

# C언어 강의자료

문정욱

A decorative graphic consisting of several light blue squares of varying sizes arranged in a stepped pattern on the left side of the slide.

# C언어 맛보기 1

# 기본 프로그램의 작성과 이해

## ■ Hello World 프로그램

- preprocessing directive
  - preprocessor가 처리하는 명령어
  - #으로 시작한다.
- main 함수
  - 프로그램에서 시작 위치를 의미한다.
  - 반드시 이름이 main이어야 한다.
- return 0
  - 이 문장을 수행하면 main 함수가 종료된다.
  - 0은 정상적 함수가 종료되었음을 의미한다.
    - cf) 비정상적 종료에서는 다른 값을 반환하기도 하지만 큰 의미는 없다.

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

← preprocessing  
directive

```
int main(void)
{
    printf("Hello, World\n");
    return 0;
}
```

← main  
function

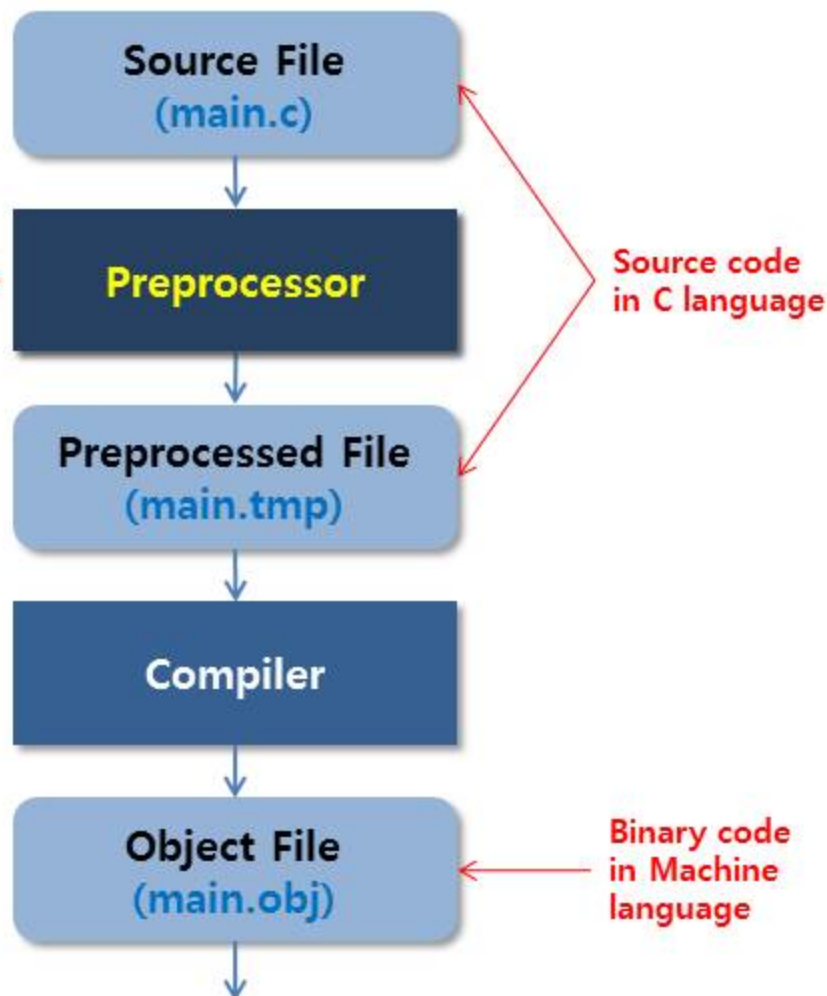
↑  
The return value of  
main function

# 기본 프로그램의 작성과 이해

## Hello World 프로그램

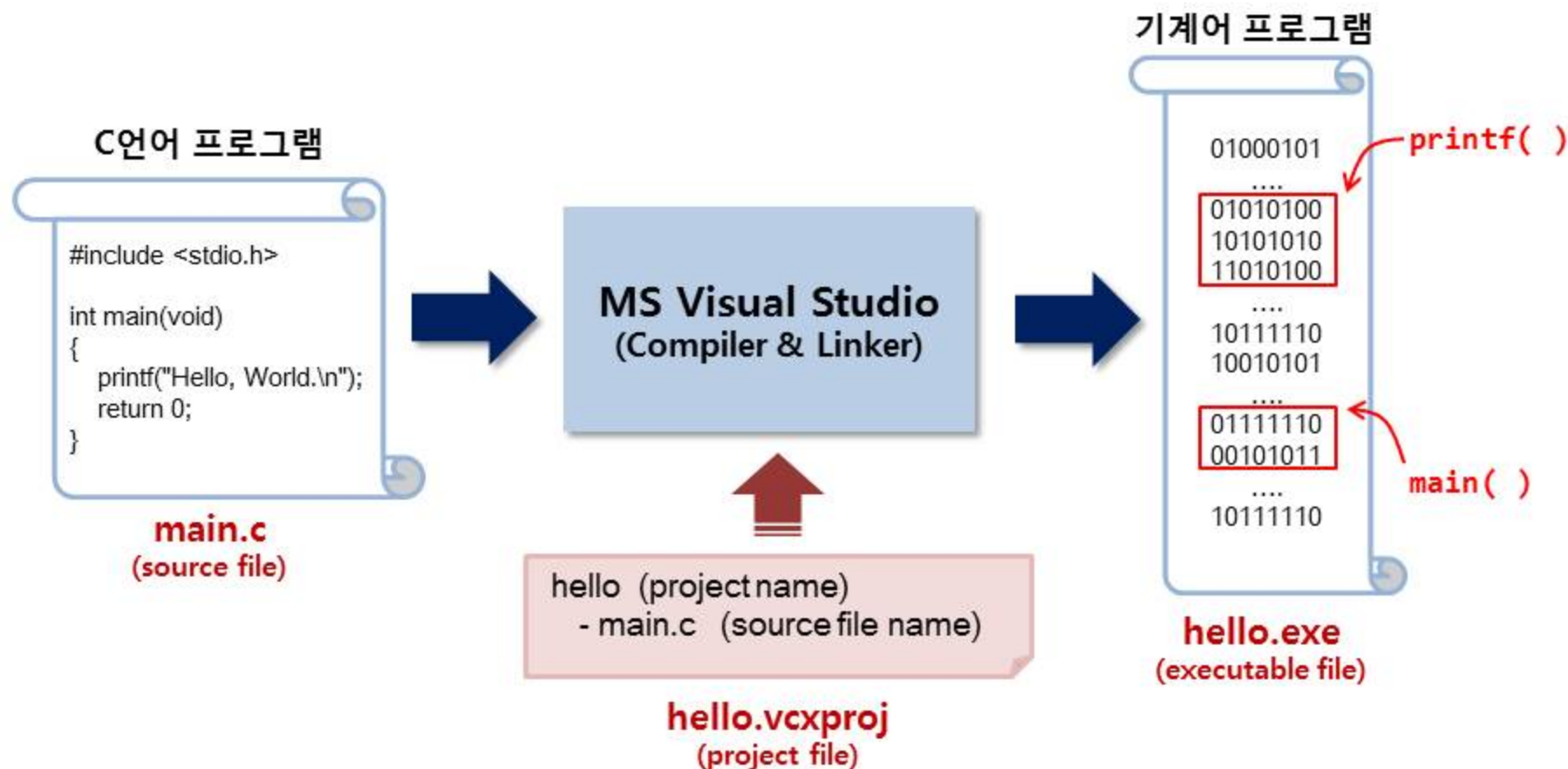
```
#include <stdio.h>
```

```
int main(void)
{
    printf("Hello, World\n");
    return 0;
}
```



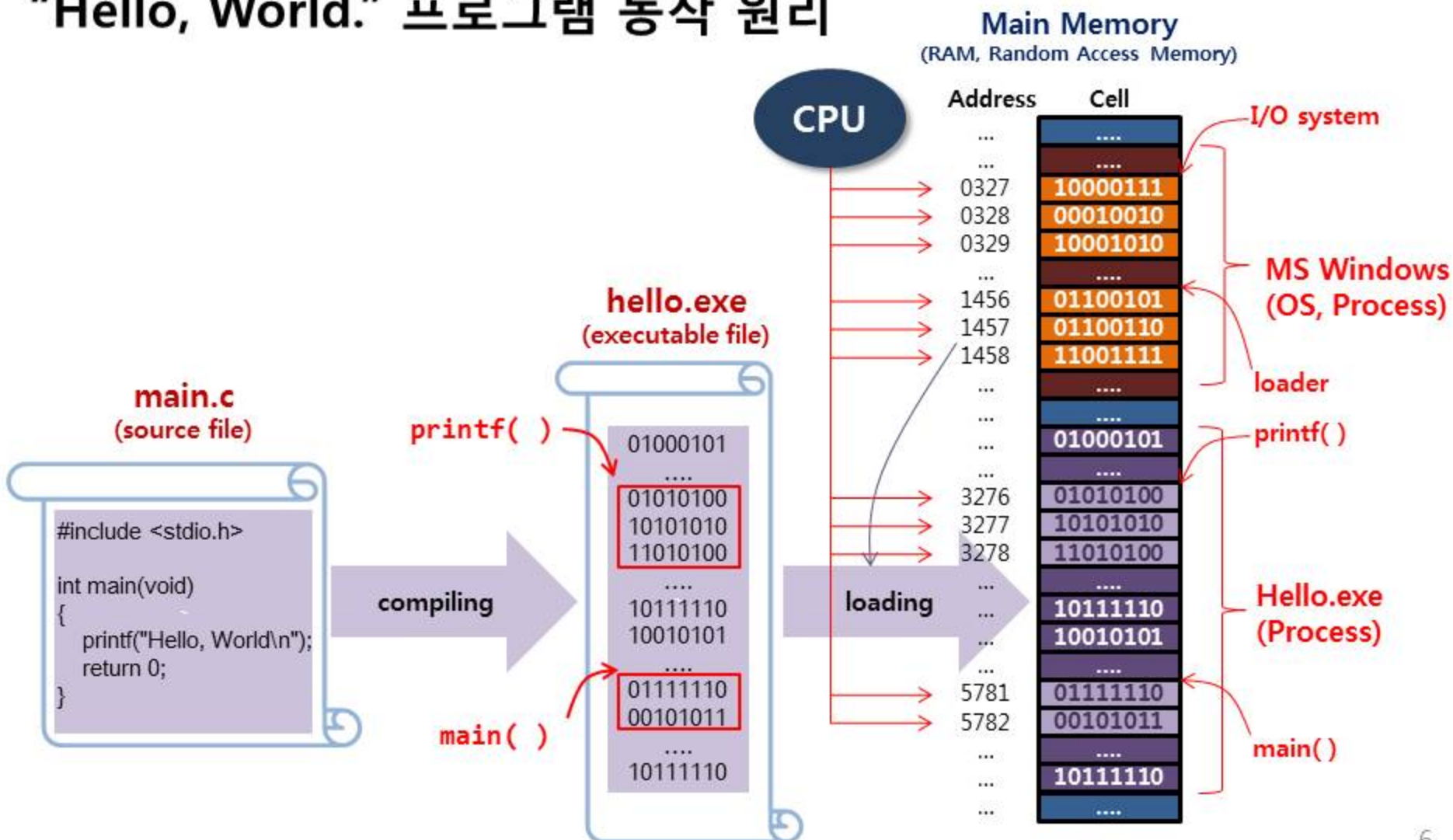
# 프로그램의 동작 원리

## "Hello, World." 프로그램 동작 원리



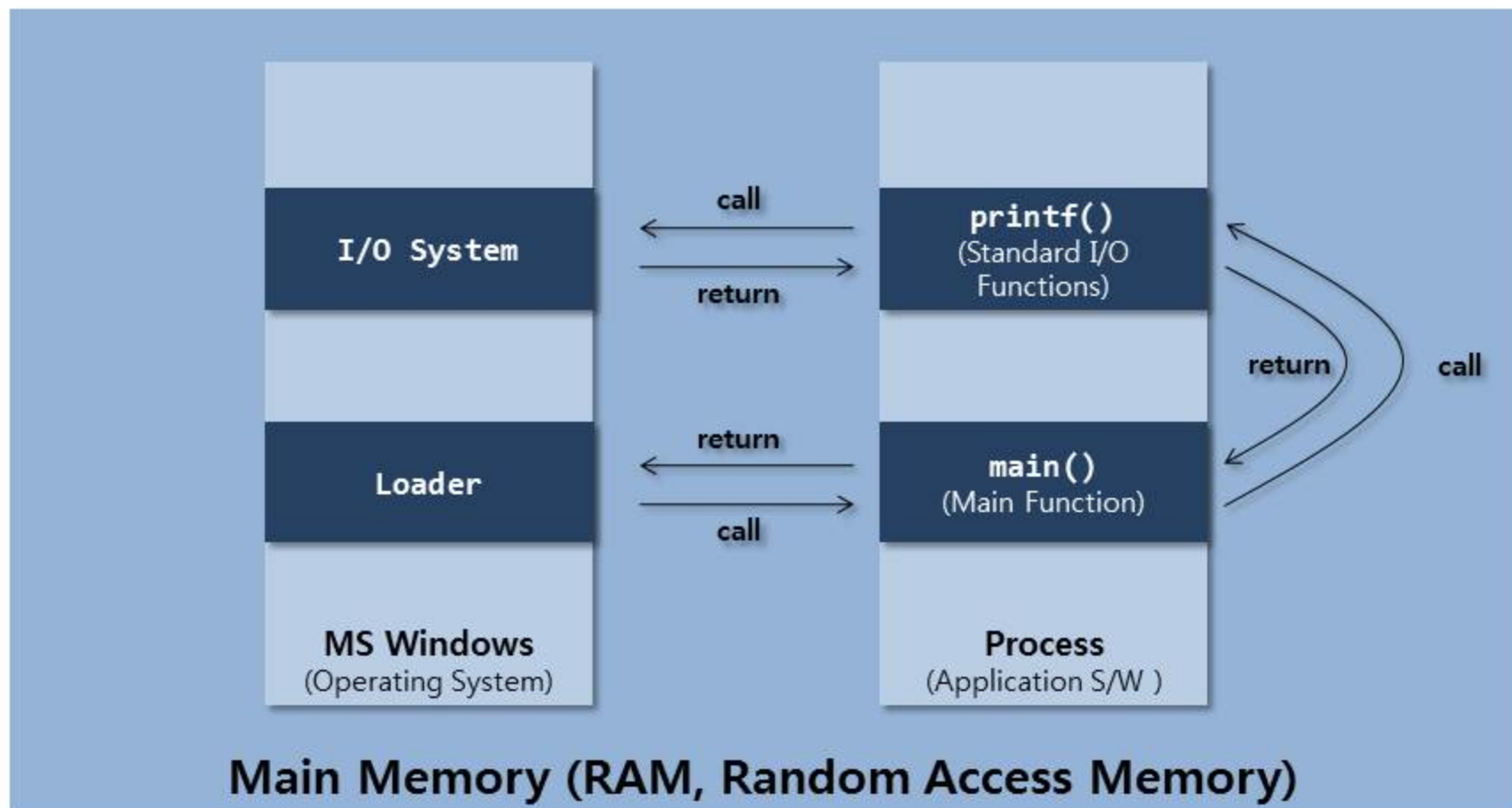
# 프로그램의 동작 원리

## "Hello, World." 프로그램 동작 원리



# 프로그램의 동작 원리

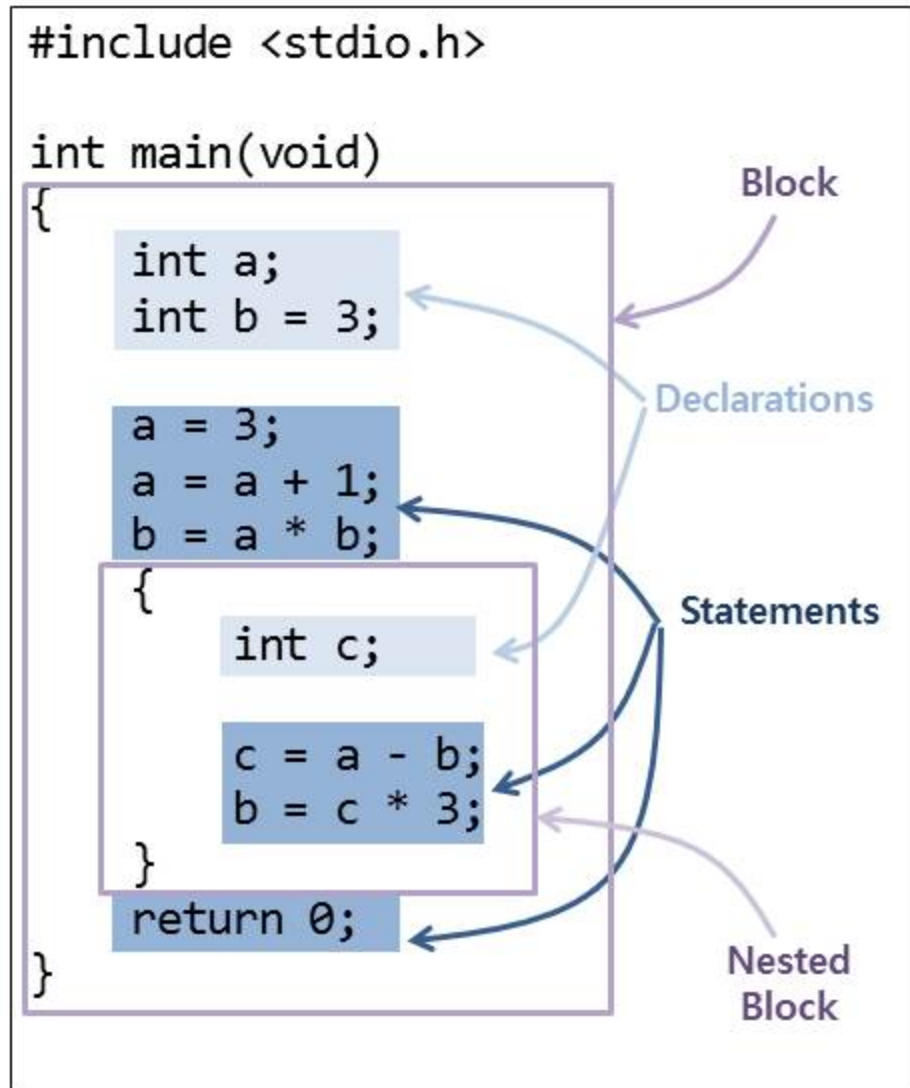
## "Hello, World." 프로그램 동작 원리





# C언어 프로그램의 구조

- 선언(declaration)
  - 식별자(타입, 변수, 함수)의 선언
  - 식별자는 반드시 선언된 후 사용되어야 함.
  - 세미콜론(;)으로 끝남.
  - 변수, 함수의 주소는 컴파일러와 운영체제에 의해 결정.
  - 선언은 기계어로 번역 안됨.
- 문장(statement)
  - C 언어에서 명령 수행 단위
  - 세미콜론(;)으로 끝남.
  - 문장은 기계어로 번역 됨.
- 블록(block)
  - Brace( { } )로 표시
  - 복합 문장을 하나의 문장처럼 취급
  - 블록 = 선언문 + 문장
  - 함수 시작과 끝을 나타냄





# C언어 프로그램의 구조

- `main()` 함수에서 선언과 문장의 개수
  - 선언(declaration):
    - 앞으로 사용할 식별자의 컴파일러에게 미리 알림.
    - 세미콜론으로 끝남.
  - 문장(statement):
    - 문장은 반드시 선언 다음에 나타남.
    - 선언이 아니며 세미콜론으로 끝남.
    - 블록은 하나의 문장으로 간주.

```
#include <stdio.h>

int main(void)
{
    int a;
    int b = 3;
    }

    a = 3;
    a = a + 1;
    b = a * b;
    {
        int c;

        c = a - b;
        b = c * 3;
    }
    return 0;
}
```

2 declarations

5 statements

# C언어 프로그램의 구조


- 선언과 문장의 순서
  - 문장은 모든 선언 뒤에 나타나야 한다.
  - 그렇지 않을 경우 **구문오류 (syntax error)** 발생
    - C89(C90)에서는 선언과 문장의 순서를 지켜야 하나, C++을 위해 만들어진 C언어 표현인 C99 이후의 표준에서는 순서가 중요하지 않다.

```
#include <stdio.h>

int main(void)
{
    int a;
    a = 3;
    int b = 3;

    a = a + 1;
    b = a * b;
    {
        int c;

        c = a - b;
        b = c * 3;
    }
    return 0;
}
```

 wrong position  
(syntax error)

# 주석

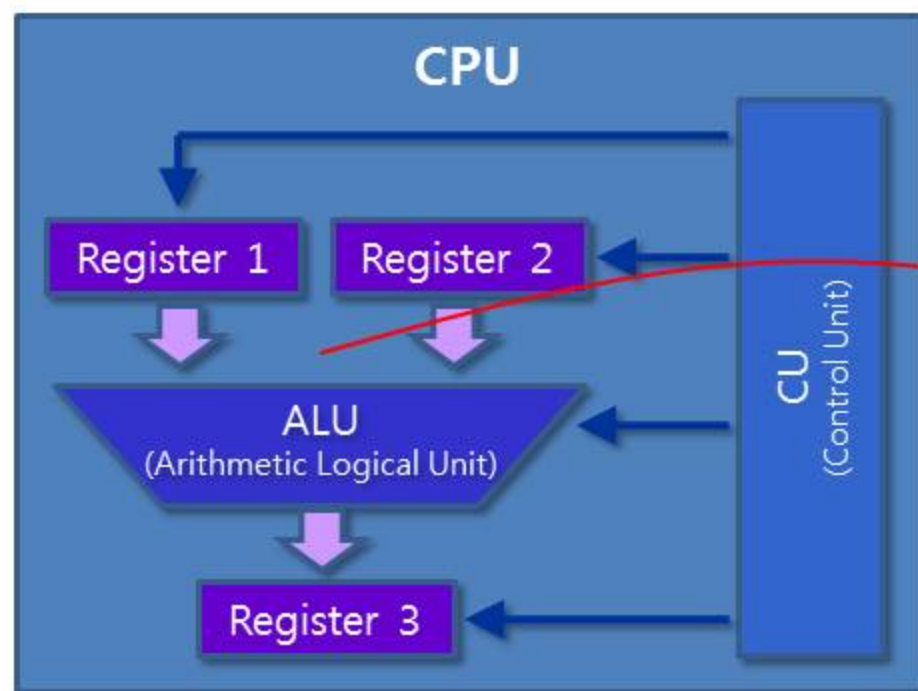
## ■ 주석문(comment)

- Multi-line comment
  - `/*`로 시작해서 `*/` 끝남
- Single-line comment
  - `///`로 시작함.
  - Single-line comment는 C89(C90)의 문법이 아니다. 하지만, 대부분의 현대 컴파일러들은 이 부분을 C언어 문법으로 간주한다.

```
/* this is a comment. */  
#include <stdio.h>  
  
int main(void)  
{  
    /* This is  
    a comment, too. */  
    printf("Hello, World\n");  
    return 0; // return zero  
}
```

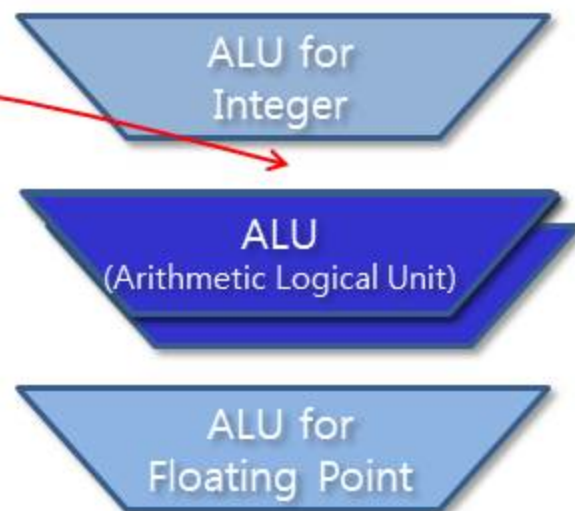
# 변수

## ALU의 구조



CPU(Central Processing Unit)

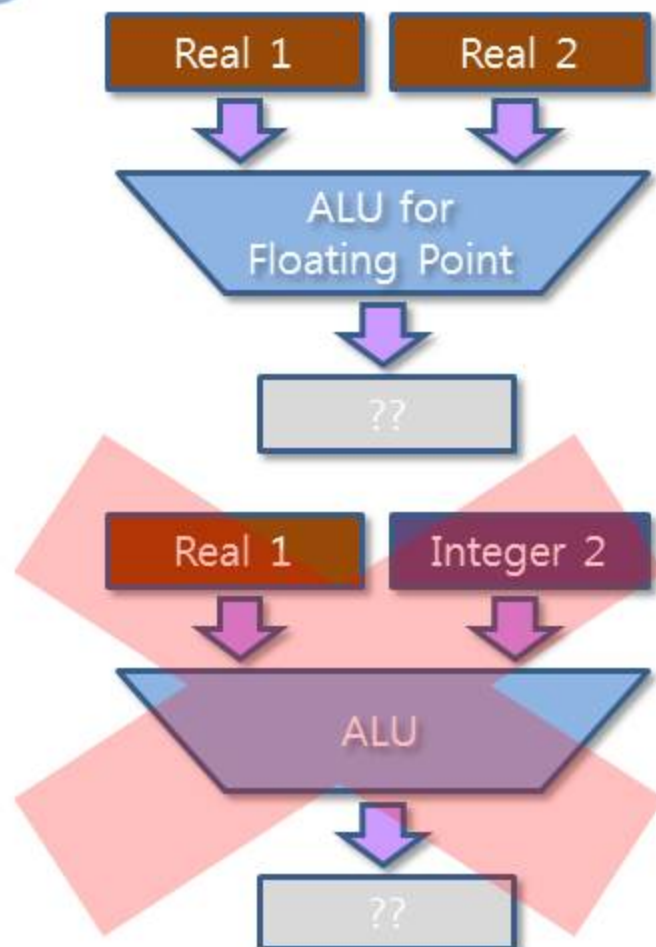
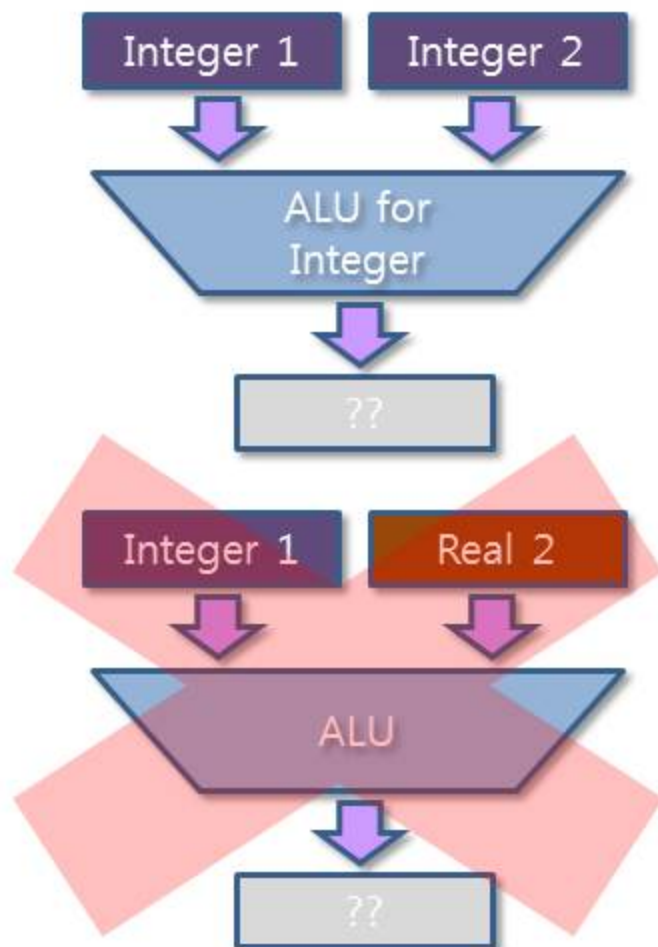
CPU가 계산할 수 있는 값:  
**정수, 실수**



# 변수

## ALU의 구조

CPU 안에 회로가 많아지고  
구조가 복잡해지면 CPU의 온도가 높아지므로  
CPU의 동작 클럭을 높이기 힘들어진다.  
그러므로 정수 ALU와 실수 ALU만 만든다.





# 변수

- 변수의 정의
  - 값을 저장하는 메모리 공간  
( 변수 = 메모리(main memory) )

- 변수의 선언

*Variable Declaration:*

**type variable;**

- 변수의 종류

종류	설명
int	정수형(integer type)
double	실수형(real type) (floating-point type)

```
#include <stdio.h>

int main(void)
{
    int a ; // integer type
    double b ; // real type
    return 0;
}
```

Diagram illustrating variable declaration in the code snippet above:

- variable** (red text): Points to the variable names `a` and `b`.
- type** (red text): Points to the data types `int` and `double`.



# 변수

## ■ 변수(Variable)의 이름

- 구성
  - 알파벳, 언더바(\_), 숫자
- 제약조건
  - 첫 글자는 알파벳 또는 언더바(\_)
  - 대소문자를 구별함
  - 31자 이하로 작성할 것
  - keyword(reserved word)를 사용하면 안 됨

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

```
#include <stdio.h>

int main(void)
{
    int a;           // ok
    int abc123;      // ok
    int abc_123;     // ok
    int _abc123;     // ok

    int 123abc;      // error
    int abc^^;       // error
    int 한글변수;    // error
    int void;        // error
    int return;      // error

    return 0;
}
```

# 변수

## ■ 변수의 초기화

*Variable Declaration:*

**type variable = value;**


- 여러 변수의 선언과 초기화
  - 콤마(,)를 사용하여 여러 개의 변수를 선언하고 초기화할 수 있다.

```
#include <stdio.h>

int main(void)
{
    int    a = 3    ;
    double b = 3.0  ;

    int    c, d=3, e;
    double f=2.1, g=5.6;

    return 0;
}
```

 **initializer**

# 연산자

## ■ 대입 연산자(assignment operator)

*Assignment Operator:*

**variable = value;**

- 왼쪽 변수에 오른쪽 값을 저장한다.
- 변수에 값 대입: 좌/우 피연산자의 타입이 같아야 한다.

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

```
int main(void)
```

```
{
```

```
    int    a = 3    ;
```

```
    double b = 3.0  ;
```

```
    a = 5 ;
```

```
    b = 5.0 ;
```

```
    return 0;
```

```
}
```

integer value

floating point value

assignment operator

# 연산자

- 대입 연산자(assignment operator)
  - 동작
    - 오른쪽 수식의 값을 왼쪽 변수에 저장한다.
  - 변수 값의 증감
    - 즉, 변수 값의 증감 효과가 있다.
    - 주의: "같다"는 뜻이 아님.

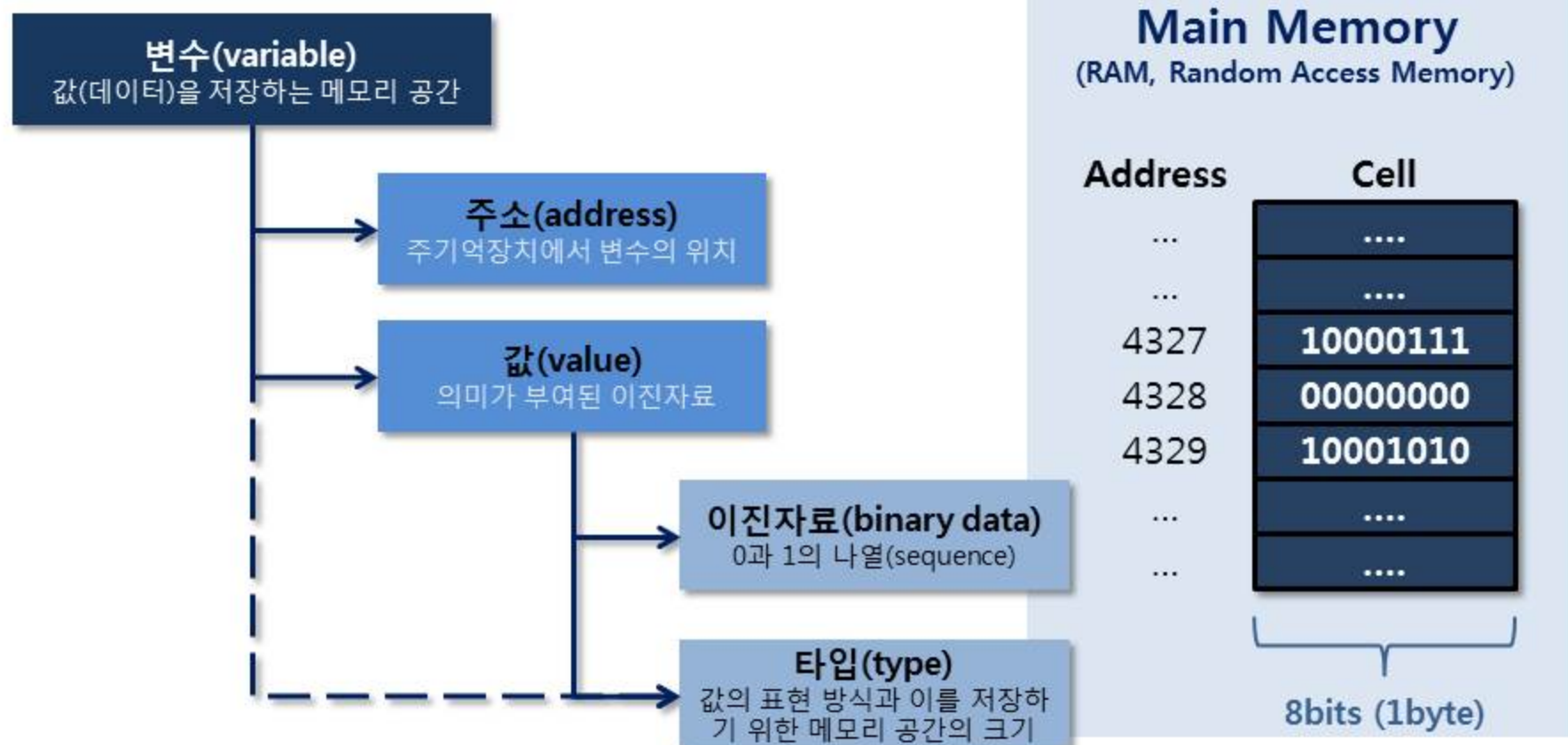
```
#include <stdio.h>

int main(void)
{
    int a = 1;

    a = a + 1;      // a = 2;
    a = a + 2;      // a = 4;
    a = a - 1;      // a = 3;
    a = a - 2;      // a = 1;
    return 0;
}
```

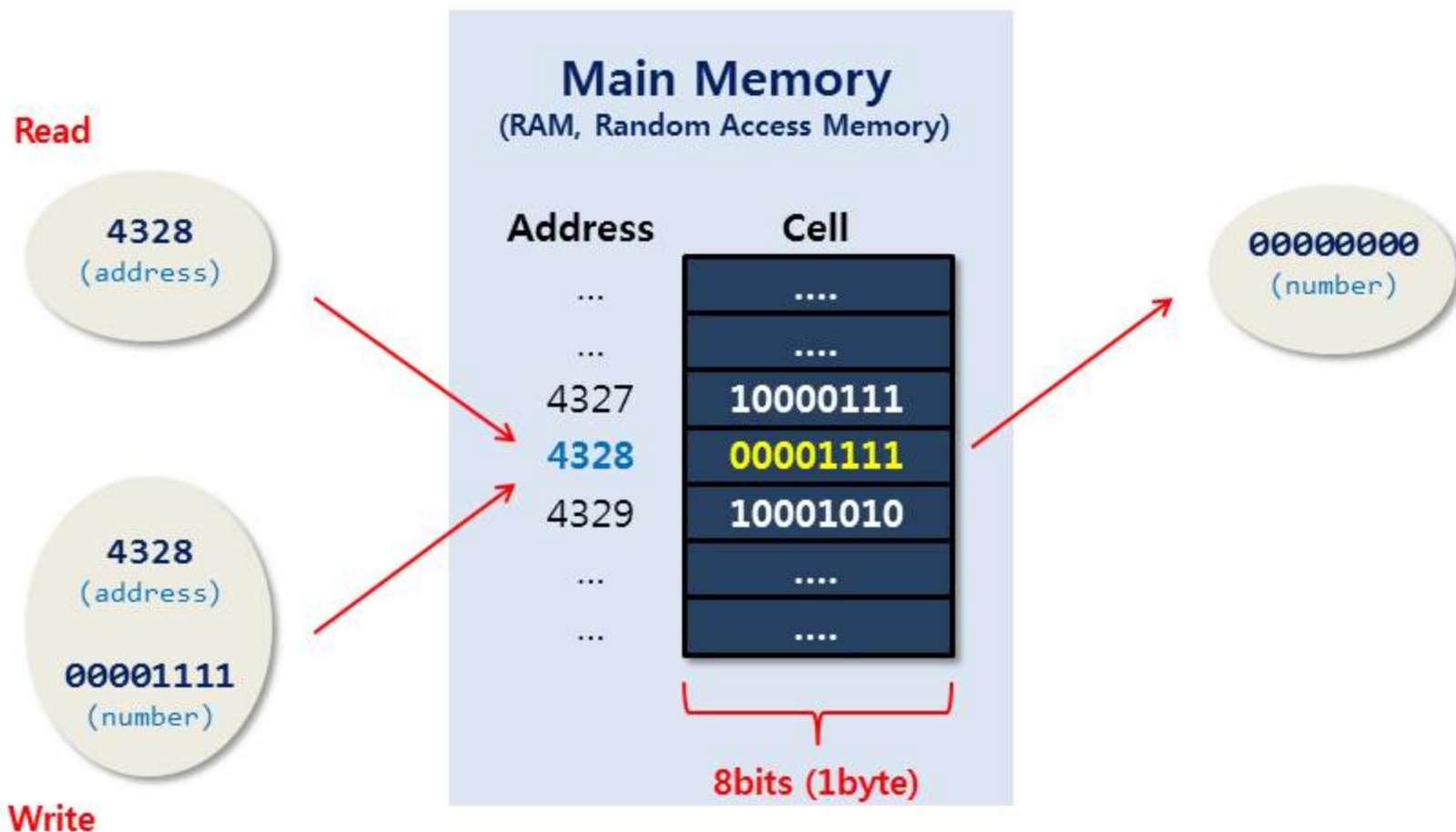
# 변수

## 변수(variable)의 의미와 구성요소



# 변수

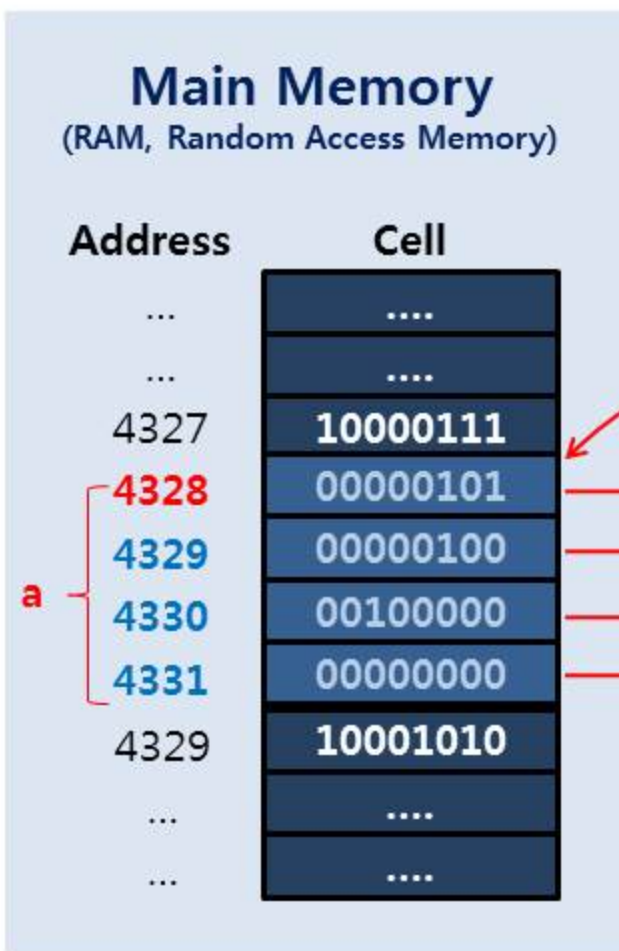
## 메인 메모리(Main Memory, RAM)의 동작





# 변수

## 정수(int)형 변수의 구조 (Intel Architecture의 경우)

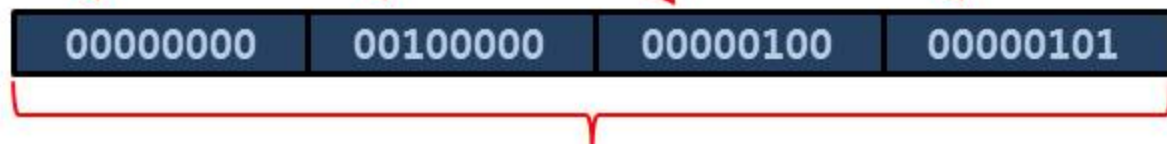


컴파일러와 운영체제가 프로그래머를 대신하여 변수의 주소는 정한다.  
그러므로, 프로그래머는 변수가 메모리에 어디에 위치할지 고민할 필요가 없다.

```
#include <stdio.h>

int main(void)
{
    int a;

    a = 2098181;
    return 0;
}
```



int: 4bytes (32bits)  
**2098181<sub>(10)</sub>**

# 표준 입출력 함수

## ■ 개행(new line) 문자

- "\n" 으로 표현
- 개행 문자 이후의 문자들은 이전 문자들이 출력된 위치 다음 줄에 출력된다.

### 입출력 결과

Hello, World. Hello, World. 계속하려면 아무 키나 누르십시오 . . .

### 입출력 결과

Hello, World. Hello, World.  
계속하려면 아무 키나 누르십시오 . . .

### 입출력 결과

Hello, World.  
Hello, World.  
계속하려면 아무 키나 누르십시오 . . .

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

```
int main(void)
```

```
{
```

```
    printf("Hello, World.");
```

```
    printf("Hello, World.");
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{
```

```
    printf("Hello, World.");
```

```
    printf("Hello, World.\n");
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{
```

```
    printf("Hello, World.\n");
```

```
    printf("Hello, World.\n");
```

```
    return 0;
```

```
}
```

# 표준 입출력 함수

- 개행(new line) 문자
  - "\n" 으로 표현
  - 개행 문자 이후의 문자들은 이전 문자들이 출력된 위치 다음 줄에 출력된다.

## 입출력 결과

```
Hello, World. 1
Hello, World. 2
Hello, World. 3
Hello, World. 4
Hello, World. 5
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    printf("Hello, World. 1\n");

    printf("Hello,");
    printf(" World. 2\n");

    printf("H");
    printf("e");
    printf("l");
    printf("l");
    printf("o, World. 3\n");

    printf("Hello, World. 4\nHello, World. 5\n");
    return 0;
}
```

# 표준 입출력 함수

- 정수 및 실수의 출력
  - 모니터 출력: printf() 함수
- FSF(Format Specification Field)

함수	정수	실수
printf()	%d	%f

입출력 결과

```
7
3
10
7.700000
3.300000
10.000000
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a=3;
    double b=3.3;

    printf("%d\n", 7); // ok
    printf("%d\n", a); // ok
    printf("%d\n", a+7); // ok

    printf("%f\n", 7.7); // ok
    printf("%f\n", b); // ok
    printf("%f\n", b+6.7); // ok
    return 0;
}
```

*integer* (points to the first three %d calls)

*floating point* (points to the last three %f calls)



# 표준 입출력 함수

- 정수 및 실수의 출력
  - 잘못된 FSF를 사용할 경우 일반적으로 논리오류가 발생한다.
  - 경우에 따라 실행오류가 발생할 수도 있다.

logical error

입출력 결과

```
0.000000
0.000000
0.000000
-858993459
1717986918
0
```

계속하려면 아무 키나 누르십시오 . . .

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

```
int main(void)
{
```

```
    int a=3;
```

```
    double b=3.3;
```

integer

```
    printf("%f\n", 7 ); // wrong
```

```
    printf("%f\n", a ); // wrong
```

```
    printf("%f\n", a+7 ); // wrong
```

```
    printf("%d\n", 7.7 ); // wrong
```

```
    printf("%d\n", b ); // wrong
```

```
    printf("%d\n", b+6.7 ); // wrong
```

```
    return 0;
```

```
}
```

floating point

# 표준 입출력 함수

- 정수 및 실수의 출력
  - 다른 문자와 함께 출력
  - 한번에 여러 수 출력

## 입출력 결과

a = 3, b = 3.300000

a = 3, b = 3.300000

계속하려면 아무 키나 누르십시오 . . .

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

```
int main(void)
```

```
{
```

```
    int a=3;
```

```
    double b=3.3;
```

```
    printf("a = %d, ", a );
```

```
    printf("b = %f\n", b );
```

```
    printf("a = %d, b = %f\n", a, b );
```

```
    return 0;
```

```
}
```

same result



# 표준 입출력 함수

- 정수 및 실수의 입력
  - 키보드 입력: scanf() 함수
- FSF(Format Specification Field)

함수	정수	실수
scanf()	%d	%lf

## 입출력 결과

```
3
5
3.14
5.140000
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    scanf("%d", &a);
    a=a+2;
    printf("%d\n", a);

    scanf("%lf", &b);
    b=b+2.0;
    printf("%f\n", b);

    return 0;
}
```

# 표준 입출력 함수

- 정수 및 실수의 입력
  - 한번에 여러 수 출력

## 입출력 결과

```
3 3.5
a = 3, b = 3.500000
3 3.5
a = 3, b = 3.500000
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

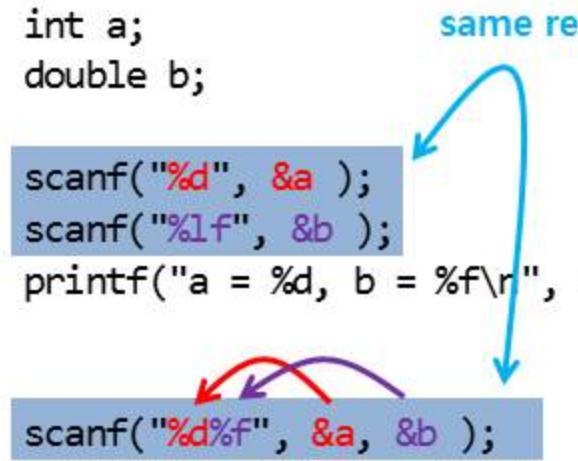
int main(void)
{
    int a;
    double b;

    scanf("%d", &a );
    scanf("%lf", &b );
    printf("a = %d, b = %f\r", a, b);

    scanf("%d%f", &a, &b );
    printf("a = %d, b = %f\n", a, b);

    return 0;
}
```

same result



# 연산자

- 산술(arithmetic) 연산자
  - 정수 값에 대한 산술 연산의 결과 값은 **정수**이다.

## 입출력 결과

```
a + b = 8
a - b = 2
a * b = 15
a / b = 1
a % b = 2
```

quotient

remainder

계속하려면 아무 키나 누르십시오 . . .

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

```
int main(void)
```

```
{
```

```
    int a=5, b=3;
```

```
    printf("a + b = %d\n", a + b );
```

```
    printf("a - b = %d\n", a - b );
```

```
    printf("a * b = %d\n", a * b );
```

```
    printf("a / b = %d\n", a / b );
```

```
    printf("a % b = %d\n", a % b );
```

```
    return 0;
```

```
}
```

integer  
value

# 연산자

- 산술(arithmetic) 연산자
  - 실수 값에 대한 산술 연산의 결과 값은 **실수**이다.
  - 실수 값에 대한 나머지 연산은 적용할 수 없다. (**구문오류**)

## 입출력 결과

```
c + d = 3.000000
c - d = 2.400000
c * d = 0.810000
c / d = 9.000000
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main(void)
```

```
{
```

```
    double c=2.7, d=0.3;
```

```
    printf("c + d = %f\n", c + d );
```

```
    printf("c - d = %f\n", c - d );
```

```
    printf("c * d = %f\n", c * d );
```

```
    printf("c / d = %f\n", c / d );
```

```
    // printf("c %% d = %f\n", c % d );
```

```
    return 0;
```

```
}
```

floating point  
value

Syntax Error

# 변수

- 초기화 안된 변수의 활용
  - 초기화 되지 않은 변수에는 **쓰레기 값(garbage value)**이 저장되어 있음.
  - 임의의 값이 저장되어 있지만 그 값이 무엇인지 예측할 수 없다. (**논리 오류**를 발생시킬 수 있다.)

입출력 결과

1559447316

계속하려면 아무 키나 누르십시오 . . .

입출력 결과

3

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

int main(void)
{
    int a;    // garbage value

    printf("%d\n", a);
    return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    int a;    // garbage value

    a = 3;
    printf("%d\n", a);
    return 0;
}
```

# 연산자

- 산술(arithmetic) 연산자
  - 산술 연산에 대하여 **정수와 실수를 피연산자로 혼용하는 것은 바람직하지 못하다.**
    - 그 이유는 CPU의 ALU는 정수 전용 ALU와 실수 전용 ALU만 있기 때문이다.
  - 경우에 따라 **논리 오류**의 원인이 될 수 있다.

## 입출력 결과

10.000000

18.000000

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

int main(void)
{
    double f, c=10.0;

    f = 9 / 5 * c;           // undesirable
    printf("%f\n", f);

    f = 9.0 / 5.0 * c;       // ok
    printf("%f\n", f);
    return 0;
}
```



# 연산자

- Divided by Zero
  - 나누기 연산자의 오른쪽 피연산자가 0(zero)이면 **실행 오류(run-time error)**가 발생한다.
  - %(나머지 연산자)도 마찬가지.

입출력 결과

```
12 3
4
계속하려면 아무 키나 누르십시오 . . .
```

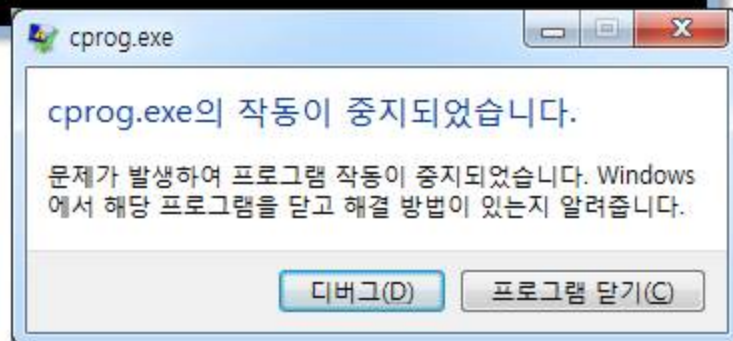
```
#include <stdio.h>
```

```
int main(void)
{
    int a, b;

    scanf("%d%d", &a, &b);
    printf("%d\n", a / b );
    return 0;
}
```

입출력 결과

12 0



# 표준 입출력 함수

- 정수 및 실수의 입력
  - 정수형/실수형 변수 앞에 **& (ampersand)**를 사용해야 한다.
  - 그렇지 않을 경우 **실행오류**가 발생할 수 있다.

```
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    scanf("%d", &a); // ok.
    scanf("%lf", &b); // ok.

    scanf("%d", a); // run-time error
    scanf("%lf", b); // run-time error
    return 0;
}
```

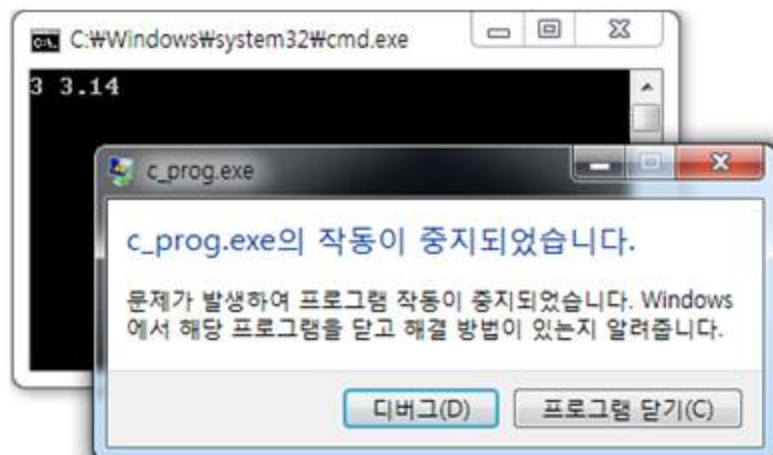
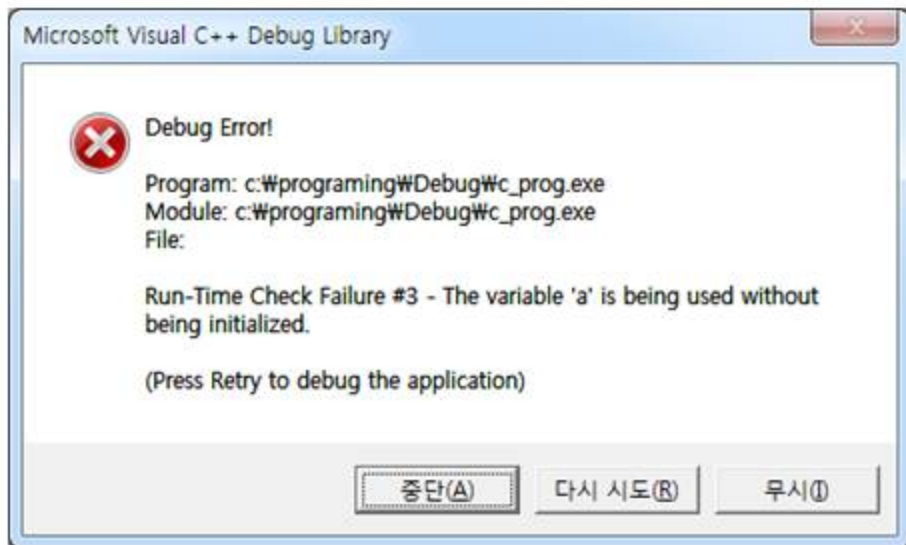
# 표준 입출력 함수

```
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    scanf("%d", a); // run-time error
    scanf("%lf", b); // run-time error
    return 0;
}
```

운영체제 (Operating System)이  
프로그램을 강제 종료시킴.  
(보안상의 이유)



# 표준 입출력 함수

- 정수 및 실수의 입력
  - 잘못된 FSF를 사용할 경우 일반적으로 실행오류가 발생한다.
  - 경우에 따라 논리오류가 발생할 수도 있다.

```
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    scanf("%lf", &a); // wrong
    scanf("%d", &b);  // wrong
    return 0;
}
```

# 표준 입출력 함수

- 정수 및 실수의 출력
  - scanf() 함수는 FSF 이외의 다른 문자의 사용은 주의해야 한다.
  - scanf() 함수의 사용법을 완전히 알기 전까지 **FSF만 사용**하는 것이 바람직함.

logical error:  
input trouble

입출력 결과

3

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d\n", &a); // trouble
    printf("%d\n", a);
    return 0;
}
```