## ◆ 식별자를 만드는 규칙

- 식별자는 영어의 대소문자, 숫자, 밑줄 문자 \_로 이루어진다.
- 식별자는 숫자로 시작할 수 없다.
- 대문자와 소문자를 구별하며 C 언어의 키워드와 똑같은 이름은 허용되지 않는다.

## ◆ 식별자의 예:

- s, s1, student\_number: 올바른 식별자
- \$s, 2nd\_student, int: 잘못된 식별자

# 자료형 (data type)

◆ **자료형(data type):** 변수가 저장할 데이터가 정수인지 실수인지, 아니면 또 다른 어떤 데이터인지를 지정하는 것

기본 자료형		설 명	비트 수	값의 범위
char (문자형)	(signed) char	문자 및 정수	8	-128 ~ 127
	unsigned char	문자 및 부호 없는 정수	8	0 ~ 255
int (정수형)	(signed) short	short형 정수	16	-32768 ~ 32767
	unsigned short	부호 없는 short형 정수	16	0 ~ 65535
	(signed) int	정수 (integer)	32	-2147483648 ~ 2147483647
	unsigned int	부호 없는 정수	32	0 ~ 4294967295
	(signed) long	long형 정수	32	-2147483648 ~ 2147483647
	unsigned long	부호 없는 long형 정수	32	0 ~ 4294967295
floating point (부동소수점)	float	단일 정밀도 부동소수점	32	1.2E-38 ~ 3.4E38
	double	두 배 정밀도 부동소수점	64	2.2E-308 ~ 1.8E308
	long double	두 배 정밀도 부동소수점	64	2.2E-308 ~ 1.8E308

# 상수 (Constant)

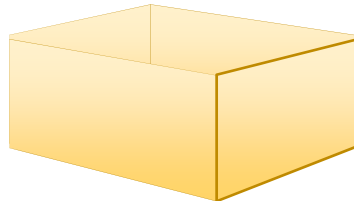
- ◆ 상수(constant): 그 값이 프로그램이 실행하는 동안 변하지 않는 수

PI = 3.141592;

x = 200;

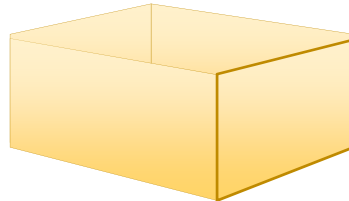
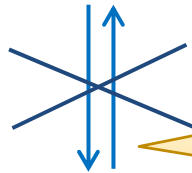
상수

200



변수

3.14



상수

변수는 실행도중에 값을 변경할 수 있으나 상수는 한번 값이 정해지면 변경이 불가능합니다.



# 수식 (expression)

## ◆ 수식(expression):

- 연산자 (operator)와 피연산자 (operand)로 구성된 식

## ◆ 수식의 실행에서 계산 결과값을 등호 왼쪽의 변수에 저장한다.

```
sum = a + b;  
y = x*x - 2*x + 3;
```

x가 3일 때 수식  
 $x^2 - 2x + 3$ 의 값을  
계산하라.



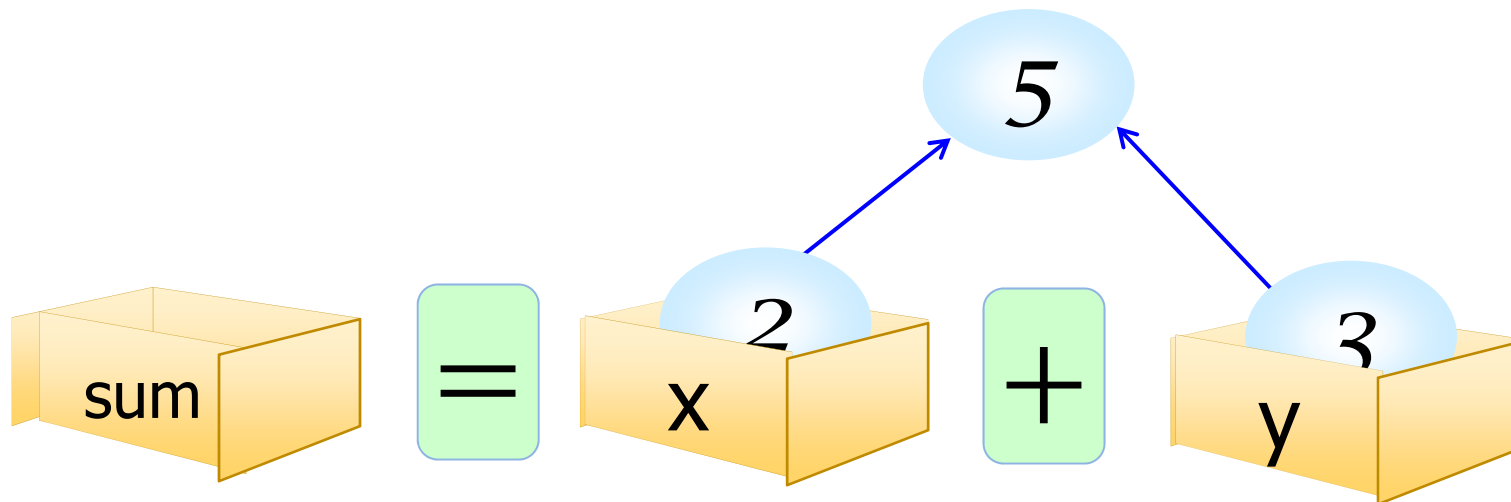
```
int x, y;  
  
x = 3;  
y = x * x - 2 * x + 3;  
printf("%d\n", y);
```

## 산술 연산

연산	연산자	C 수식	수학에서의 기호
덧셈	+	$x + y$	$x + y$
뺄셈	-	$x - y$	$x - y$
곱셈	*	$x * y$	$xy$
나눗셈	/	$x / y$	$x / y$
나머지	%	$x \% y$	$x \bmod y$

# 산술 연산

```
sum = x + y;
```

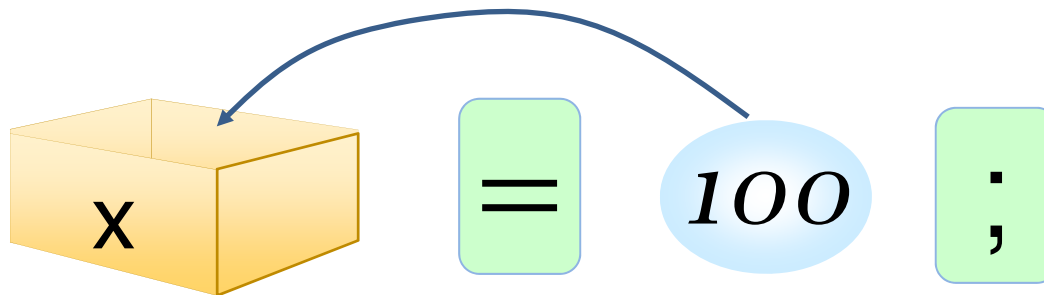


# 대입 연산 (Assignment Operation)

```
x = 100;
```

◆ 대입 연산(assignment operation): 변수에 값을 저장하는 연산

◆ 대입 연산 = 배정 연산 = 할당 연산

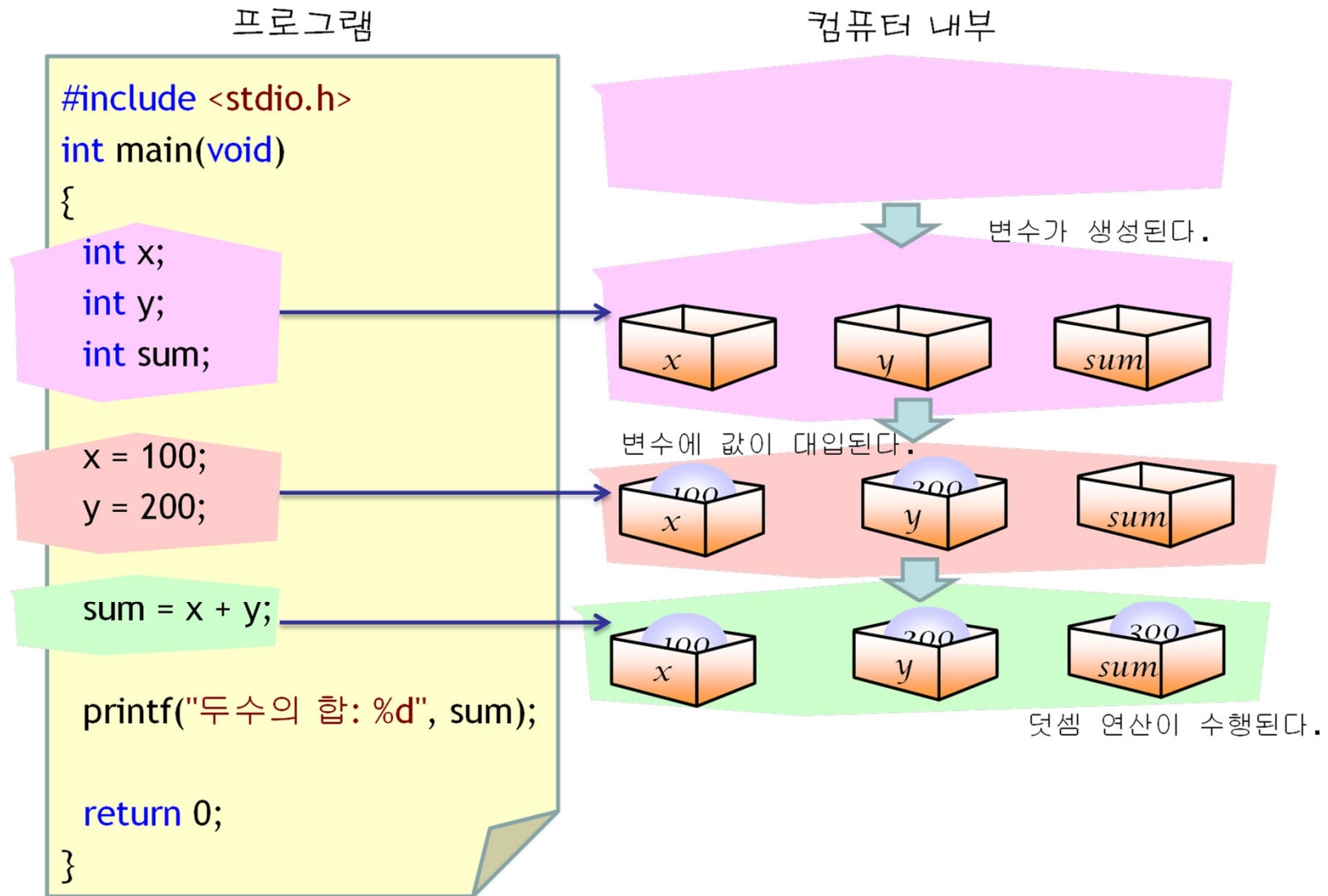


= 연산자는  
변수에 값을  
저장합니다.





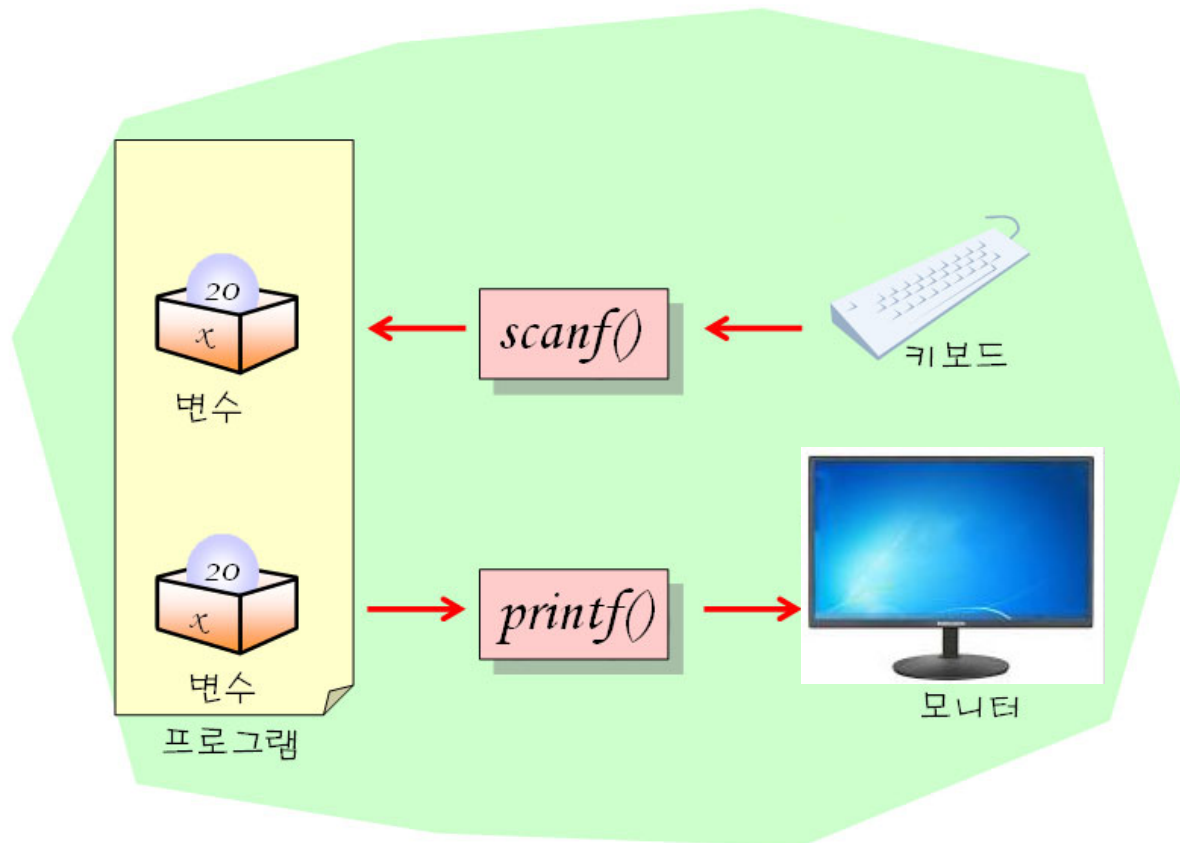
# 프로그램의 실행과 데이터 정리



## **프로그램의 입력과 출력**

# scanf()와 printf()

- ◆ **scanf():** 지정된 형식에 따라 표준 입력 장치 (키보드)로부터 데이터를 입력 받기 위한 표준 입력 라이브러리 함수
- ◆ **printf():** 지정된 형식에 따라 표준 출력장치 (모니터)에 출력을 하기 위한 표준 출력 라이브러리 함수



# 문자열 (string) 출력

- ◆ 문자열(string): "Hello World!\n"와 같이 문자들을 여러 개 나열한 것

```
printf("Hello World!\n");
```

Hello World !

printf()



# 변수값 출력

```
printf("두수의 합: %d", sum);
```



형식 지정자의 개수와  
변수의 개수, 순서는  
같아야 합니다.



# printf() 함수의 형식 지정자

◆ 형식 지정자: printf()에서 값을 출력하는 형식을 지정한다.

형식 지정자	의미	예	실행 결과
%d	10진 정수로 출력	printf("%d\n", 10);	10
%f	실수 (float)로 출력	printf("%f\n", 3.14);	3.140000
%lf	실수 (double)로 출력	printf("%lf\n", 2.5);	2.500000
%c	문자로 출력	printf("%c\n", 'a');	a
%s	문자열로 출력	printf("%s\n", "Hello");	Hello

# 여러 개의 변수 값들을 한 줄에 출력하기

형식 제어 문자열

```
printf( "학번 %d 의 성적은 %f \n" , st_id , score );
```

학번 23의 성적은 3.99

형식 지정자 의 개수와  
변수의 개수와 순서는  
같아야 한다.

# 덧셈 및 평균계산 프로그램

```
/** Addition and Average */

#include <stdio.h>

int main(void)
{
    int x;          // 첫 번째 정수를 저장할 변수
    int y;          // 두 번째 정수를 저장할 변수
    double sum;     // 2개의 정수의 합을 저장할 변수
    double avg;     // 2개의 정수의 평균을 저장할 변수

    printf("첫 번째 숫자를 입력하십시오:"); // 입력 안내 메시지 출력
    scanf("%d", &x);          // 하나의 정수를 받아서 x에 저장

    printf("두 번째 숫자를 입력하십시오:"); // 입력 안내 메시지 출력
    scanf("%d", &y);          // 하나의 정수를 받아서 x에 저장

    sum = x + y;              // 변수 2개를 더하고, 결과를 sum에 저장
    avg = sum / 2.0;         // 변수 2개의 평균을 계산하여 avg에 저장

    printf("두 수의 합 (%lf), 평균(%lf)\n", sum, avg);
    // sum의 값을 10진수 형태로, 평균은 실수 (double) 형태로 출력

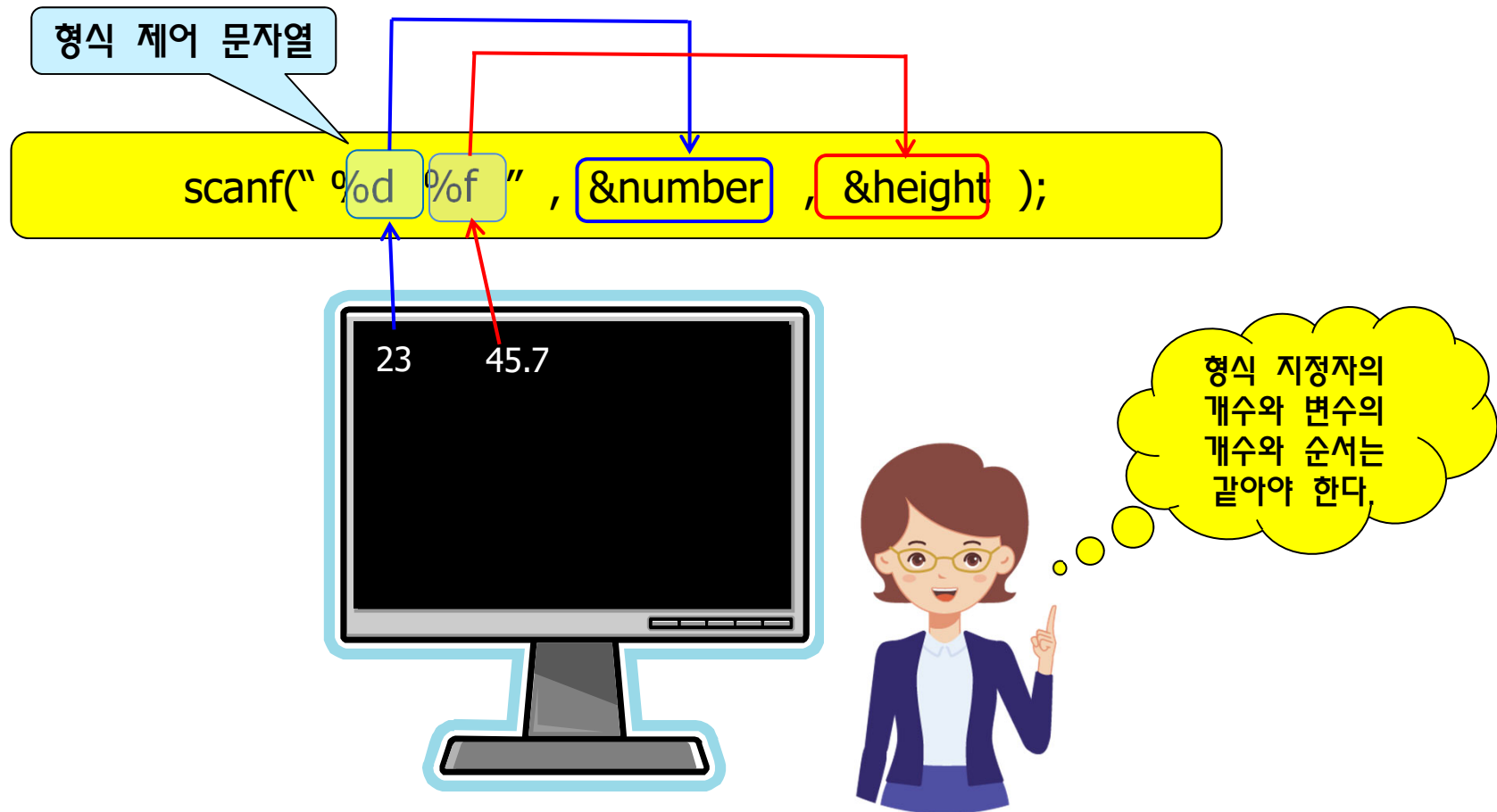
    return 0;               // 0을 외부로 반환
}
```





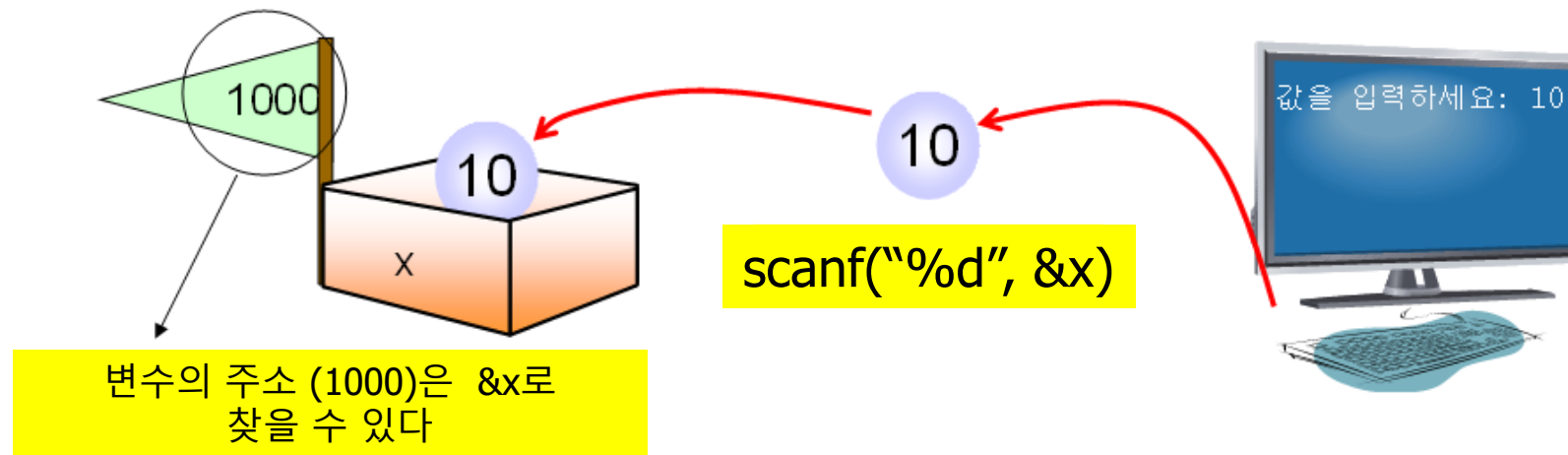
# scanf()

- ◆ **scanf()**: 표준입력장치 (키보드)로부터 입력을 하기 위한 라이브러리 함수



## scanf()의 동작

- ◆ 키보드로부터 문자열 (**string**)을 받아서 형식 지정자에 따라 변환한 후, 지정된 변수에 저장한다.
- ◆ 변수의 주소 (**address**)를 필요로 한다.



# scanf()의 형식지정자

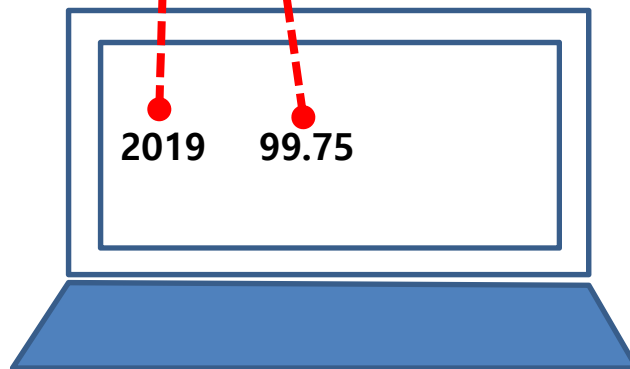
형식 지정자	의미	예
%d	정수를 10진수로 입력한다	scanf("%d", &i);
%f	float 형의 실수로 입력한다.	scanf("%f", &f);
%lf	double 형의 실수로 입력한다.	scanf("%lf", &d);
%c	문자 형태로 입력한다.	scanf("%c", &ch);
%s	문자열 형태로 입력한다.	char s[10]; scanf("%s", &s);

# scanf()

## ◆ 형식 지정자와 변수의 자료형은 일치하여야 함

- 형식제어 문자열 (예) : "%d %lf"

```
int number;  
double score;  
scanf("%d %lf", &number, &score):
```



scanf()나 printf()함  
수의 형식제어 문  
자열은 해당 변수  
의 자료형과 일치  
하여야 합니다.

# 실수 (real number) 입력

## ◆ 실수 (real number) 입력

- float (%f) 또는 double (%lf) 형식으로 입력

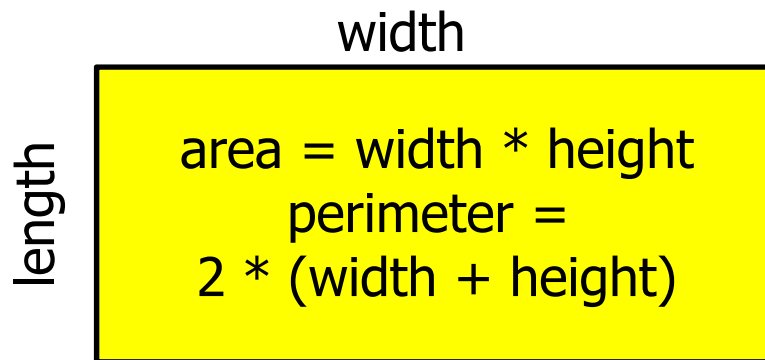
◆ **float** ratio = 0.0;  
scanf("%f", &ratio);

◆ **double** scale = 0.0;  
scanf("%lf", &scale);

주의!!!

## 사각형의 둘레와 면적

- ◆ 필요한 변수는 가로(width), 세로(length), 면적(area), 둘레(perimeter)라고 하자.
- ◆ 변수의 자료형은 실수를 저장할 수 있는 double형으로 하자.
- ◆  $\text{area} = \text{width} * \text{length};$
- ◆  $\text{perimeter} = 2 * (\text{width} + \text{length});$



# 사각형의 둘레와 면적 계산 – Pseudo code

## Algorithm PerimeterAndArea\_Rectangle()

```
1: // input arguments: none
2: double width; // local variable
3: double length; // local variable
4: double area;
5: double perimeter;
6: input width;
7: input length;
8: area = width * length;
9: perimeter = (width + length) * 2.0;
10: printout area and perimeter of rectangle
    with its width and length;
11: end
```

# 사각형의 둘레와 면적 계산 – Source Coding

```
/** Perimeter and Area of Rectangle */  
#include <stdio.h>  
  
int main(void)  
{  
    double width, length;  
    double area, perimeter;  
  
    printf("사각형의 폭과 높이를 입력하시오 : ");  
    scanf("%lf", &width);  
    scanf("%lf", &length);  
    area = width * length;  
    perimeter = 2 * (width + length);  
    printf("폭(%lf), 높이(%lf)인 사각형 : ", width, length);  
    printf(" 넓이 : %lf, ", area);  
    printf(" 둘레 : %lf\n", perimeter);  
    return 0;  
}
```



## 효과적인 프로그램 개발 방법

# 효과적인 프로그램 개발 방법 Tip 1

## ◆ 프로그램 설계 (알고리즘 및 자료구조 설계)에 더 많은 시간을 투자할 것

- 주어진 문제를 해결하기 위한 효율적인 알고리즘을 먼저 구성 할 것
- 구성된 알고리즘에 적합한 자료구조를 선택할 것
- 예) 100,000개의 학생 데이터로 부터 0.01초의 데이터 처리 시간 내에 특정 조건을 만족하는 학생을 탐색 (search)하여야 하는 경우, 어떤 자료구조를 사용하며, 어떤 방식으로 탐색할 것인가 ?
  - array vs. linked list vs. binary tree
  - linear search vs. binary search

## 효과적인 프로그램 개발 방법 Tip 2

### ◆ 프로그램 오류 수정을 위한 **debugging** 방법을 먼저 숙달 할 것

- 프로그램 소스코드 상에 존재하는 문법적 에러 및 논리적인 에러를 빨리 찾아내는 방법을 먼저 숙달
- Visual studio의 debugging 기능을 먼저 확인하고, 숙달할 것
  - break point 기능
  - trace 기능
  - 각 단계에서의 local variable 값 확인

# 오류메시지의 예

## ◆ C 프로그래밍에서 자주 발생하는 오류메시지

C 프로그램 문법적 오류	오류 메시지	올바른 입력
#include <stdjo.h>	C1083 포함파일을 열 수 없습니다: 'stdjo.h': No such file or directory	#include <std <b>i</b> o.h>
#include stdio.h	C2006: '#include': expected "FILENAME" or <FILENAME>	#include <stdio. <b>&gt;</b> h>
int x double y;	C2144: 구문오류: double은 ';'다음에 와야 합니다.	int x; double y;
retun 0;	C2065 'retun': 선언되지 않은 식별자입니다.	return 0;
main() { .... }	C4430: 형식지정자가 없습니다. int로 가정합니다.	<b>int</b> main() { .... return 0; }
void main() int x;	E0130 '{'가 필요합니다. C2144: int는 ';'다음에 와야 합니다.	void main() <b>{</b> int x; <b>}</b>

# Homework 1

# Homework 1

- 1.1 사용자로부터 원기둥의 반지름, 원기둥의 높이를 입력받고, 이 원기둥의 체적 (volume)과 표면적 (surface area)를 계산하는 프로그램의 **pseudo code**를 작성하라.
- 1.2 위1.1문제에서 작성한 **pseudo code**에 따라 C 프로그램으로 작성하고, 실행화면을 **capture**하여 **source code**와 함께 제출하라.
- 1.3 100명의 학생들의 성적을 차례로 입력하고, 가장 우수한 성적, 가장 낮은 성적, 평균 성적을 구하는 알고리즘의 **pseudo code**를 작성하라. (반복문 사용 가능)
- 1.4 C 프로그램에서 많이 발생하는 오류메시지 20개 이상의 이유와 해결 방법을 표로 만들어라.