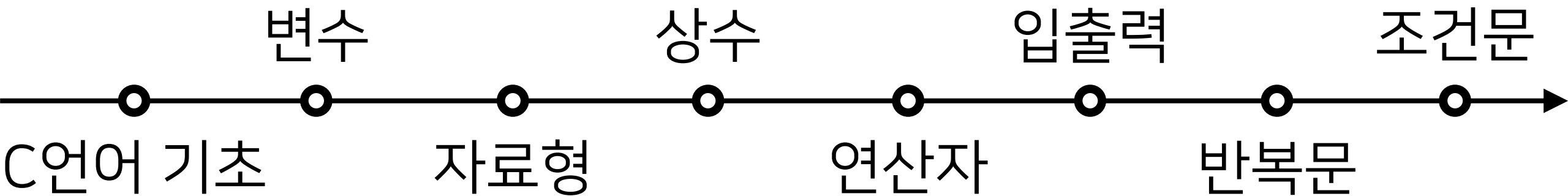



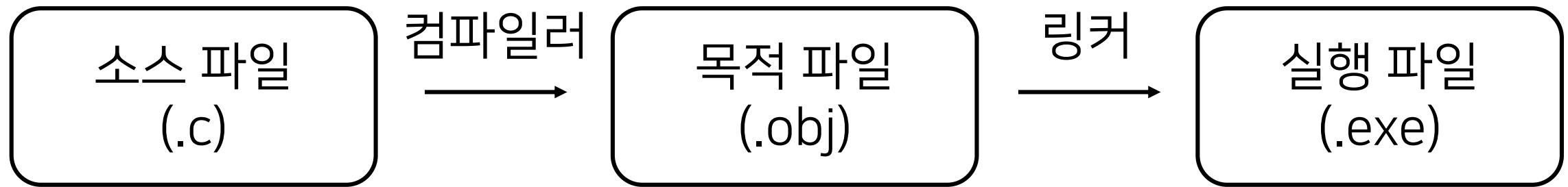
EC C언어 스터디

-2강-

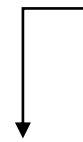




C언어 기초



STanDard Input Output



- #include <stdio.h>
- #define a b

한 줄 주석

// 주석 내용

여러 줄 주석

```
/*  
주석 내용1  
주석 내용2  
*/
```

- 세미콜론은 모든 명령어 뒤에 써야 합니다.
- 중괄호는 포함 관계를 나타냅니다.



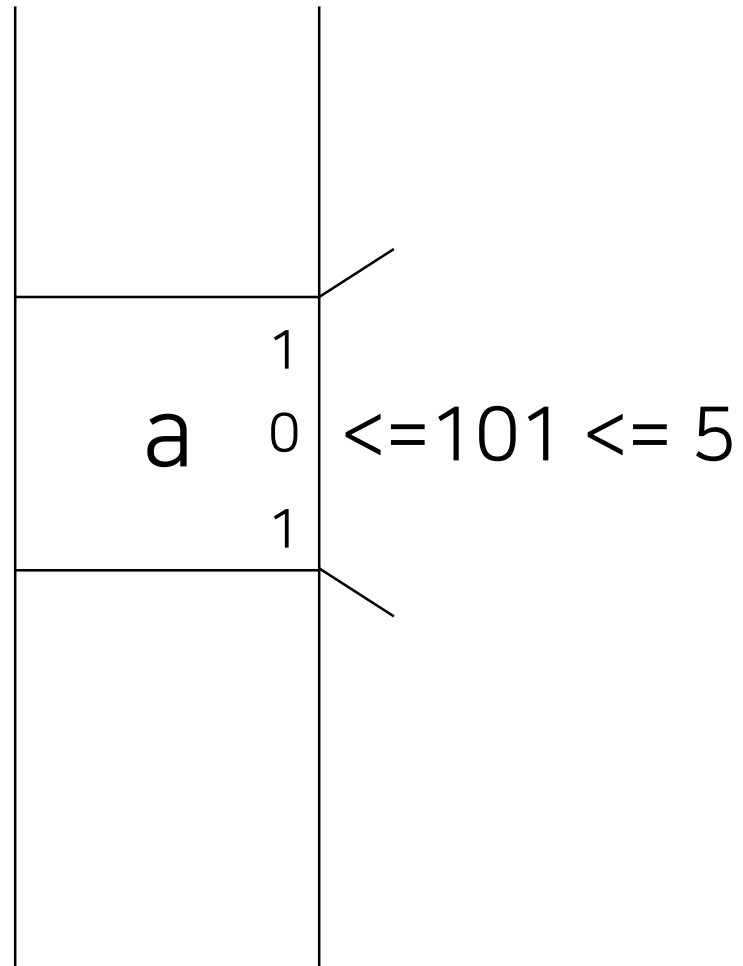
변수

변수를 만드는 것



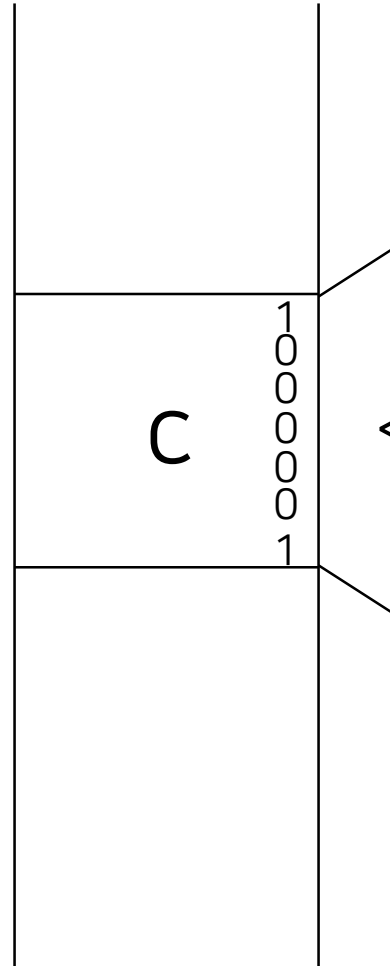
컴퓨터 메모리 공간에
이름을 붙이는 것

int a = 5



<메모리>

int c = 'A'



<=10000001 <= 'A'

<메모리>

주소가 필요함!

int a = 5

a

1

0

1

$\leq 101 \leq 5$

<메모리>

13 => 1 1 0 1



4비트의 크기

- 변수를 만든다는 것은 메모리에 공간을 할당받고 값을 저장하는 것이다.
- 변수의 이름은 메모리 공간 자체의 이름 이다.
- 해당 변수에 접근하기 위해서 해당 메모리 위치의 주소가 필요하고 이것이 변수의 주소이다

- 변수의 이름은 알파벳, 숫자, _로 이루어 진다
- 변수의 이름은 숫자로 시작할 수 없고 C언어에 존재하는 키워드와 같은 이름으로는 만들 수 없다.
- 변수의 이름은 가급적 의미있게 짓는 것이 좋다.

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

```
int main()
```

```
{
```

```
    int a = 1, b = 2;
```

```
    printf("%d", c);
```

```
    int c = a + b;
```

```
}
```

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

```
int main()
```


```
{
```

```
    int a, b;
```

```
    int c = a + b;
```

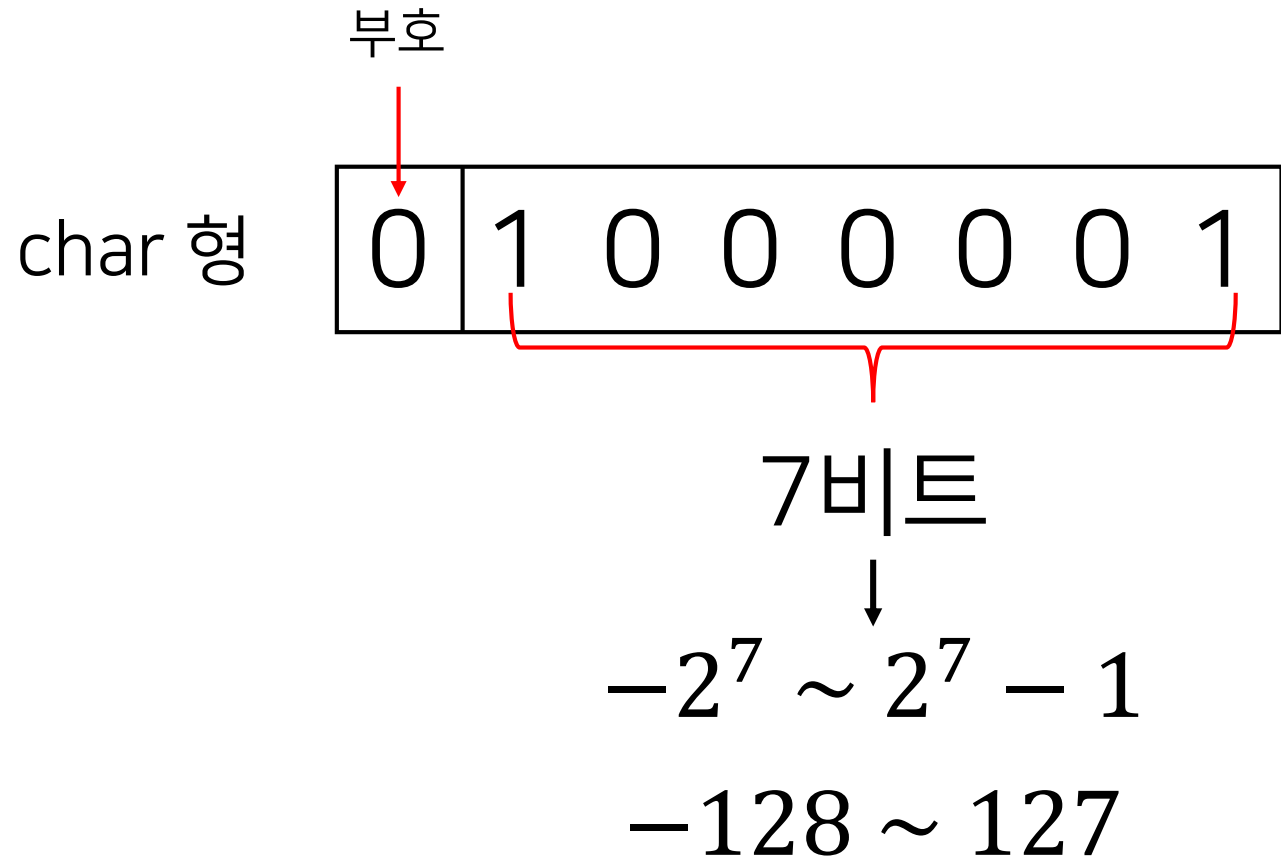
```
    printf("%d", c);
```

```
}
```

자료형

정수형 자료형	크기	범위
char(character)	1바이트	-128 ~ 127
int(integer)	4바이트	-2,147,483,648 ~ 2,147,483,647
long long int	8바이트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807



```
char c = 'A'
```

```
printf("%c", c + 1); => B
```

- 오버플로우 : 최대값보다 더 큰 값을 변수에 저장하려고 할 때
 - 범위를 벗어난 만큼 최솟값에서 증가한 값을 가진다
- 언더플로우 : 최솟값보다 더 작은 값을 변수에 저장하려고 할 때
 - 범위를 벗어난 만큼 최댓값에서 감소한 값을 가진다

- `char c = 128` -> 오버플로우 발생 -> `c = -128`
- `char c = -130` -> 언더플로우 발생 -> `c = 126`

실수형 자료형	크기	범위	소수점
float	4바이트	$3.4 \times 10^{38} \sim 3.4 \times 10^{-38}$	6자리
double	8바이트	$-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$	14자리



상수


```
int a = 5  
char c = 'A'  
printf("Hello")
```

```
#define PI 3.14
```

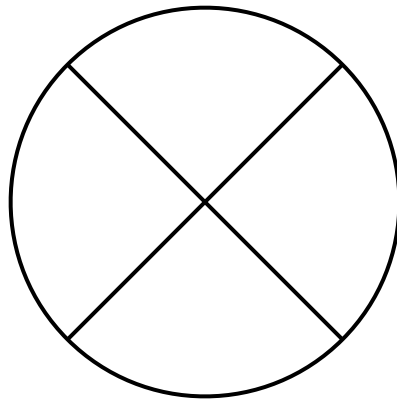
```
const double PI = 3.14
```



연산자

산술 연산자	+ - / * %
증강 연산자	++ --
관계 연산자	< > <= >= == !=
논리 연산자	&& !
비트 연산자	& ~ ^ << >>
기타	& ?:(조건 연산자) ,

$$(a + n*b) \% b = a$$



```
int a = 1;  
printf("%d",a++) // 1  
printf("%d",a)   // 2
```

<후위 연산>

```
int a = 1;  
printf("%d",++a) // 2  
printf("%d",a)  // 2
```

<전위 연산>

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A B
0	0	0
0	1	1
1	0	1
1	1	1

A	!A
0	1
1	0

비트 연산자	사용 예제	연산 후
&	101 & 011	001
	101 011	111
~	~ 101	010
^	101^010^110	001
<<	101 << 2	10100
>>	101 >> 2	1

$a \ll b = a \times 2^b$



구조	이름	내용
&변수	주소 연산자	변수의 주소 반환
(조건)?(내용1):(내용2)	조건 연산자(삼항 연산자)	조건이 참이면 내용1 실행, 조건이 거짓이면 내용 2 실행
명령, 명령	coma 연산자	명령을 한 줄에 이어서 쓸 수 있게 함

풀어보세요!

- [사칙연산](#) : 산술 연산자를 써봅시다.

문제 풀이

- [A/B](#) : 문제 조건을 잘 보고 잘 출력해 줘야 합니다.
printf는 기본적으로 소수점 6자리만 출력하기 때문입니다.



```
printf("서식 문자", 변수);
```

₩(역슬래쉬), " 를 출력하고 싶으면 ₩을 붙여야 합니다.

%를 출력하고 싶으면 %%로 써야 합니다.

이것들이 문자열 내부에선 특별한 키워드이기 때문입니다.

상황	예제	결과
공백 맞춰서 출력	<pre>printf("%5d\\n", 12); printf("%5d", 123);</pre>	12 123
0을 맞춰서 출력	<pre>printf("%05d\\n", 12); printf("%05d", 123);</pre>	00012 00123
반올림	<pre>printf("%.4f", 1.5555555);</pre>	1.5556

```
scanf("서식 문자", 변수주소);
```

상황	예제	결과
n개씩 입력받기	<code>scanf("%1d",&n);</code>	숫자를 한자리만 입력 받는다 1234 입력 -> n=1
데이터 무시	<code>scanf("%*2d%d",&n);</code>	처음 두자리의 숫자를 무시하고 나머지를 받음 1234 입력 -> n=34
	<code>scanf("%*d%d",&n);</code>	숫자하나를 무시하고 뒤에 것을 받음 12 34 입력 -> n=34

풀어보세요!

- 개 : 그대로 출력하는 문제입니다. 한번쯤은 해보는 것이 좋습니다.

문제 풀이

- 고양이 : 항상 문제에서 주어진 형식대로 출력해야 합니다.
그럴 땐 예제를 복붙하는 것이 좋습니다.



반복문

while(조건)

{

반복 내용

}

for(선언부; 조건; 반복후 실행)

{

반복 내용

}

```
for(;;)  
{  
    반복 내용  
}
```

풀어보세요!

- 합: 반복문을 통해 1부터 n까지의 합을 구합시다
- 구구단: 구구단의 N단을 출력하는 문제입니다.

문제 풀이

- 숫자의 합: 숫자를 한 자리씩 입력받아 더해야 합니다.
아까 배운 `scanf("%1d", &n)`를 이용해 봅시다



조건문

```
if(조건)
{
    실행 내용
}
```

```
if(A)
{
```

```
}
```

```
else
```

```
{
```

```
}
```

=

```
if(A)
{
```

```
}
```

```
if(!A)
```

```
{
```

```
}
```

if(A)		if(A)
{		{
}		}
else if(B)	=	if(!A&&B)
{		{
}		}

풀어보세요!

- 세 수 : 조건문을 이용해 중간값을 구해보세요
- 최댓값 : 반복문과 조건문을 이용해 최댓값을 구해보세요

문제 풀이

- 시험 성적 : if-else if-else 구조를 자유자재로 사용할 수 있어야 합니다



수고하셨습니다!