변수와 연산자
데이터 표현 방식의 이해

20200512

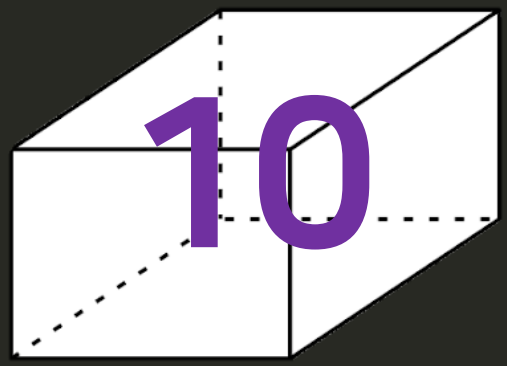
```c
#include <stdio.h>

int main(void)
{
    int num;

    num = 10;

    printf("%d", num);

    return 0;
}
```



num

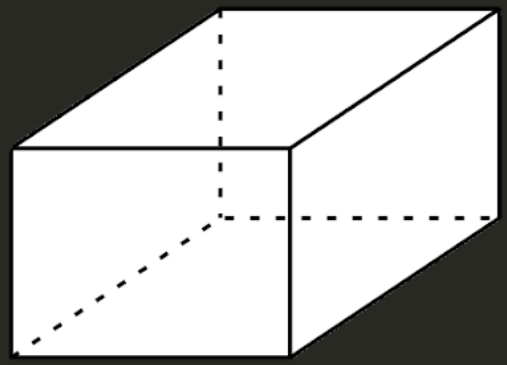
```c
#include <stdio.h>

int main(void)
{
    int num;

    num = 10;

    printf("%d", num);

    return 0;
}
```



num

C언어에서의 변수 선언 규칙

1. 변수의 이름은 알파벳(A-Z, a-z), 숫자(0-9), 언더바(_) 로 구성
2. 숫자로 시작할 수 없고, 키워드 혹은 예약어는 변수의 이름으로 사용 불가능 ex) return, void, int, while, for, if 등
3. 이름 사이에 공백 허용 X ex) how much = 1000; <- Error
4. 대소문자 구별 ex) Num 과 num 은 다른 거야~

변수의 자료형

정수형 변수	char (1)	char c = ‘a’;
	short (2)	short num1 = 12;
	int (4)	int num2 = 24;
	long (4)	long num3 = 1200;
실수형 변수	float (4)	float num1 = 0.123f;
	double (8)	double num2 = 3.14;

Practice 1 #변수 선언 규칙 #변수의 자료형 #변수와 연산자

<출력 결과>

10과 20의 합은 30입니다.

12.3과 34.5의 합은 46.8입니다.

<조건>

변수 이름은 아래와 같이 작성

int_x, int_y, dob_x, dob_y,
int_res, dob_res (6개)

Summary

1. 변수의 개념 / 변수에 아무것도 안 들어간다면?
2. 변수 선언 규칙

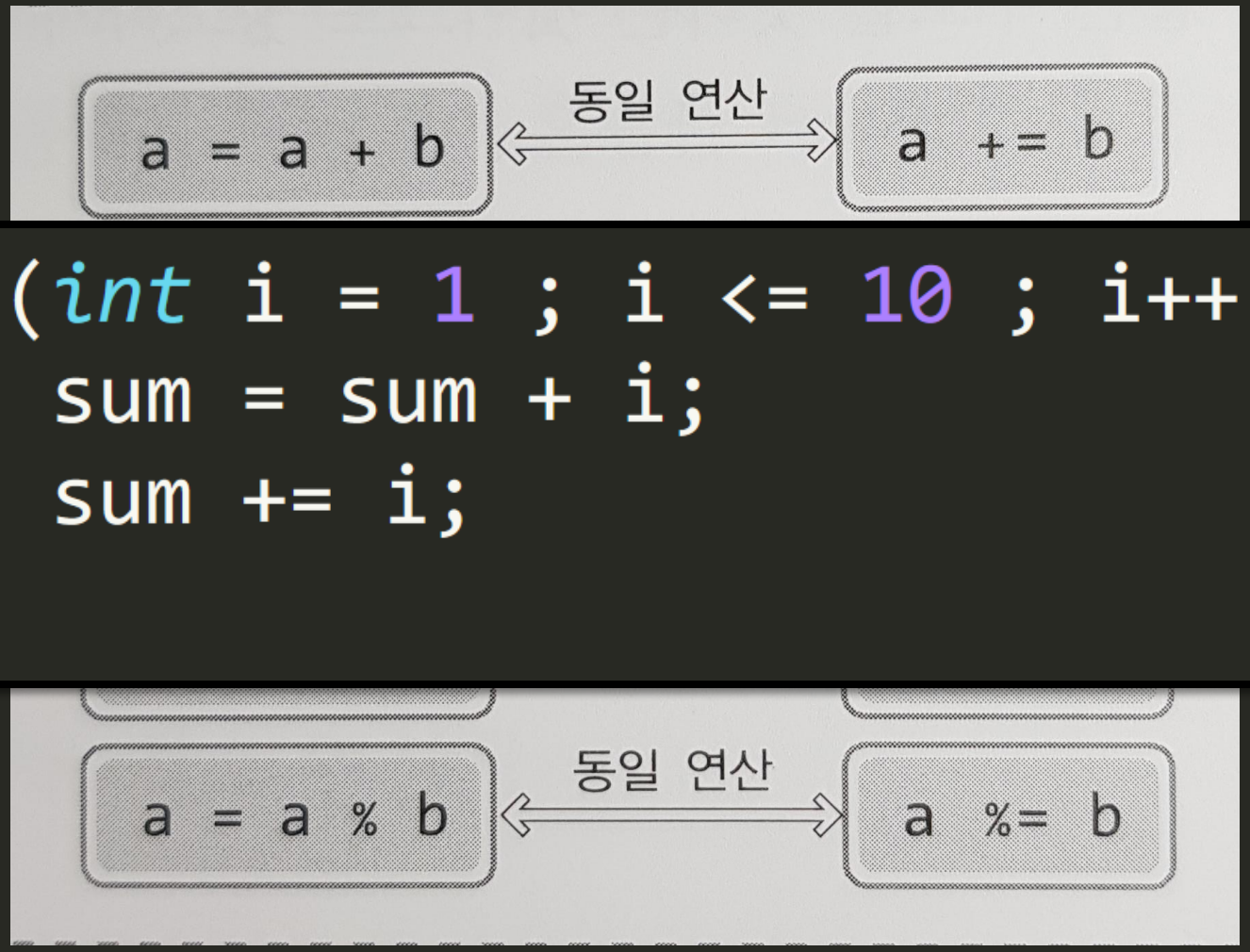
C언어에서 연산자는 어떤 종류가 있을까요

1. 산술 연산자 (Arithmetic Operator)
2. 대입 연산자 (Assignment Operator)
3. 관계 연산자 (Relational Operator)
4. 논리 연산자 (Logical Operator)
5. 비트 연산자 (Bitwise Operator)
6. 관계 연산자 (Conditional Operator)

산술 연산자와 대입 연산자

연산자	설명 및 예시	결합 방향
=	오른쪽에 있는 값을 왼쪽에 있는 변수에 대입 num = 100;	<-
+	num = 10 + 5;	->
-	num = 10 - 5;	->
*	num = 10 * 5;	->
/	num = 10 / 5;	->
%	num = 10 % 5; num = 7 % 3;	->

복합 대입 연산자




```java
for(int i = 1 ; i <= 10 ; i++) {
    sum = sum + i;
    sum += i;
}
```

Practice 2 #산술 연산자 #대입 연산자 #복합 대입 연산자

```c
#include <stdio.h>

int main(void)
{
    int num1=2, num2=4, num3=6;

    ?

    printf("Result: num1=%d, num2=%d, num3=%d\n", num1, num2, num3);
    return 0;
}
```

<조건>
num1은 더하기 연산,
num2은 곱하기 연산,
num3은 나머지(%) 연산을
사용해서 작성하시오.
(단, 복합 대입 연산자를 사용하기)

<출력 예시>
Result: num1=5, num2=16, num3=1

증가, 감소 연산자

```c
#include <stdio.h>

int main(void)
{
    int num1 = 9; // num1++
    int num2 = 9; // ++num2

    // num1++ 후위 증가 예시
    printf("Before num1 : %d\n", num1);
    printf("Operate num1 : %d\n", num1++);
    printf("After num1 : %d\n", num1);

    // ++num2 전위 증가 예시
    printf("Before num2 : %d\n", num2);
    printf("Operate num2 : %d\n", ++num2);
    printf("After num2 : %d\n", num2);

    return 0;
}
```

++은 "1을 더한다"라는 의미.

근데 앞에서 더하냐

뒤에서 더하냐에 따라서

계산이 달라진다.

Practice 3 #증가 감소 연산자 #후위 증가

오른쪽 코드의 출력 결과를
예상해보시오.

(난이도 조금 있지만 차근차근 단계별로
생각해보면 충분히 풀 수 있는 문제)

후위 : 대입하고 증가/감소
전위 : 증가/감소하고 대입

```c
#include <stdio.h>

int main(void)
{
    int num1 = 10;
    int num2 = (num1--) + 2;

    printf("num1: %d\n", num1);
    printf("num2: %d\n", num2);

    return 0;
}
```

Summary

1. 변수의 개념 / 변수에 아무것도 안 들어간다면?
2. 변수 선언 규칙
3. num1++ 는 대입하고 증가, ++num1는 증가하고 대입

관계 연산자 (비교 연산자)

연산자	설명 및 예시	결합 방향
<	작다 (왼쪽 기준)	->
>	크다	->
<=	작거나 같다	->
>=	크거나 같다	->
==	n1 == n2	->
!=	n1 != n2	->

관계 연산자 (비교 연산자)

```c
#include <stdio.h>

int main(void)
{
```

관계 연산자 (비교 연산자)들은

'조건을 만족하면 1을, 만족하지 않으면 0을 반환' 한다.

1은 참(True)을, 0은 거짓(False)을 대표하는 값이다.

```c
    printf("res1의 결과: %d\n", res1);
    printf("res2의 결과: %d\n", res2);
    printf("res3의 결과: %d\n", res3);
    printf("res4의 결과: %d\n", res4);

    return 0;
}
```

논리 연산자

연산자	설명 및 예시	결합 방향
&&	A와 B 모두 '참'이면 '참'을 반환 예시) A && B	->
||	A와 B 둘 중 하나라도 '참'이면 '참'을 반환 예시) A || B	->
!	반대의 것으로 반환 예시) !A	<-

&&

A	B	결과
거짓	거짓	거짓
거짓	참	거짓
참	거짓	거짓
참	참	참

VS

A	B	결과
거짓	거짓	거짓
거짓	참	참
참	거짓	참
참	참	참

||

논리 연산자

```c
#include <stdio.h>

int main(void)
{
    int num1 = 10;

    res3 = (!num1);

    printf("res1의 결과: %d\n", res1);
    printf("res2의 결과: %d\n", res2);
    printf("res3의 결과: %d\n", res3);

    return 0;
}
```

C언어는 0이 아닌 모든 값을 '참'으로 간주한다!

2.12 Precedence and order of evaluation (우선 순위 및 계산 순서)

precedence 높음

^
|
|
|
|
|
v

precedence 낮음

TABLE 2-1. PRECEDENCE AND ASSOCIATIVITY OF OPERATORS

OPERATORS	ASSOCIATIVITY
`( )  [ ]  ->  .`	left to right
`! ~ ++ -- + - * & (type) sizeof`	right to left
`*  /  %`	left to right
`+  -`	left to right
`<<  >>`	left to right
`<  <=  >  >=`	left to right
`==  !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right

Unary +, -, and * have higher precedence than the binary forms.

참고: 2장에서 다루지 않은 operator들도 포함됨

Summary

1. 변수의 개념 / 변수에 아무것도 안 들어간다면?
2. 변수 선언 규칙
3. num1++ 는 대입하고 증가, ++num1는 증가하고 대입
4. 관계 연산자, 논리 연산자 모두 참, 거짓과 관련되어 있음