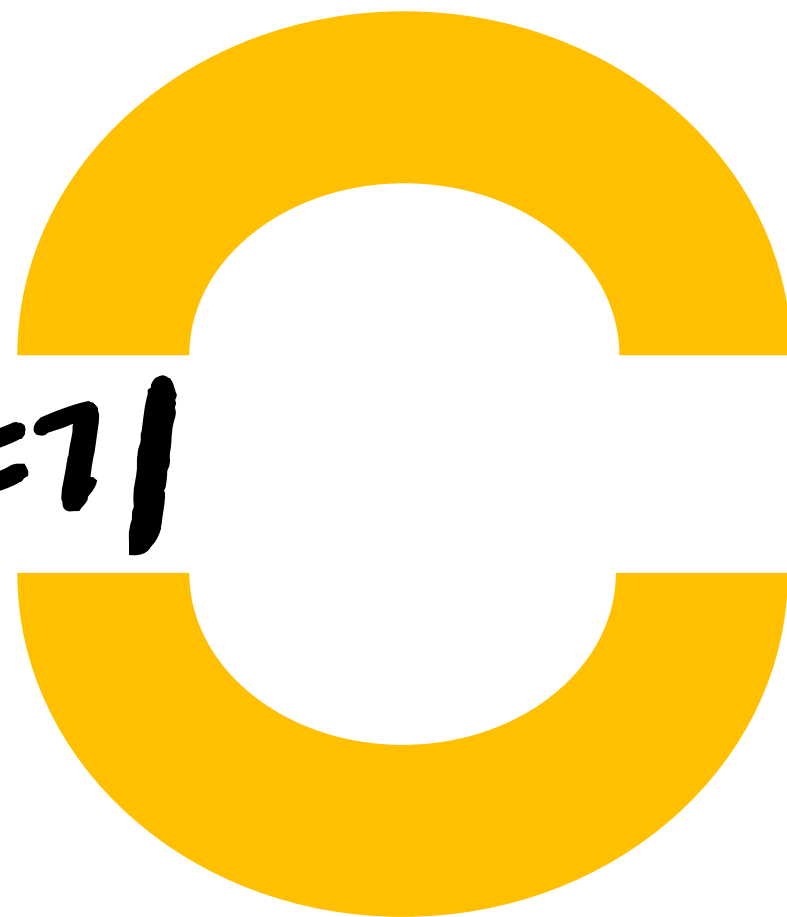


시글사는 개발자의

C언어 이야기



## 0. 시작하는 이야기

### 1. 프로그래밍 기본 원리

### 2. C언어의 특징과 구조

### 3. 데이터형과 변수

### 4. 문자와 문자열

### 5. 연산자

### 6. 조건문

### 7. 반복문

### 8. 배열

### 9. 포인터

### 10. 구조체

### 11. 함수포인터

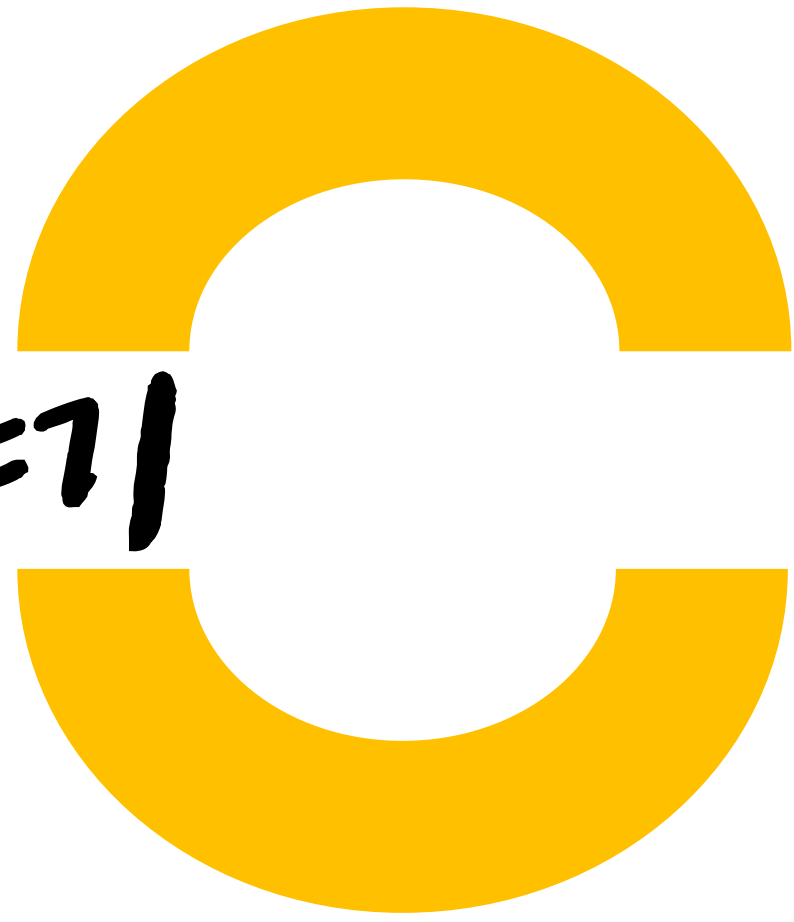




0. 시작하는 이야기

시글사는 개발자의

C언어 이야기



## 0. 시작하는 이야기

:: 강의 방향 및 소개

:: 사전 준비 사항

2진수와 16진수의 이해 [https://youtu.be/IZKej5s3T\\_w](https://youtu.be/IZKej5s3T_w)

문자를 처리하는 프로그래밍 원리 <https://youtu.be/apZFc0fAr5w>

Vi(Vim) 시작하기 [https://youtu.be/GWo\\_MxMIJJ4](https://youtu.be/GWo_MxMIJJ4)

윈도우10에서 우분투 APP 설치 <https://youtu.be/RKASf-XmPSw>

C언어를 공부하면 좋은 이유 <https://youtu.be/qlBGuYtY6GA>

Makefile 시작하기 <https://youtu.be/jnJL6ppn26Q>

Ctags 시작하기 <https://youtu.be/uYZOWAm5rWE>

시골사는 개발자의  
C언어 이야기

## 0. 시작하는 이야기

:: 강의 방향 및 소개


Youtube + PPT + GIT

Storytelling

Only C Programming

Includes many but not everything

The Curse of Knowledge



시글사는 개발자의  
C언어 이야기

## 0. 시작하는 이야기


:: 사전 준비 사항 (강의 환경)

Ubuntu Linux (in Windows10 WSL2)

- MobaXterm (<https://mobaxterm.mobatek.net>)
- XShell (<https://www.netsarang.com/ko/xshell>)

Editor & Compiler

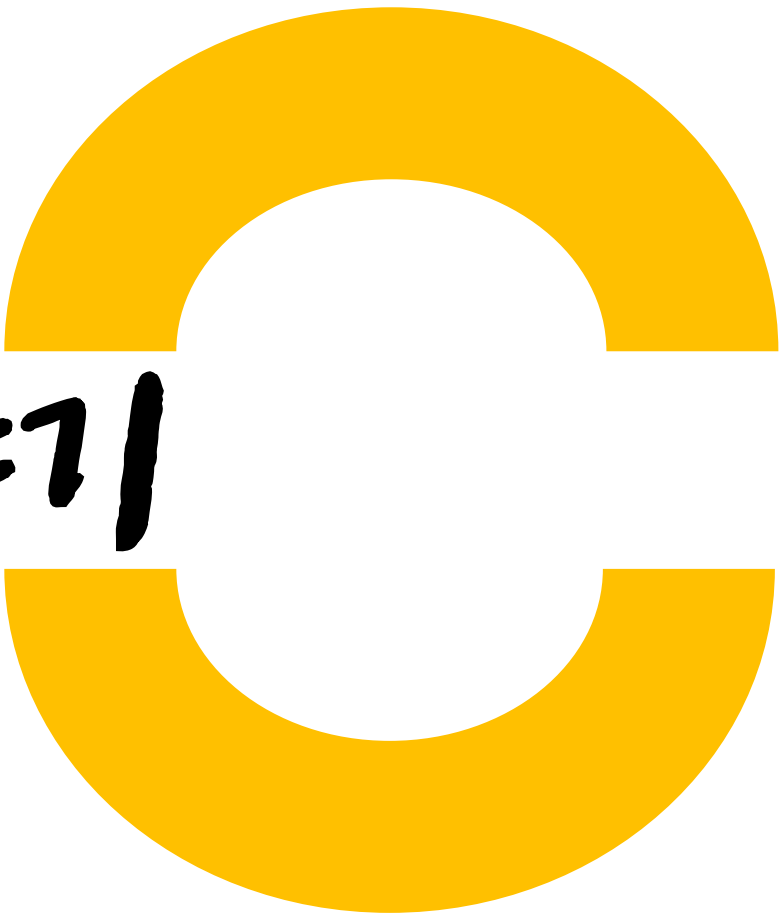
- vim & gcc
- vscode (<https://code.visualstudio.com>)
- Online ([https://www.onlinegdb.com/online\\_c\\_compiler](https://www.onlinegdb.com/online_c_compiler))



시골사는 개발자의  
C언어 이야기

1. 프로그래밍 기본 원리

시글사는 개발자의  
C언어 이야기



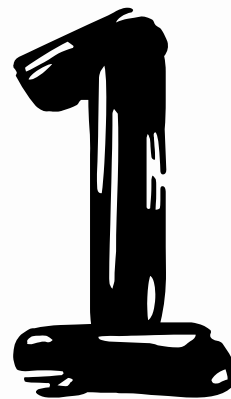
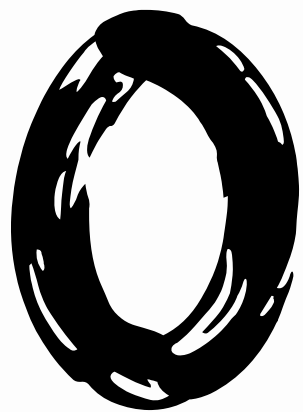
“컴퓨터는  
몇 가지 언어를  
이해할까요?”





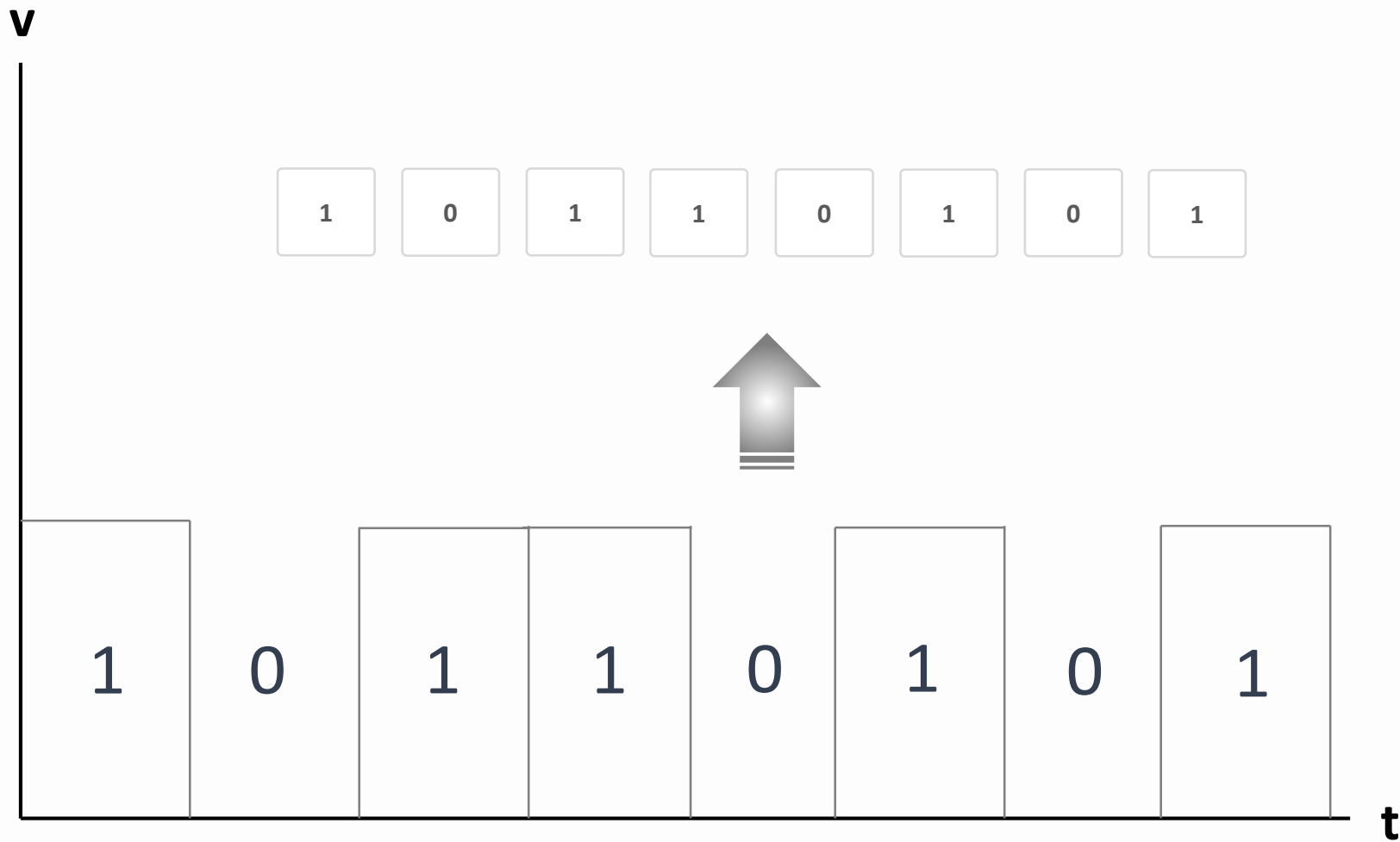


## 1. 프로그래밍 기본 원리



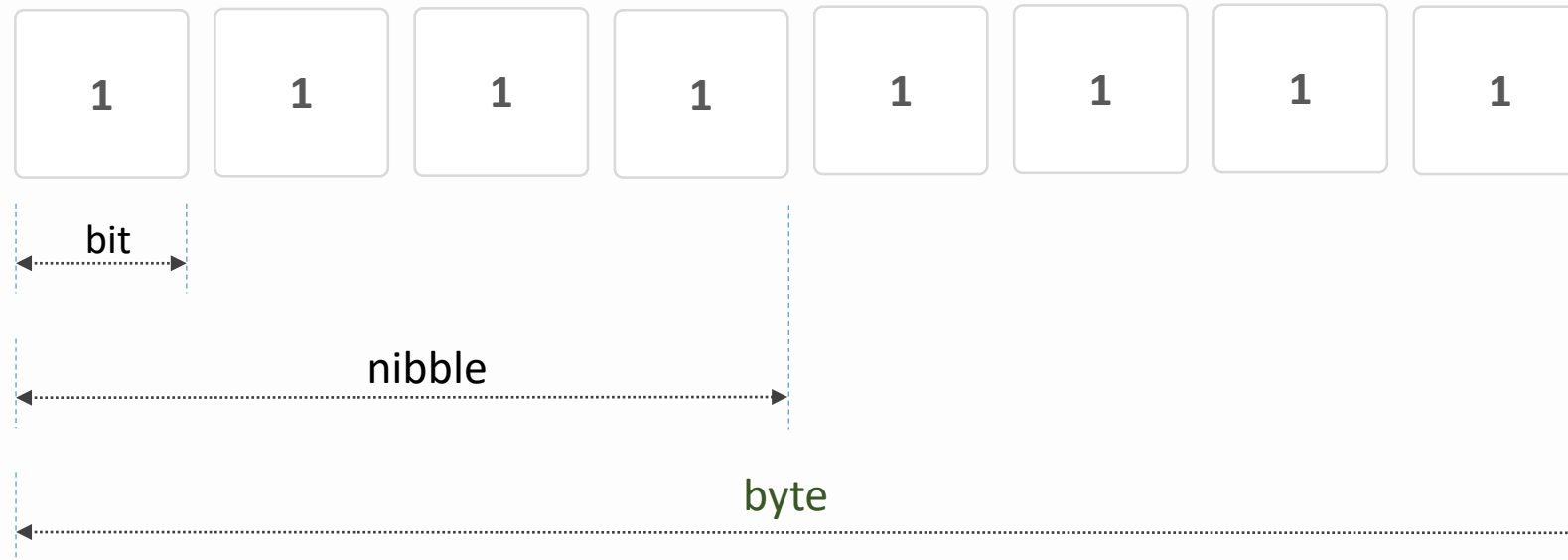


## 1. 프로그래밍 기본 원리





## 1. 프로그래밍 기본 원리

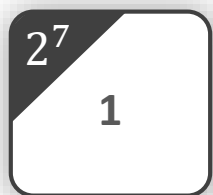


$$8\text{bits} = 2\text{nibble} = 1\text{byte}$$

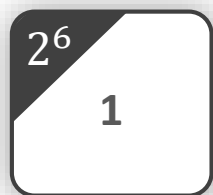
1byte = 데이터가 저장되는 최소 단위



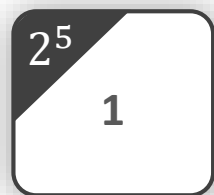
## 1. 프로그래밍 기본 원리



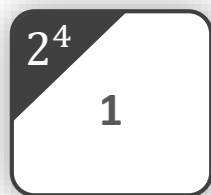
$$2^7 \times 1$$



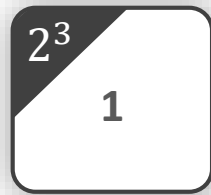
$$2^6 \times 1$$



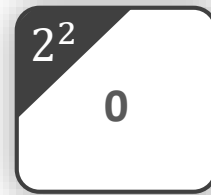
$$2^5 \times 1$$



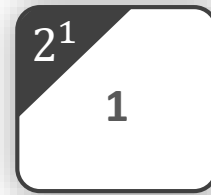
$$2^4 \times 1$$



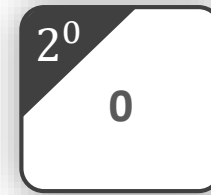
$$2^3 \times 1$$



$$2^2 \times 0$$



$$2^1 \times 1$$

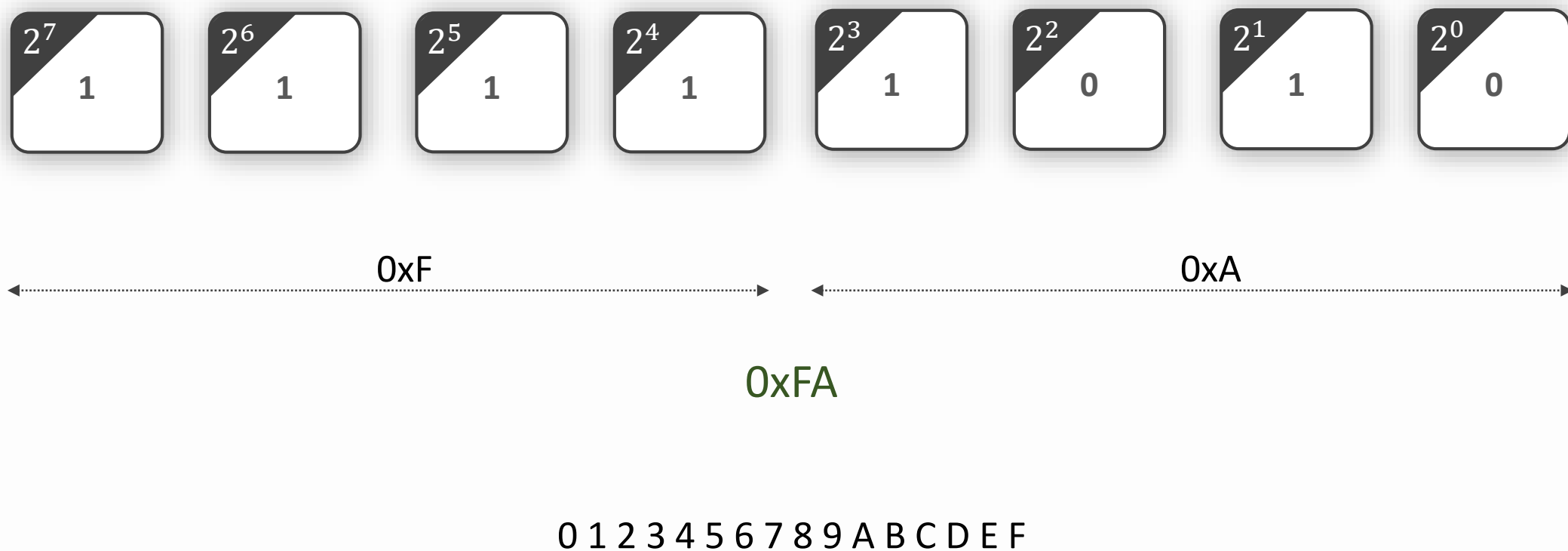


$$2^0 \times 0$$

250

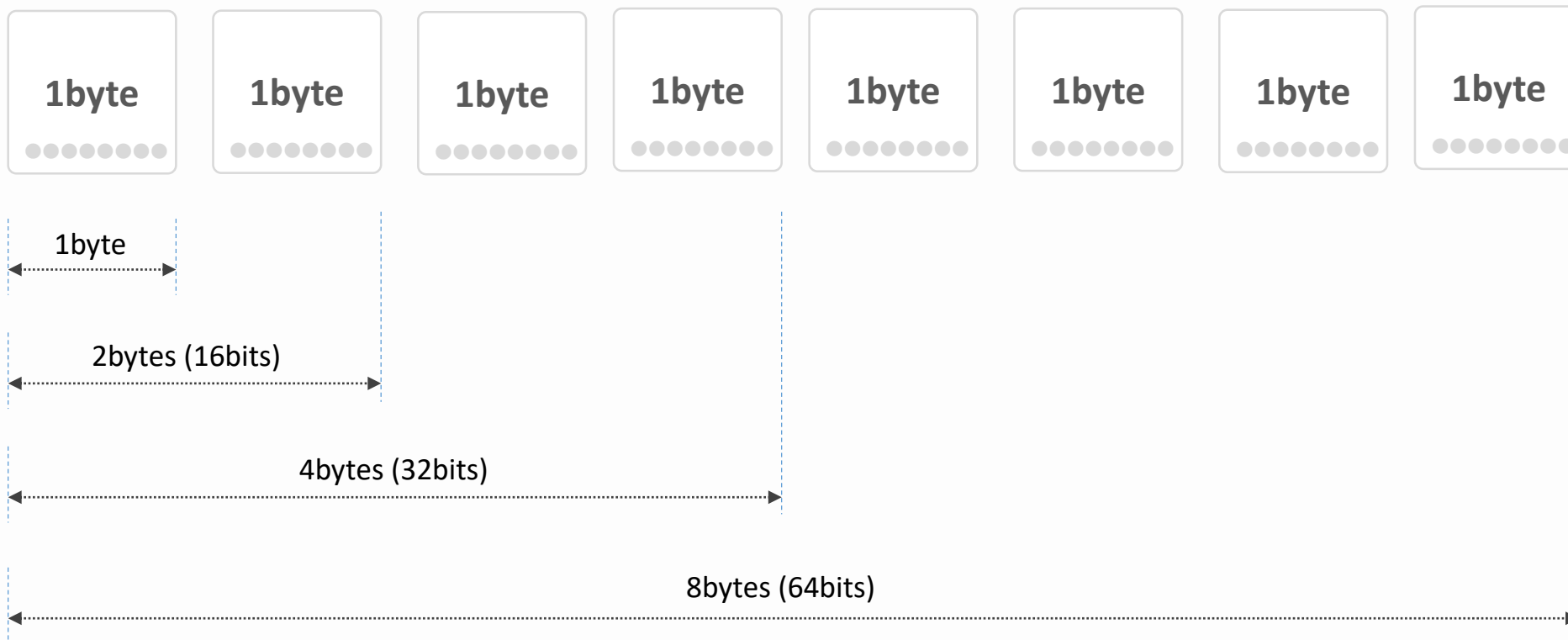


## 1. 프로그래밍 기본 원리



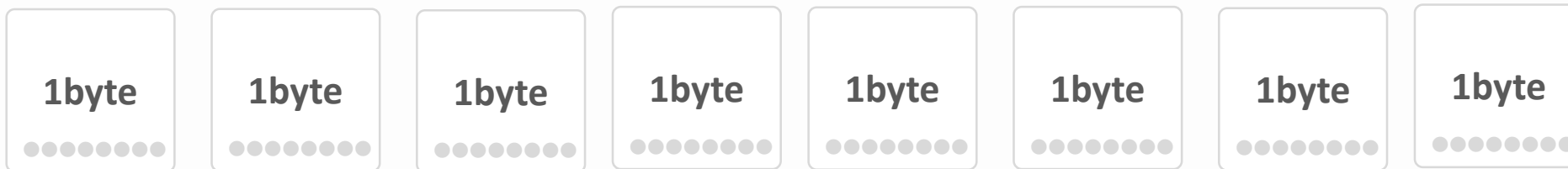


## 1. 프로그래밍 기본 원리





## 1. 프로그래밍 기본 원리



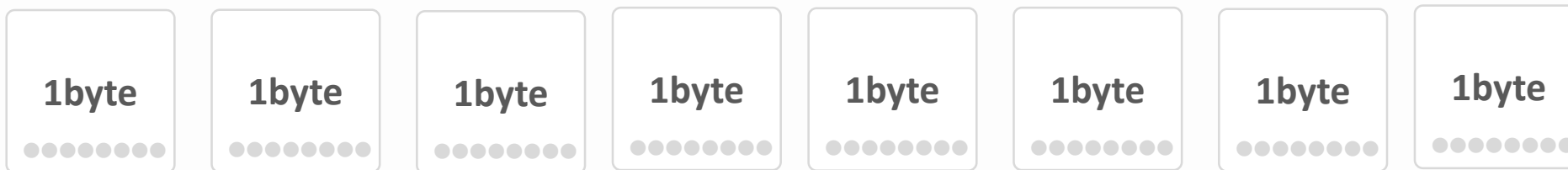
휘발성



비휘발성



## 1. 프로그래밍 기본 원리



8bits = 1Byte

1000bytes = 1KByte

1000Kbytes = 1MByte

1000Mbytes = 1GByte

1000Gbytes = 1TByte



## 1. 프로그래밍 기본 원리

컴퓨터는 0과 1만 이해

비트의 확장을 통해 데이터를 표현

8bits = 2nibble = 1byte (데이터 저장 최소 단위)

바이트의 확장을 통해 더 큰수를 표현

컴퓨터 공학에 편리한 16진수

:: 2강 관련 영상 및 정보

2진수와 16진수의 이해 [https://youtu.be/IZKej5s3T\\_w](https://youtu.be/IZKej5s3T_w)

문자를 처리하는 프로그래밍 원리 <https://youtu.be/apZFcOfAr5w>

시글사는 개발자의  
C언어 이야기

# 2. 언어의 특징과 구조

시글사는 개발자의  
C언어 이야기



## 2. C언어의 특징과 구조

1970년 - B 언어 개발 ([켄 톰프슨](#))

1972년 - 벨 연구소 (Bell Laboratories) 에 있는 [Dennis Ritchie](#)가 B의 후속으로 C 개발

1983년 - 미국 국가 표준 협회(ANSI, American National Standards Institute)에서  
짐 브로디(Jim Brodie) 주축으로 X3J11 위원회 소집

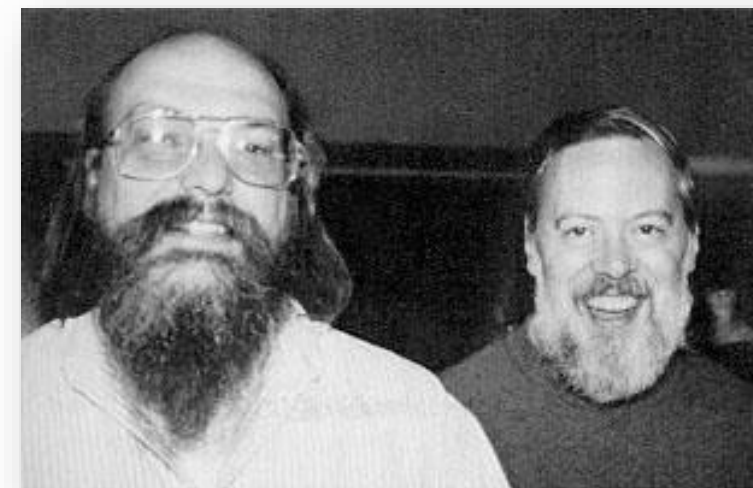
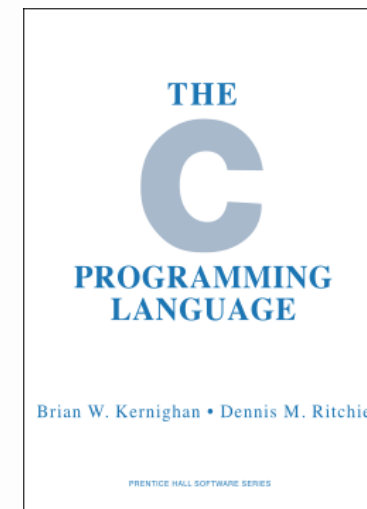
1983년 12월 14일 - ANSI X3.159-1989 라는 공식명칭으로 C 언어 표준 지정

1999년 - C99 표준안이 ISO/IEC 9899:1999라는 명칭으로 출간됨

2000년 5월 - ANSI의 표준으로 C99가 채택됨

2011년 - 12월 8일 C11 표준안이 ISO/IEC 9899:2011라는 명칭으로 출간됨

2018년 - C17 표준안이 ISO/IEC 9899:2018이라는 명칭으로 출간됨



\*출처 : [https://ko.wikipedia.org/wiki/C\\_\(프로그래밍\\_언어\)](https://ko.wikipedia.org/wiki/C_(프로그래밍_언어))

## 2. C언어의 특징과 구조

very fast!

no virtual machine required

very small and simple!

eco system!

trust the programmer!

**TRUST ME**  
**I'M A PROGRAMMER**

## 2. C언어의 특징과 구조

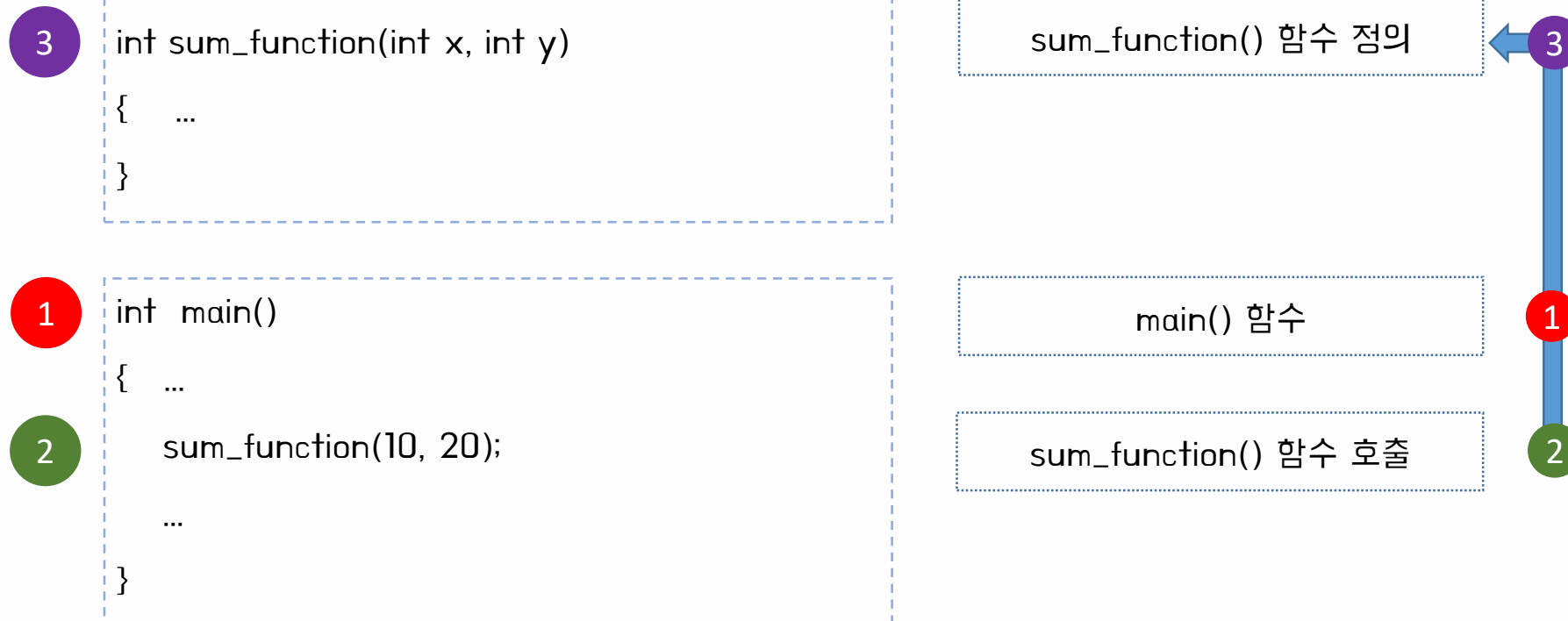
```
#include <stdio.h>
```

```
데이터형 함수명()  
{  
    함수호출();  
    return 데이터값;  
}
```



```
int main()  
{  
    printf("hello world\n");  
    return 0;  
}
```

## 2. C언어의 특징과 구조





## 2. C언어의 특징과 구조



## 2. C언어의 특징과 구조

함수 지향적 언어

절차 지향적 언어

컴파일 언어

:: C언어 강의 관련 영상 및 정보

Makefile 시작하기 <https://youtu.be/jnJL6ppn26Q>

Ctags 시작하기 <https://youtu.be/uYZ0WAm5rWE>

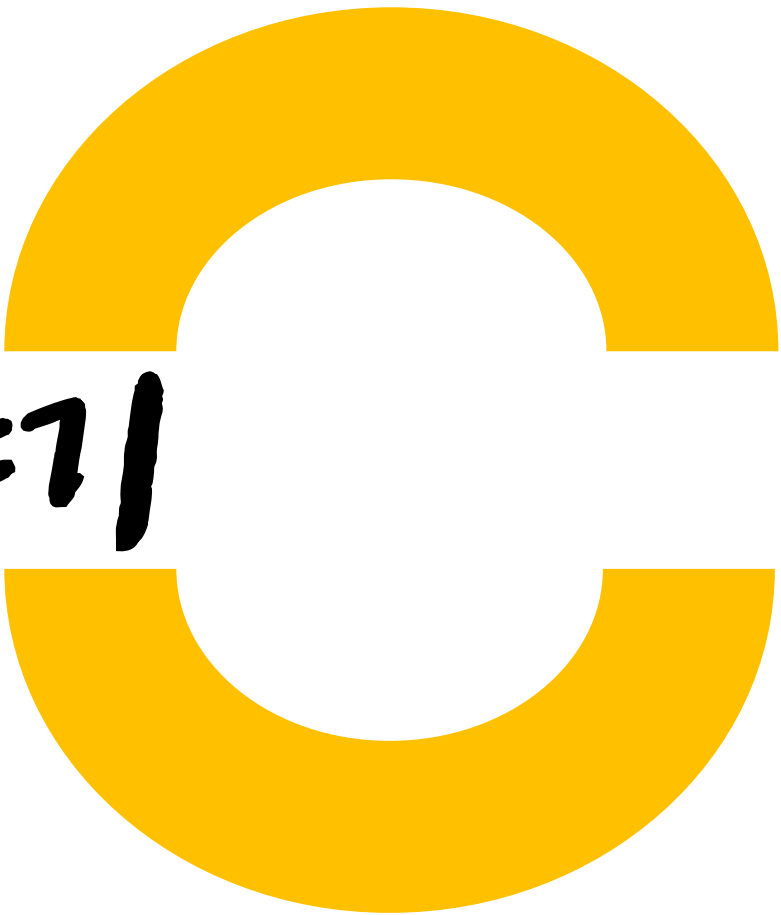
Vi(Vim) 시작하기 [https://youtu.be/GWo\\_MxMIJJ4](https://youtu.be/GWo_MxMIJJ4)

시글사는 개발자의  
**C언어 이야기**



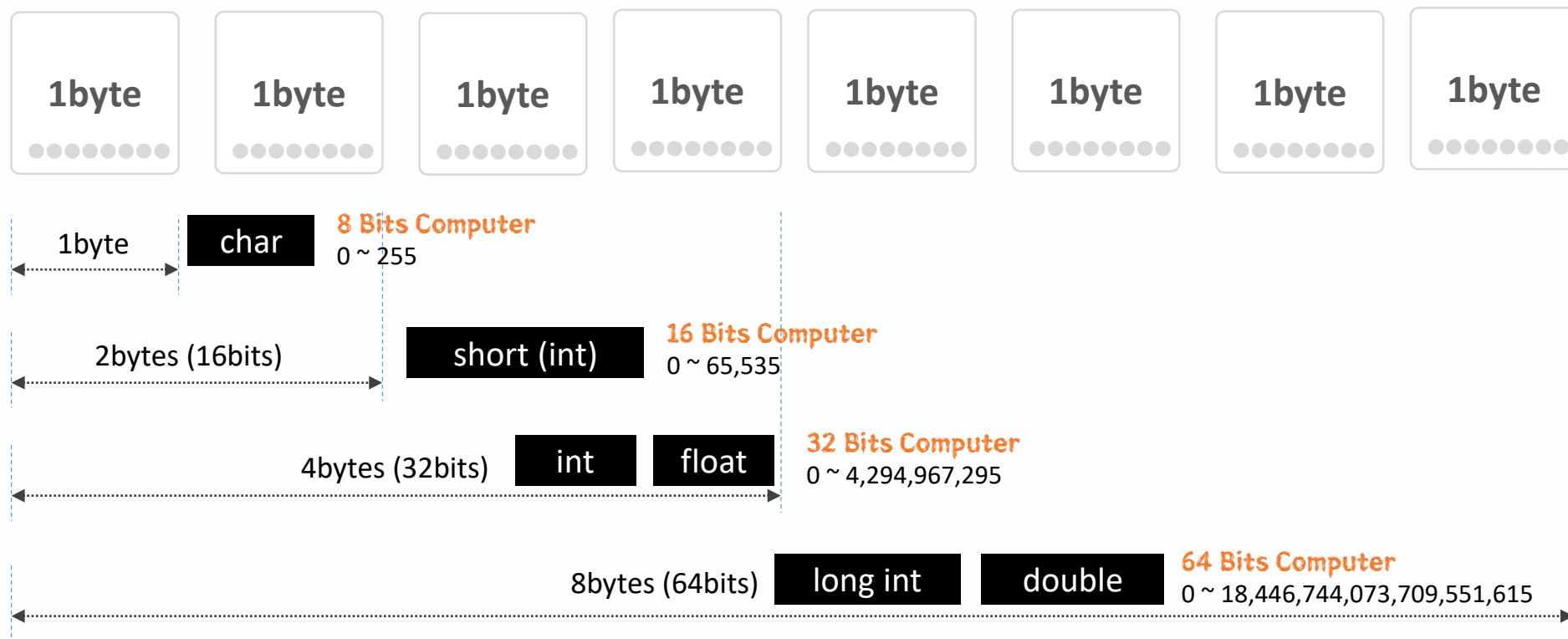
3. 데이터형과 변수

시글사는 개발자의  
C언어 이야기



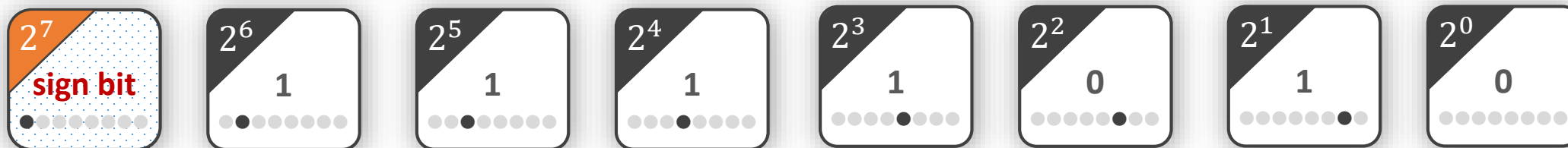


### 3. 데이터 형과 변수



\* `int/float`의 사이즈는 컴파일러와 OS(32bits/64bits)에 따라 달라질 수 있다.

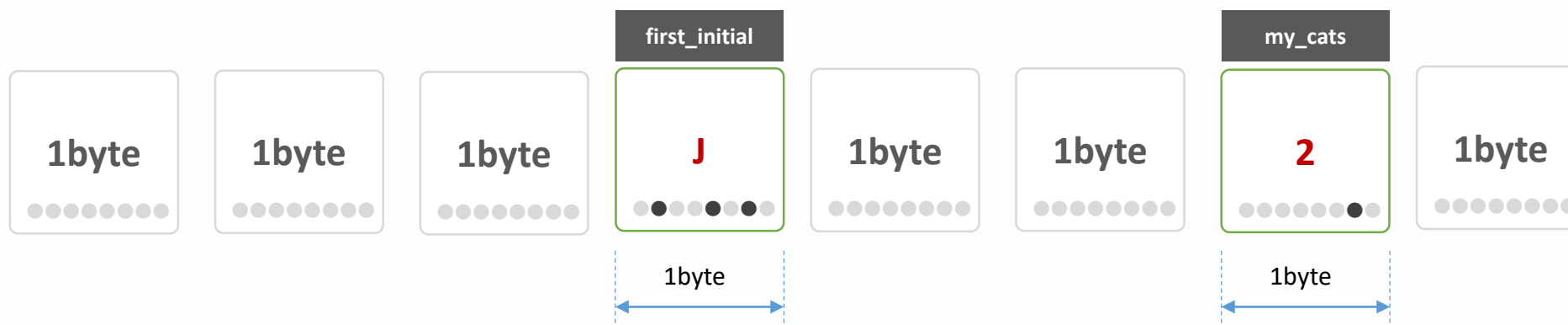
## 3. 데이터 형과 변수



| Sign Bit ON  | Sign Bit OFF                         |
|--|--------------------------------------|
| char == signed char  | unsigned char                        |
| short == signed short<br>== short int<br>== signed short int | unsigned short<br>unsigned short int |
| int = signed signed  | unsigned int                         |

\* float / double은 signed bit를 선택할 수 없다.

### 3. 데이터 형과 변수

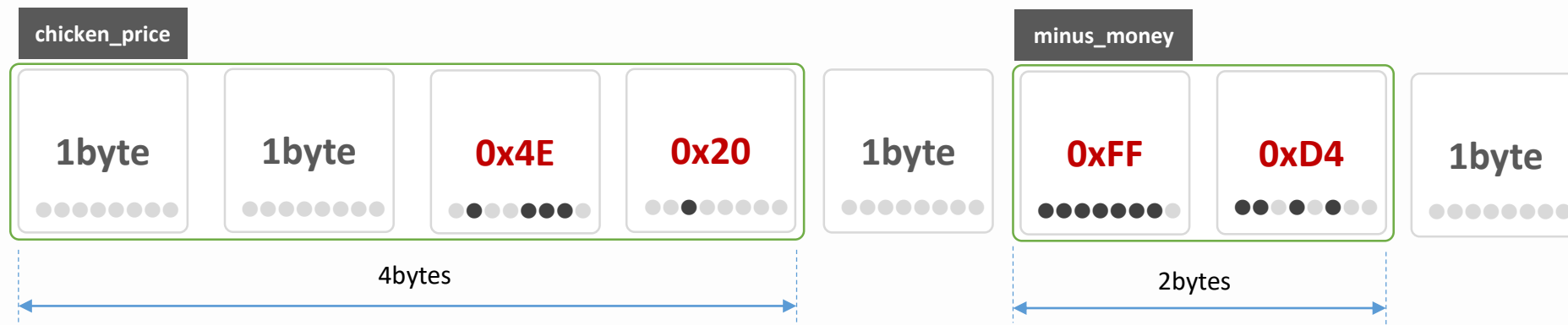


`char` 변수명 = 데이터(값);

ex) `char first_initial = 'J';`

ex) `char my_cats = 2;`

## 3. 데이터 형과 변수



int 변수명 = 데이터(값);

ex) int chicken\_price = 20000;

short int 변수명 = 데이터(값);

ex) short int minus\_money = -300;

### 3. 데이터 형과 변수

\* 상수 : 변하지 않는, 변할 수 없는 불변 데이터를 정의

```
const int MAX_USER = 100;
```

```
#define MAX_USER 100
```

```
char *str = "Hello, world";
```



### 3. 데이터 형과 변수

변수의 유효 범위

- 지역 변수 (local variable) : stack area, initial, lifetime(program)
- 전역 변수 (global variable) : local variable, stack area, un-initial, lifetime(scope)
- 정적 변수 (static variable) : static variable, data area, lifetime(program)

```
int var1; // global variable
```

```
int func() {
```

```
    int var2; // local variable
```

```
    static int var3; // static variable
```

```
}
```

### 3. 데이터 형과 변수

#### [Must]

변수명은 대소문자를 구분한다.

변수명은 \_, 영문자(대소문자)로 시작하여야 한다.

변수명은 C언어에서 예약된 키워드를 사용할 수 없다.

변수 선언과 초기화 과정을 구분한다.


변수는 유효 범위(scope or lifetime)를 가진다.

#### [Should]

변수명은 그 의미를 이해할 수 있게 작성되어야 한다.

C언어의 변수명은 주로 스네이크 표기법을 사용한다.

C언어의 상수명은 대문자로 표기한다.

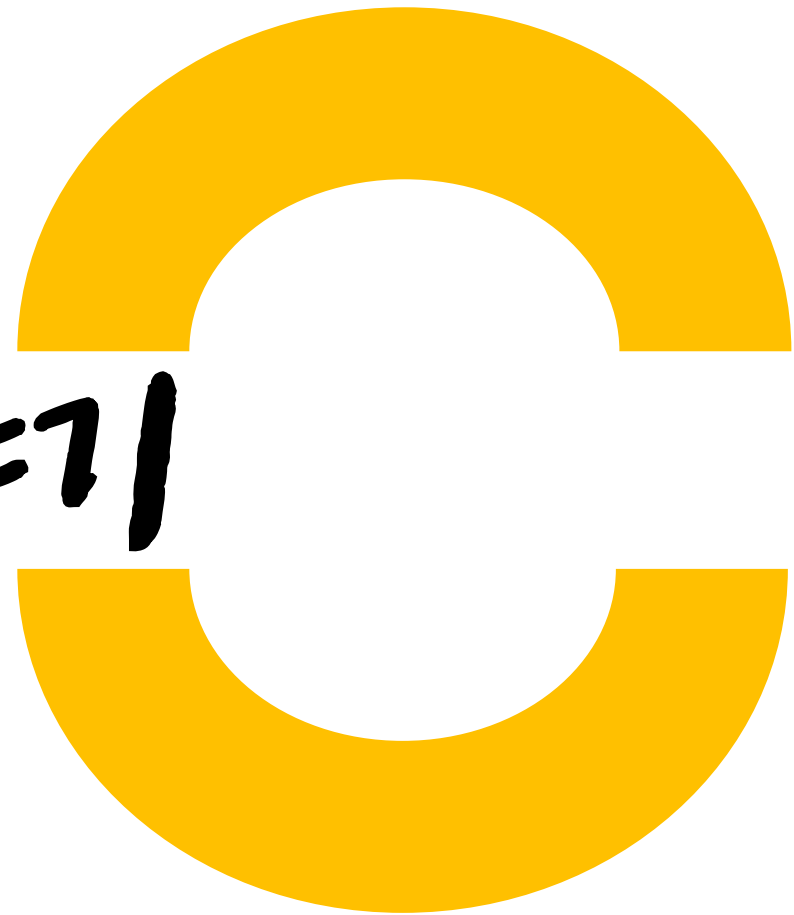


시글사는 개발자의  
C언어 이야기

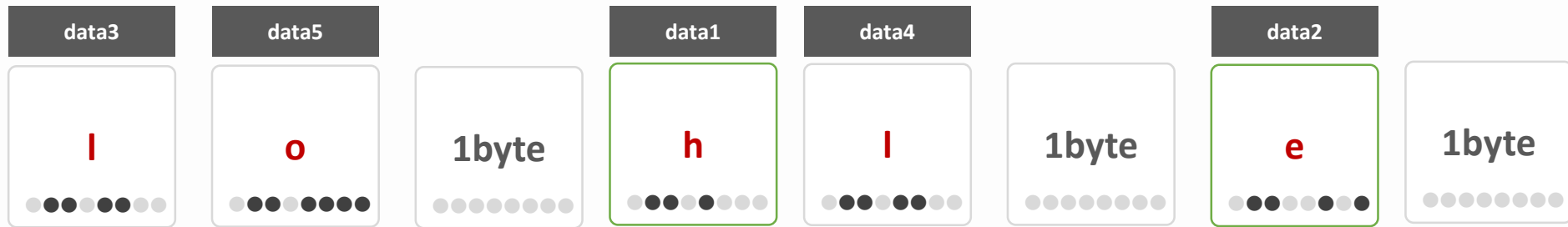


## 4. 문자와 문자열

시글사는 개발자의  
C언어 이야기



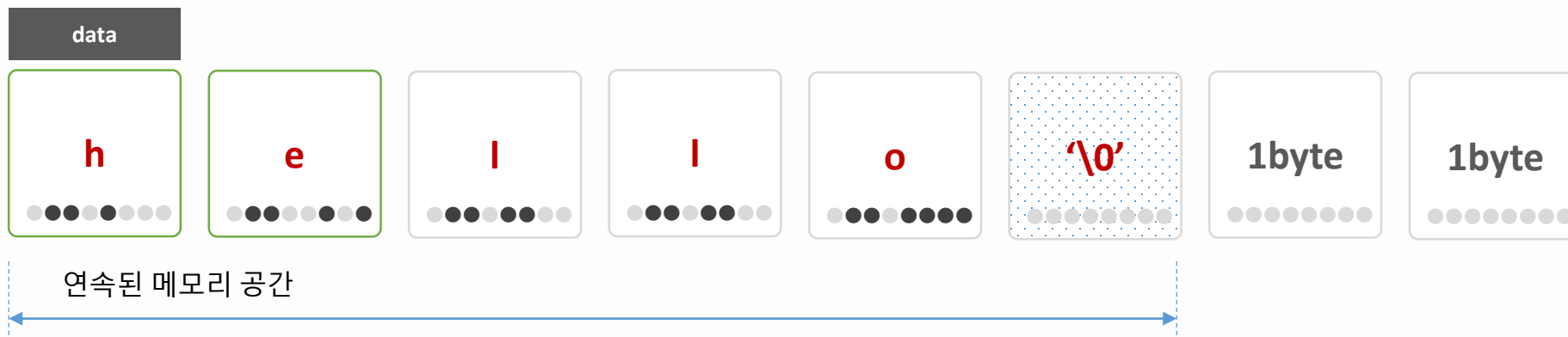
## 4. 문자와 문자열



```
char data1 = 'h';  
char data2 = 'e';  
char data3 = 'l';  
char data4 = 'l';  
char data5 = 'o';
```

```
printf("%c %c %c %c %c /n", data1, data2, data3, data4, data5);
```

## 4. 문자와 문자열



```
char data[6] = "hello";
```

```
printf("%s/n", data);
```

## 4. 문자와 문자열

## ASCII TABLE

| Decimal | Hex | Char                   | Decimal | Hex | Char    | Decimal | Hex | Char | Decimal | Hex | Char  |
|---------|-----|------------------------|---------|-----|---------|---------|-----|------|---------|-----|-------|
| 0       | 0   | [NULL]                 | 32      | 20  | [SPACE] | 64      | 40  | @    | 96      | 60  | `     |
| 1       | 1   | [START OF HEADING]     | 33      | 21  | !       | 65      | 41  | A    | 97      | 61  | a     |
| 2       | 2   | [START OF TEXT]        | 34      | 22  | "       | 66      | 42  | B    | 98      | 62  | b     |
| 3       | 3   | [END OF TEXT]          | 35      | 23  | #       | 67      | 43  | C    | 99      | 63  | c     |
| 4       | 4   | [END OF TRANSMISSION]  | 36      | 24  | \$      | 68      | 44  | D    | 100     | 64  | d     |
| 5       | 5   | [ENQUIRY]              | 37      | 25  | %       | 69      | 45  | E    | 101     | 65  | e     |
| 6       | 6   | [ACKNOWLEDGE]          | 38      | 26  | &       | 70      | 46  | F    | 102     | 66  | f     |
| 7       | 7   | [BELL]                 | 39      | 27  | '       | 71      | 47  | G    | 103     | 67  | g     |
| 8       | 8   | [BACKSPACE]            | 40      | 28  | (       | 72      | 48  | H    | 104     | 68  | h     |
| 9       | 9   | [HORIZONTAL TAB]       | 41      | 29  | )       | 73      | 49  | I    | 105     | 69  | i     |
| 10      | A   | [LINE FEED]            | 42      | 2A  | *       | 74      | 4A  | J    | 106     | 6A  | j     |
| 11      | B   | [VERTICAL TAB]         | 43      | 2B  | +       | 75      | 4B  | K    | 107     | 6B  | k     |
| 12      | C   | [FORM FEED]            | 44      | 2C  | ,       | 76      | 4C  | L    | 108     | 6C  | l     |
| 13      | D   | [CARRIAGE RETURN]      | 45      | 2D  | -       | 77      | 4D  | M    | 109     | 6D  | m     |
| 14      | E   | [SHIFT OUT]            | 46      | 2E  | .       | 78      | 4E  | N    | 110     | 6E  | n     |
| 15      | F   | [SHIFT IN]             | 47      | 2F  | /       | 79      | 4F  | O    | 111     | 6F  | o     |
| 16      | 10  | [DATA LINK ESCAPE]     | 48      | 30  | 0       | 80      | 50  | P    | 112     | 70  | p     |
| 17      | 11  | [DEVICE CONTROL 1]     | 49      | 31  | 1       | 81      | 51  | Q    | 113     | 71  | q     |
| 18      | 12  | [DEVICE CONTROL 2]     | 50      | 32  | 2       | 82      | 52  | R    | 114     | 72  | r     |
| 19      | 13  | [DEVICE CONTROL 3]     | 51      | 33  | 3       | 83      | 53  | S    | 115     | 73  | s     |
| 20      | 14  | [DEVICE CONTROL 4]     | 52      | 34  | 4       | 84      | 54  | T    | 116     | 74  | t     |
| 21      | 15  | [NEGATIVE ACKNOWLEDGE] | 53      | 35  | 5       | 85      | 55  | U    | 117     | 75  | u     |
| 22      | 16  | [SYNCHRONOUS IDLE]     | 54      | 36  | 6       | 86      | 56  | V    | 118     | 76  | v     |
| 23      | 17  | [ENG OF TRANS. BLOCK]  | 55      | 37  | 7       | 87      | 57  | W    | 119     | 77  | w     |
| 24      | 18  | [CANCEL]               | 56      | 38  | 8       | 88      | 58  | X    | 120     | 78  | x     |
| 25      | 19  | [END OF MEDIUM]        | 57      | 39  | 9       | 89      | 59  | Y    | 121     | 79  | y     |
| 26      | 1A  | [SUBSTITUTE]           | 58      | 3A  | :       | 90      | 5A  | Z    | 122     | 7A  | z     |
| 27      | 1B  | [ESCAPE]               | 59      | 3B  | ;       | 91      | 5B  | [    | 123     | 7B  | {     |
| 28      | 1C  | [FILE SEPARATOR]       | 60      | 3C  | <       | 92      | 5C  | \    | 124     | 7C  |       |
| 29      | 1D  | [GROUP SEPARATOR]      | 61      | 3D  | =       | 93      | 5D  | ]    | 125     | 7D  | }     |
| 30      | 1E  | [RECORD SEPARATOR]     | 62      | 3E  | >       | 94      | 5E  | ^    | 126     | 7E  | ~     |
| 31      | 1F  | [UNIT SEPARATOR]       | 63      | 3F  | ?       | 95      | 5F  | _    | 127     | 7F  | [DEL] |

## 4. 문자와 문자열

```
#include <string.h>
```

```
// 문자열 길이
```

```
size_t strlen(const char *s);
```

```
// 문자열 비교 (같으면 0, 다르면 0 이외의 값)
```

```
int strcmp(const char *s1, const char *s2);
```

```
int strncmp(const char *s1, const char *s2, size_t n);
```

```
// 문자열 복사
```

```
char *strcpy(char *dest, const char *src);
```

```
char *strncpy(char *dest, const char *src, size_t n);
```

```
int sprintf(char *str, const char *format, ...);
```

```
void *memset(void *s, int c, size_t n);    // sizeof
```

5. 연산자

시글사는 개발자의

C언어 이야기

## 5. 연산자

| 우선 순위 | 연산자  | 결합성          |
|-------|--|--------------|
| 1     | () [] -> . ++(후위) --(후위)                                 | → (왼쪽에서 오른쪽) |
| 2     | sizeof &(주소) ++(전위) --(전위) ~ !<br>*(역참조) +(부호) -(부호) 형변환 | ← (오른쪽에서 왼쪽) |
| 3     | *(곱셈) /(나눗셈) %(나머지)                                      | → (왼쪽에서 오른쪽) |
| 4     | +(덧셈) -(뺄셈)  | → (왼쪽에서 오른쪽) |
| 5     | << >>  | → (왼쪽에서 오른쪽) |
| 6     | < <= >= >  | → (왼쪽에서 오른쪽) |
| 7     | == !=  | → (왼쪽에서 오른쪽) |
| 8     | &(비트연산)  | → (왼쪽에서 오른쪽) |
| 9     | ^  | → (왼쪽에서 오른쪽) |
| 10    |  | → (왼쪽에서 오른쪽) |
| 11    | &&   | → (왼쪽에서 오른쪽) |
| 12    |  | → (왼쪽에서 오른쪽) |
| 13    | ?(삼항)  | ← (오른쪽에서 왼쪽) |
| 14    | = += *= /= %= &=  = <<= >>=                              | ← (오른쪽에서 왼쪽) |
| 15    | ,(콤마)  | → (왼쪽에서 오른쪽) |

출처:

<http://blog.naver.com/PostView.nhn?blogId=rnsu2011&logNo=220653971062&parentCategoryNo=&categoryNo=11&viewDate=&isShowPopularPosts=false&from=postView>



## 5. 연산자

problem = a \* b + 3 + 7 / 2

solution = ( a \* (b + 3) ) + (7 / 2)



## 5. 연산자

[산술 연산자]

```
int a = 3;
```

```
int b = 2;
```

```
int result = a + b;
```

```
int result = a - b;
```

```
int result = a * b;
```

```
int result = a / b;
```

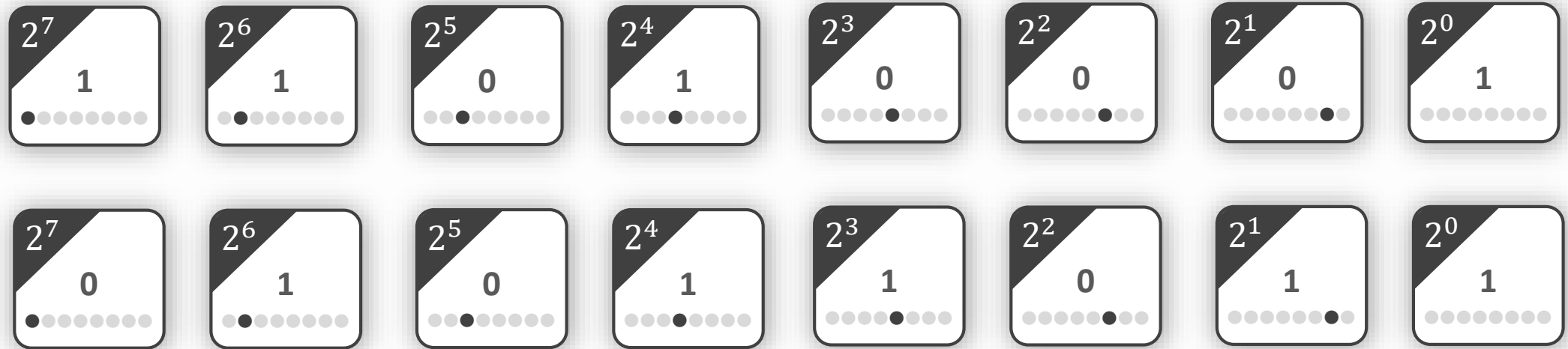
```
int result = a % b
```

```
int result = a + b * a;
```



## 5. 연산자

&  
or  
|



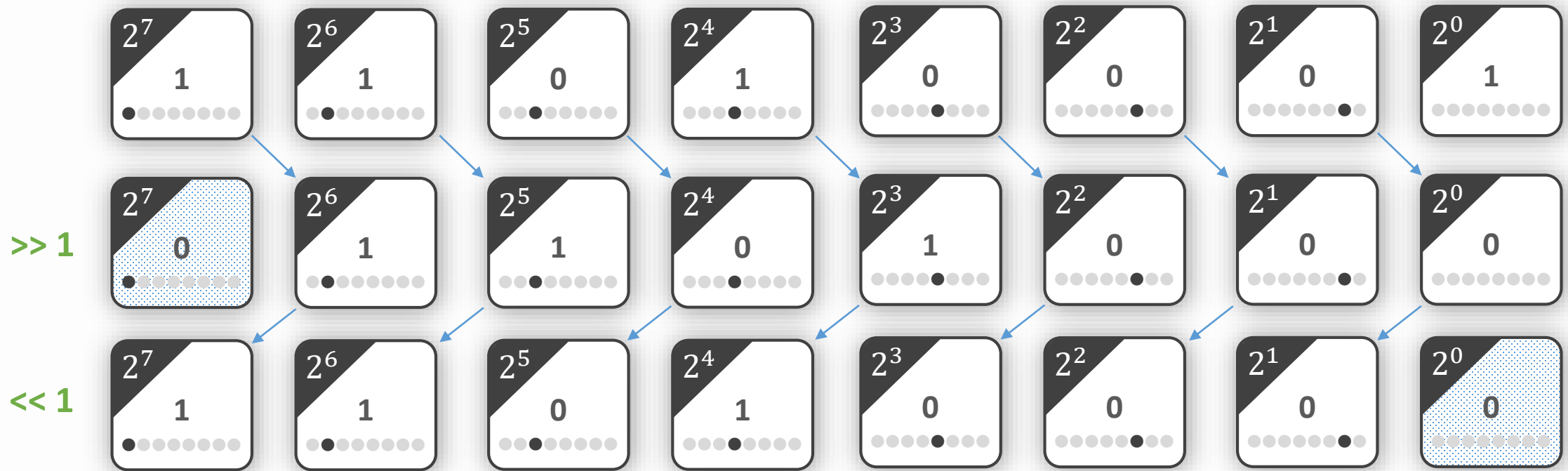
[논리 연산자]

&연산 : 모두 참(1)인 경우만 참

| 연산 : 하나라도 참(1)인 경우는 참

~연산 : 참과 거짓을 반대로 변경 (Switch On/Off)

## 5. 연산자



[비트 연산]

char data = 0b11111111;

data = data &gt;&gt; 1;      // 비트를 오른쪽으로 이동

data = data &lt;&lt; 1;      // 비트를 왼쪽으로 이동

## 5. 연산자

[증감 연산자]

```
int a = 1;
```

```
int result = ++a; // 전위 증가 :: a를 +1 증가하고 그 결과를 result에 할당
```

```
int result = a++; // 후위 증가 :: a의 현재값을 result에 할당하고 a는 +1 증가
```

```
int result = --a; // 전위 감소 :: a를 -1 감소하고 그 결과를 result에 할당
```

```
int result = a--; // 후위 감소 :: a의 현재값을 result에 할당하고 a는 -1 감소
```

```
ex) int result = a++ + a;
```

```
ex) int result = --a + a;
```

## 5. 연산자

[비교 연산자]

```
int a = 3;
```

```
int b = 2;
```

$a > b$  :  $a$ 가  $b$ 보다 크다면 1(참), 아니면 0(거짓)

$a < b$  :  $b$ 가  $a$ 보다 크다면 1(참), 아니면 0(거짓)

$a == b$  : 같은 경우 1(참), 다른 경우 0(거짓)

$a != b$  : 다른 경우 1(참), 같은 경우 0(거짓)

## 5. 연산자

### [삼항 연산자]

```
int a = 3;
```

```
int b = 2;
```

```
int result = (a > b) ? 1 : 0;
```

### [단항 연산자]

```
int a = 1;
```

```
a = !a;    // a = 0
```

```
a = !a;    // a = 1
```

### [대입 연산자]

```
int a=3, b=2;
```

```
a += b;    // a = a + b
```

```
a -= b;    // a = a - b
```

```
a *= b;    // a = a * b
```

```
a /= b;    // a = a / b
```

```
a %= b;    // a = a % b
```

## 6. 조건문

시글사는 개발자의

C언어 이야기

## 6. 조건문

```
if (조건1 == true) {  
    ... // 조건1이 참인 경우 실행  
} else if (조건2 == true) {  
    ... // 조건2가 참인 경우 실행  
} else {  
    ... // 모든 조건이 거짓인 경우 실행  
}
```



## 6. 조건문

```
switch (점수데이터) {  
    case 1:    // 조건이 1인 경우  
    {  
        ...  
        break;  
    }  
    case 2:    // 조건이 2인 경우  
    {  
        ...  
        break;  
    }  
    default:   // 모든 조건이 일치하지 않는 경우  
    {  
        ...  
        break;  
    }  
}
```

7. 반박문

시글사는 개발자의

C언어 이야기

## 7. 반복문

```
// for 반복문 형식
for (초기화; 조건문; 증감문)
{
    ...
}
```

```
// for문으로 ASCII 코드 출력
for (int i=0; i<128; i++)
{
    printf("%d (%#x) : %c", i, i, i);
    if ((i % 10) == 0) printf("/n");
}
```

## 7. 반복문

```
// while 반복문 형식
```

```
while (조건문)
```

```
{
```

```
    ...
```

```
}
```

```
// while문으로 ASCII 코드 출력
```

```
char i=0;
```

```
while (i < 128)
```

```
{
```

```
    printf("%d (%#x) : %c", i, i, i);
```

```
    i++;
```

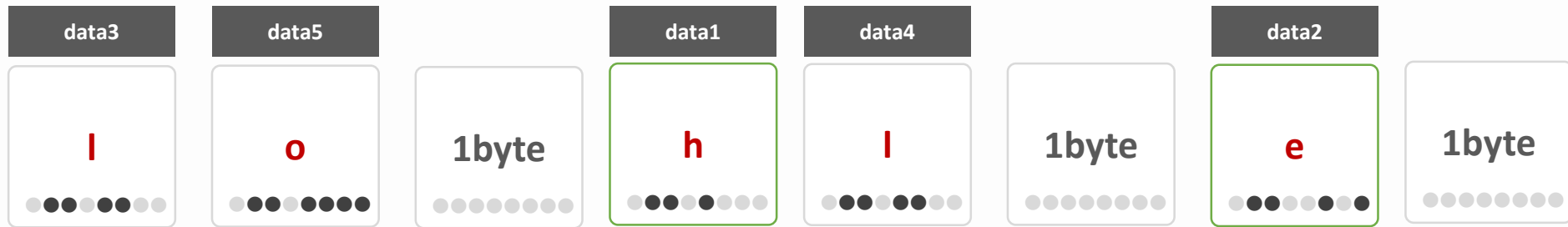
```
}
```

8. 배움

시글사는 개발자의

C언어 이야기

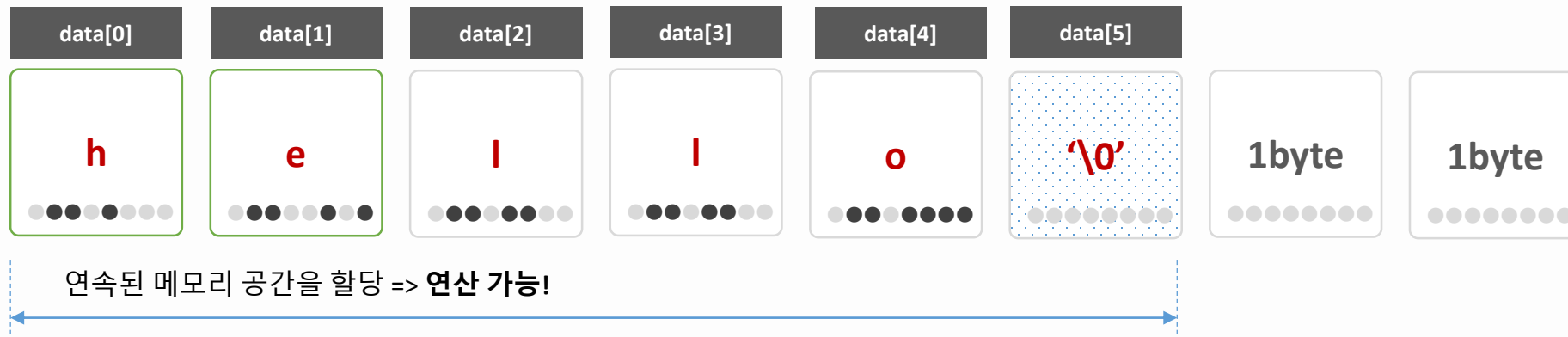
## 8. 배열



```
char data1 = 'h';  
char data2 = 'e';  
char data3 = 'l';  
char data4 = 'l';  
char data5 = 'o';
```

```
printf("%c %c %c %c %c /n", data1, data2, data3, data4, data5);
```

## 8. 배열

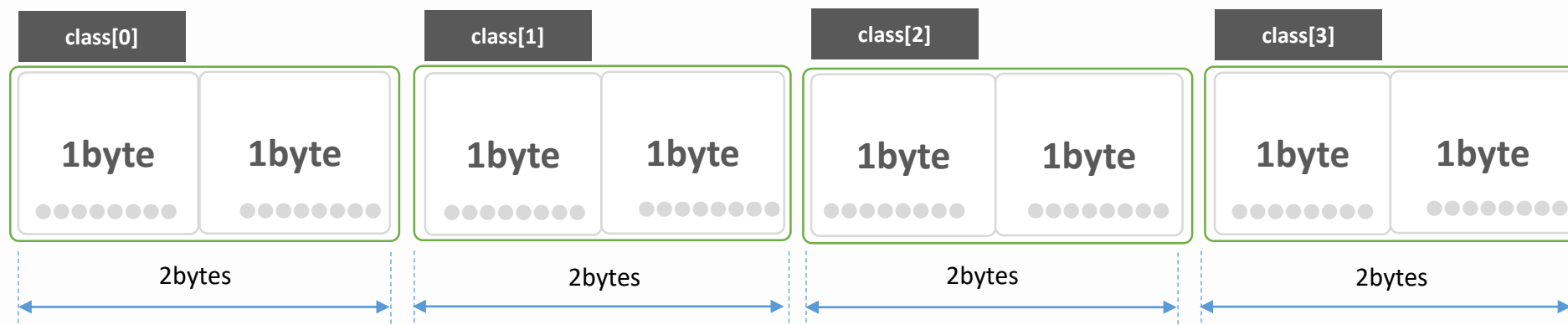


```
char data[6] = "hello";
```

```
printf("%c\n", data[1]);  
printf("%c\n", data[4]);  
printf("%s\n", data);
```



## 8. 배열



```
short 변수명[배열크기] = {초기값, };
```

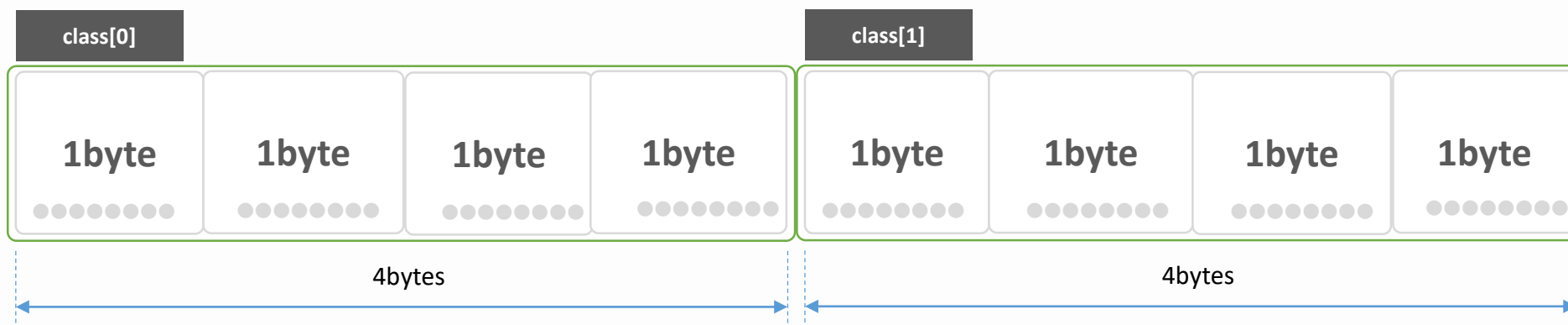
```
ex) short class[4] = {10, 20, 0, 0};
```

```
printf("class[0]:%d , class[1]:%d/n", class[0], class[1]);
```





## 8. 배열



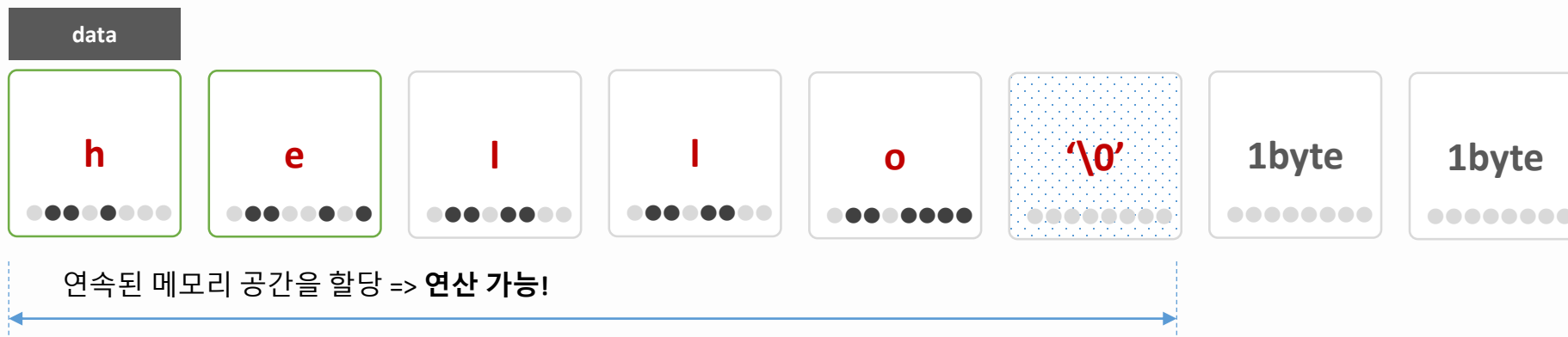
```
int 변수명[배열크기] = {초기값, };
```

```
ex) int class[2] = {10, 20};
```

```
printf("class[0]:%d , class[1]:%d/n", class[0], class[1]);
```



## 8. 배열



// 배열 이름 == 메모리 시작 주소

```
char data[8];
```

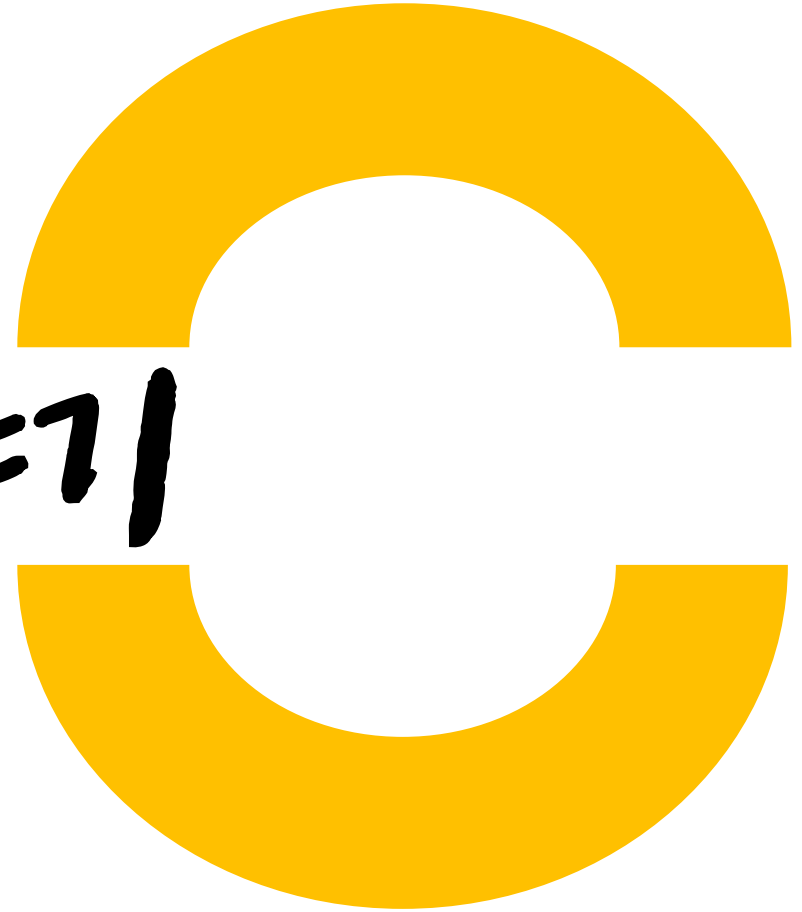
```
data == &data[0];
```

```
serial[0] == *(serial + 0);           // *(serial + (sizeof(char) * 0));
```

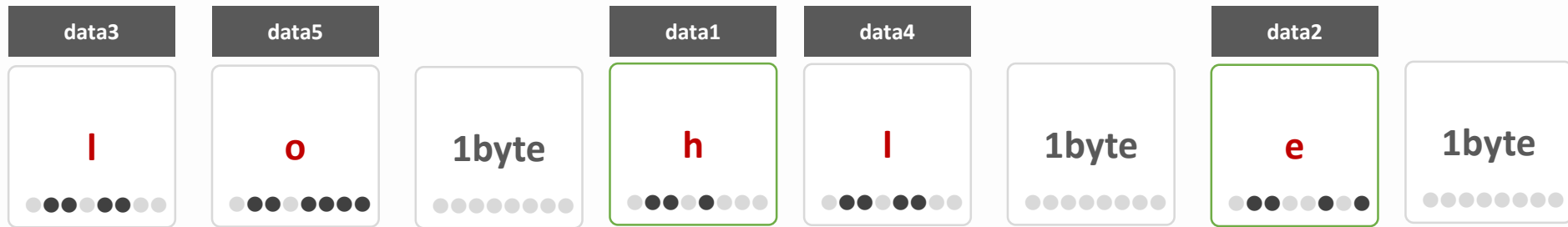
```
serial[1] == *(serial + 1);           // *(serial + (sizeof(char) * 1));
```

9. 포인트

시글사는 개발자의  
C언어 이야기



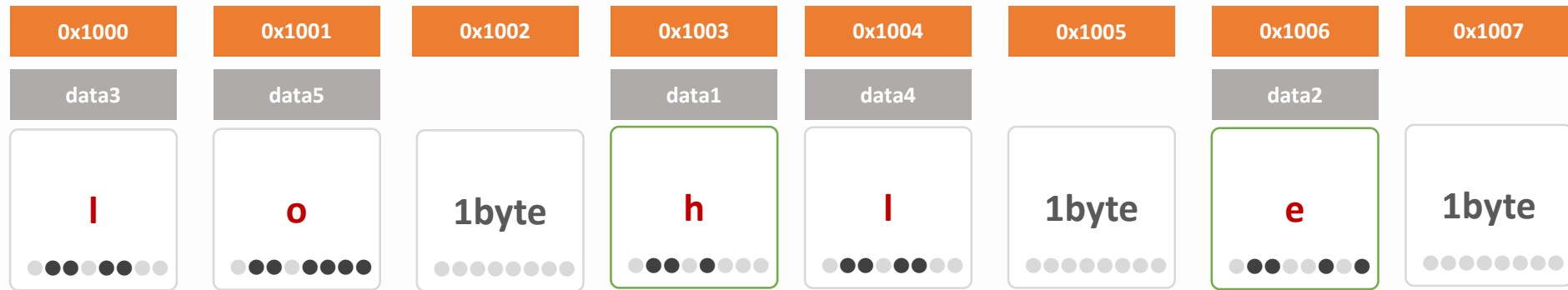
## 9. 포인터



```
char data1 = 'h';  
char data2 = 'e';  
char data3 = 'l';  
char data4 = 'l';  
char data5 = 'o';
```

```
printf("%c %c %c %c %c /n", data1, data2, data3, data4, data5);
```

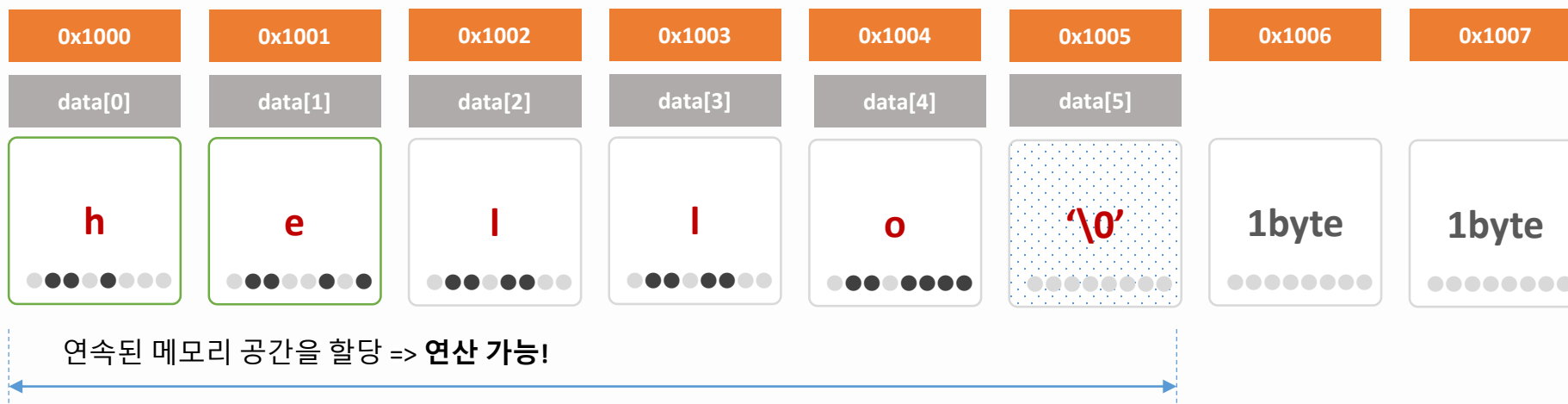
## 9. 포인터



```
char data1 = 'h';  
char data2 = 'e';  
char data3 = 'l';  
char data4 = 'l';  
char data5 = 'o';
```

```
printf("%c %c %c %c %c /n", data1, data2, data3, data4, data5);
```

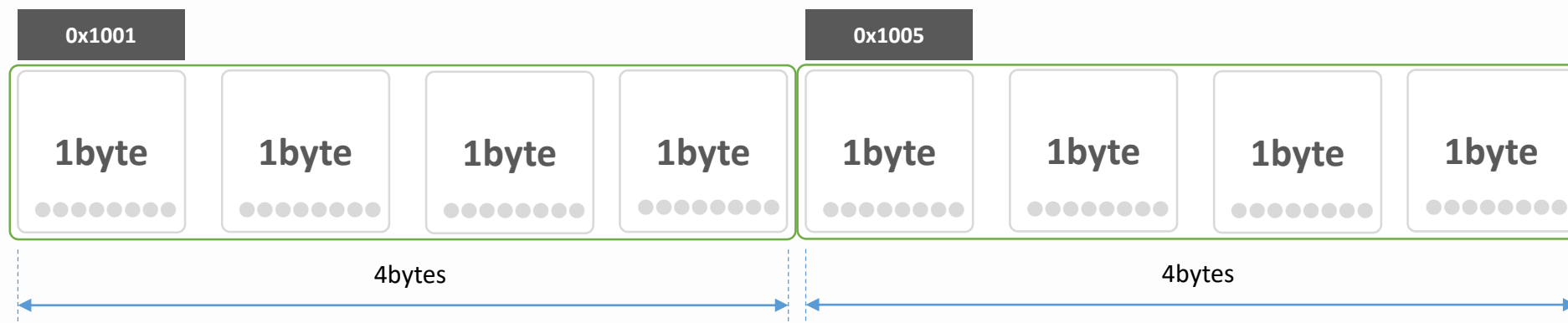
## 9. 포인터



```
char data[6] = "hello";
```

```
printf("%c\n", data[1]); // *(data + 1)
printf("%c\n", data[4]); // *(data + 4)
printf("%s\n", data);
```

## 9. 포인터



1. 일반적인 연산자에서는 곱하기

ex)  $10 * 2$

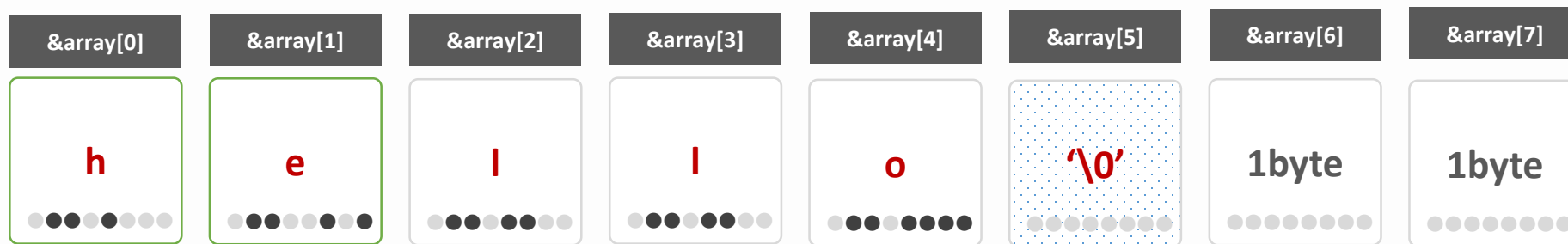
2. 변수 선언시 주소를 저장하는 포인터 변수

ex) `int *ptr = &data;`

3. 표현식에서는 포인터 변수에 저장된 값

ex) `int data = *ptr;`

## 9. 포인터



연속된 메모리 공간을 할당 => 연산 가능!

포인터와 배열은 동작 방식이 똑같다!

```
char array[8] = "hello";
```

```
char *ptr = array;
```

```
/* -----
```

```
ptr == array == &array[0]    // 주소
```

```
*ptr == *(ptr+0) == *array == array[0] == ptr[0] // 데이터
```

```
-----*/
```



## 9. 포인터

```
// call by reference
void proc_swap(int *x, int *y)
{
    // algorithm
}

int main()
{
    int a=5, b=10;
    proc_swap(&a, &b);
    return 0;
}
```

## 9. 포인터

// 동적 메모리 할당

```
char *ptr = (char*)malloc(100);
```

```
memset(ptr, 0, 100);
```

```
strcpy(ptr, "hello");
```

```
free(ptr);
```

```
char *ptr = (char*)calloc(1, 100);
```

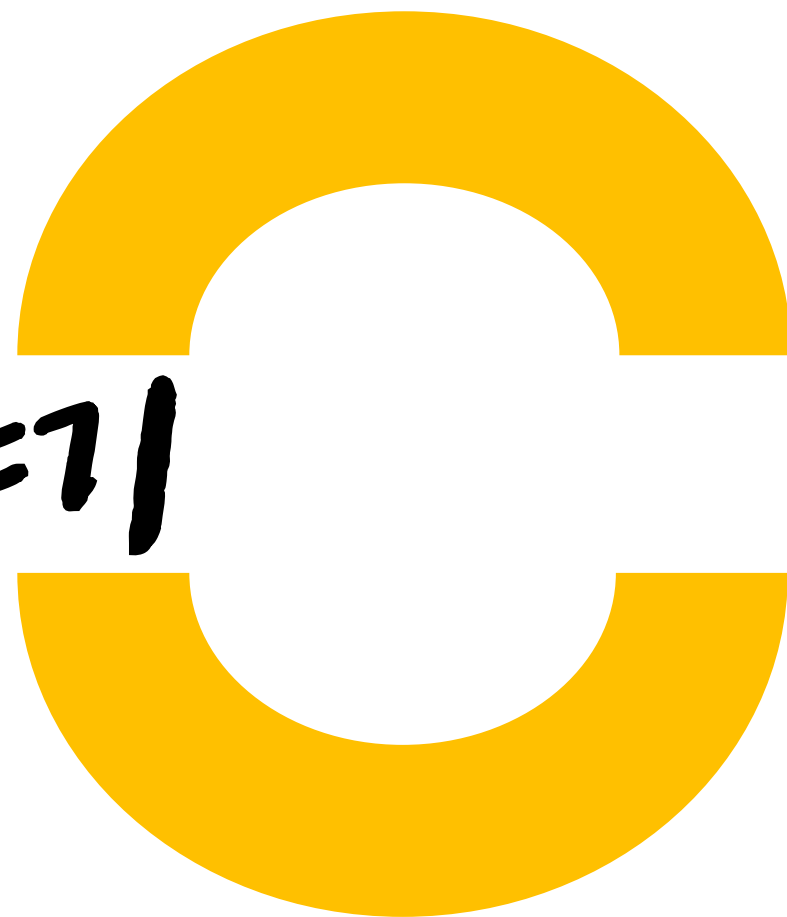
```
strcpy(ptr, "hello");
```

```
free(ptr);
```

10. 구조체

시글사는 개발자의

C언어 이야기





## 10. 구조체

### 고객 데이터 구조

```
{  
    고객 이름  
    고객 전화번호  
    고객 나이  
    고객 포인트  
};
```

### typedef struct

```
{  
    char name[20];        // 이름  
    char phone[15];       // 전화번호  
    int age;              // 나이  
    int point;            // 포인트  
} customer_data_t;
```



## 10. 구조체

// 방식#1 : 구조체 변수 선언

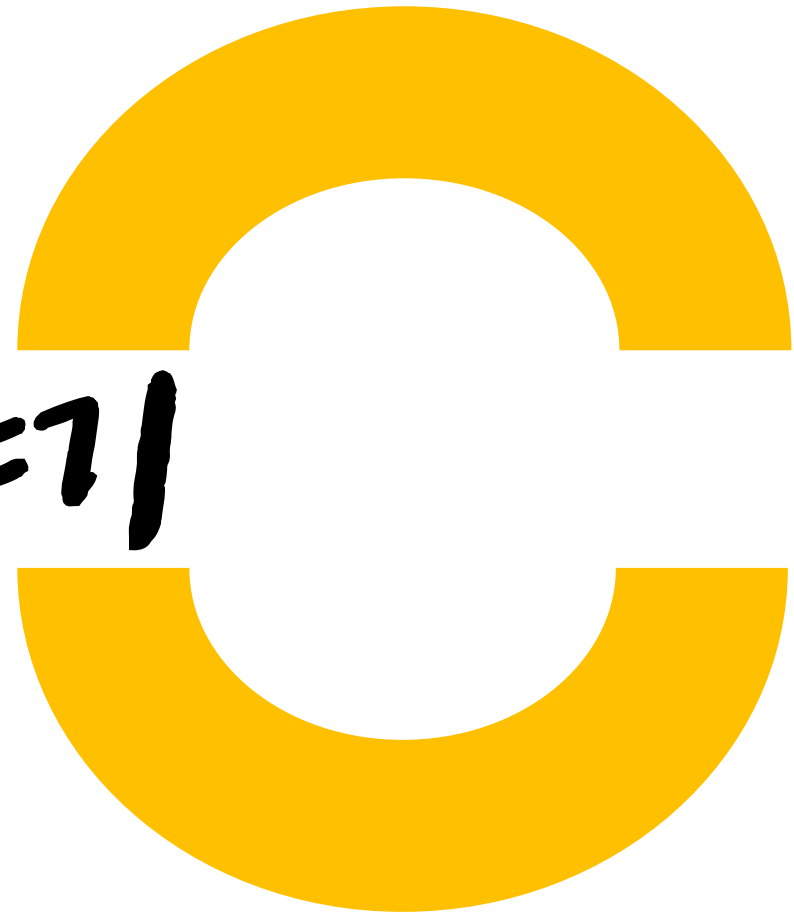
```
{  
  
    customer_data_t customerData;  
    strcpy(customerData.name, "jason");  
    strcpy(customerData.phone, "01011223344");  
    customerData.age = 22;  
    customerData.customerPoint = 30;  
  
    printf("name:%s/n", user.name);  
    printf("phone:%s/n", user.phone);  
    printf("age:%d/n", user.age);  
    printf("point:%d/n", user.point);  
}
```

// 방식#2 : 구조체 변수 선언 및 초기화

```
{  
  
    customer_data_t user = {  
        name: "jason",  
        phone: "01011223344",  
        age: 22,  
        point: 30,  
    };  
  
    printf("name:%s/n", user.name);  
    printf("phone:%s/n", user.phone);  
    printf("age:%d/n", user.age);  
    printf("point:%d/n", user.point);  
}
```

## 11. 함수포인트

시글사는 개발자의  
C언어 이야기





## 11. 함수 포인터

1. 함수도 메모리 공간의 주소를 할당한다.
2. 함수의 메모리 주소를 알면 변수처럼 활용이 가능하다.
3. 함수 포인터를 통해 함수 인자 전달이 가능하다.
4. 상태 천이 기법등에 많이 사용된다.

데이터타입 (\*함수 포인터 이름)();

int (\*proc\_func)();

## 11. 함수 포인터

```
void func_login(int id) {  
    printf("login function: %d id/n", id);  
}
```

```
int main() {  
    void (*func)() = func_login;  
    func(123);  
}
```





## 11. 함수 포인터

```
#include <stdio.h>

void func_login(int id) {
    printf("login function: %d id/n", id);
}

void func_order(int id) {
    printf("order function: %d id/n", id);
}

void func_logout(int id) {
    printf("logout function: %d id/n", id);
}

void (*my_func[3])() = {
    func_login,
    func_order,
    func_logout
};
```

```
enum {
    LOGIN_STATS,
    ORDER_STATS,
    LOGOUT_STATS
};

int main() {
    int id = 123;
    int stats = LOGIN_STATS;

    switch (stats) {
        case LOGIN_STATS:
            my_func[stats](id);
            break;
        case ORDER_STATS:
            my_func[stats](id);
            break;
        case LOGOUT_STATS:
            my_func[stats](id);
            break;
    }
}
```