프로그래밍을 이해하는 개발자의 올바른 자세

C Language

시골사는 개발자

CONTENTS

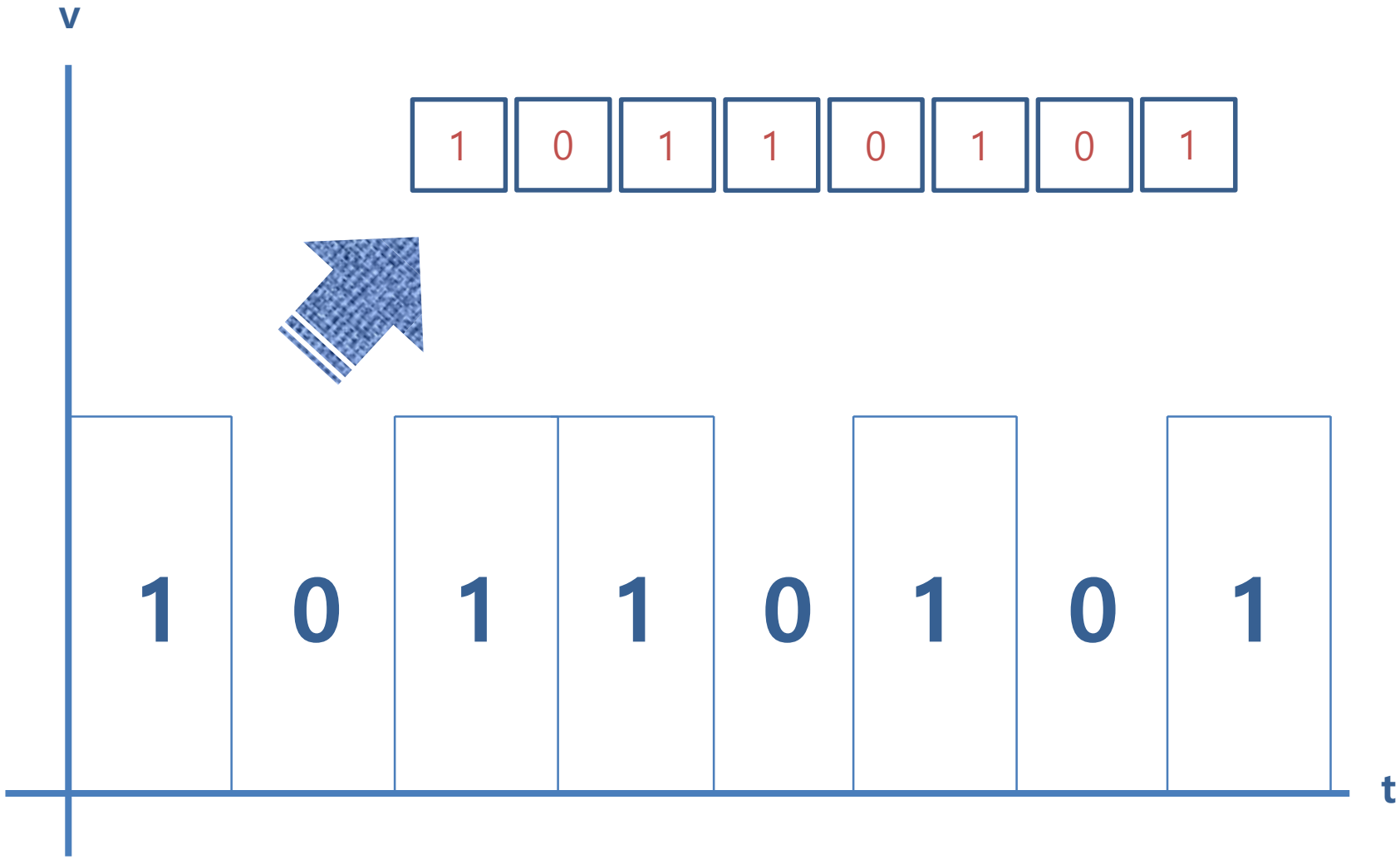
1-1

“컴퓨터는 과연 몇 가지 언어를 이해할까요?”

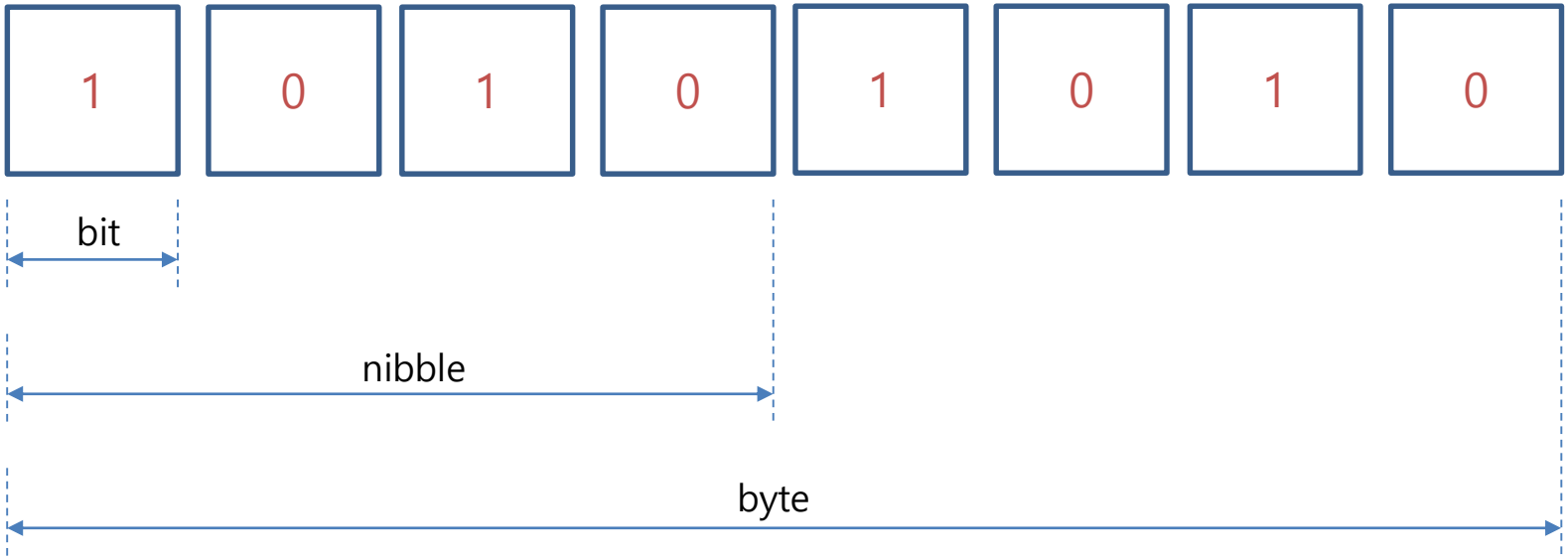
1-2

0 1

1-3



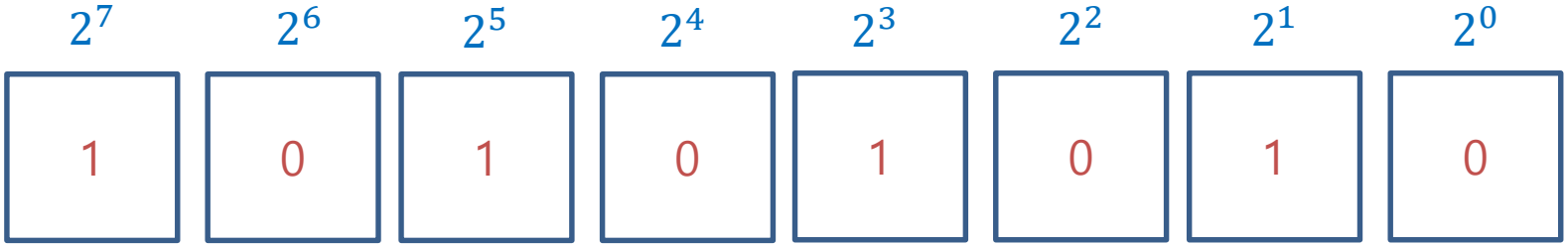
1-4



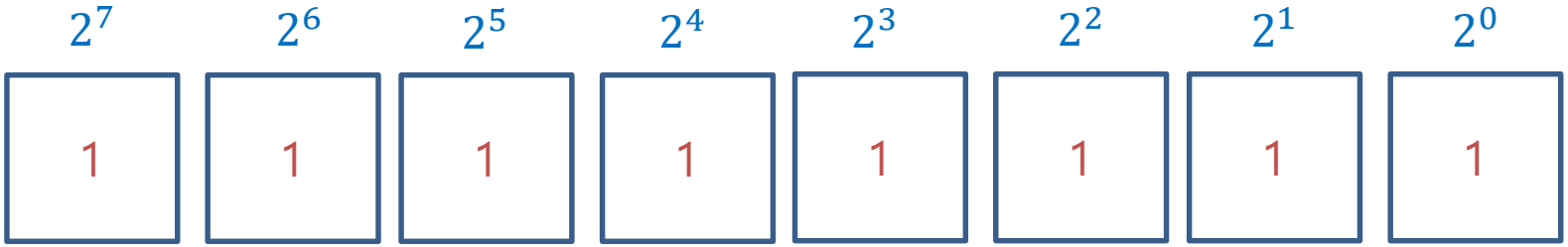
8bits = 2nibble = 1byte

1byte = 데이터가 저장되는 최소 단위

1-4



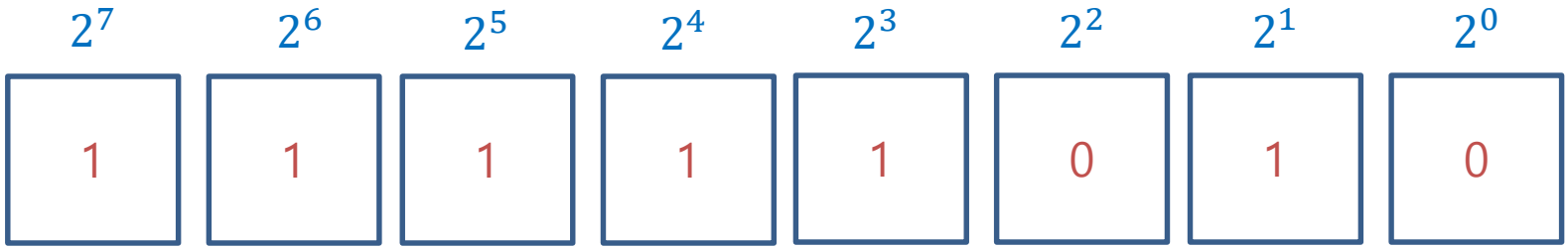
$(2^7 * 1) + (2^6 * 0) + (2^5 * 1) + (2^4 * 0) + (2^3 * 1) + (2^2 * 0) + (2^1 * 1) + (2^0 * 0) = 170$



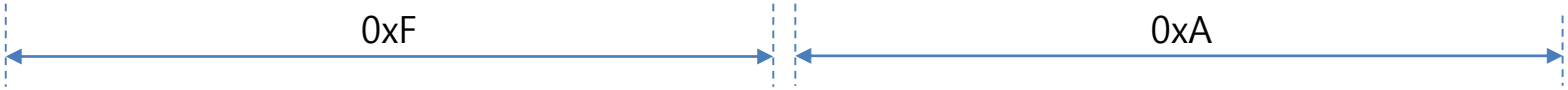
$(2^7 * 1) + (2^6 * 1) + (2^5 * 1) + (2^4 * 1) + (2^3 * 1) + (2^2 * 1) + (2^1 * 1) + (2^0 * 1) = 255$

1byte = 0 ~ 255까지 표현

1-5



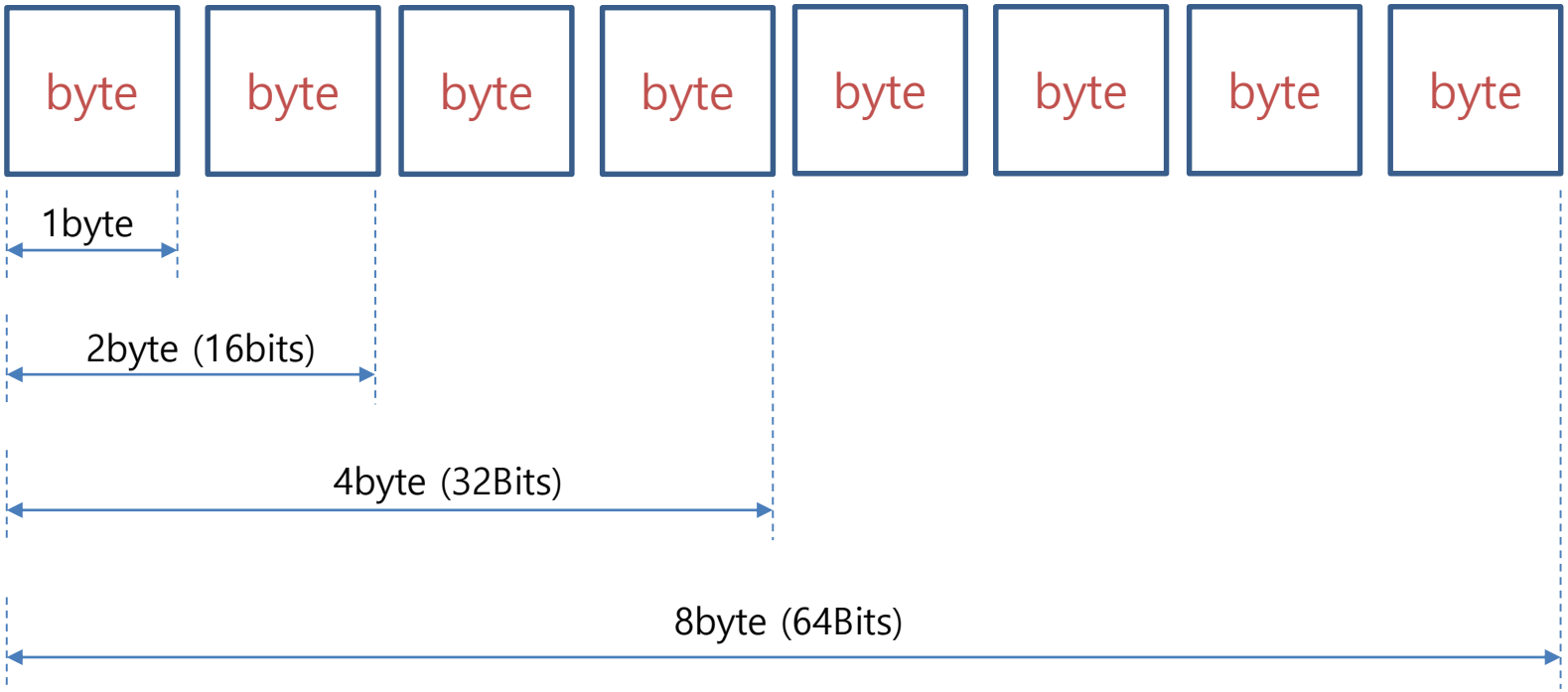
$$(2^7 * 1) + (2^6 * 1) + (2^5 * 1) + (2^4 * 1) + (2^3 * 1) + (2^2 * 0) + (2^1 * 1) + (2^0 * 0) = 250$$



0 1 2 3 4 5 6 7 8 9 A B C D E F

0xFA

1-6



컴퓨터는 숫자만 이해한다!

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
64	0x40	@	80	0x50	P	96	0x60	`	112	0x70	p
65	0x41	A	81	0x51	Q	97	0x61	a	113	0x71	q
66	0x42	B	82	0x52	R	98	0x62	b	114	0x72	r
67	0x43	C	83	0x53	S	99	0x63	c	115	0x73	s
68	0x44	D	84	0x54	T	100	0x64	d	116	0x74	t
69	0x45	E	85	0x55	U	101	0x65	e	117	0x75	u
70	0x46	F	86	0x56	V	102	0x66	f	118	0x76	v
71	0x47	G	87	0x57	W	103	0x67	g	119	0x77	w
72	0x48	H	88	0x58	X	104	0x68	h	120	0x78	x
73	0x49	I	89	0x59	Y	105	0x69	i	121	0x79	y
74	0x4A	J	90	0x5A	Z	106	0x6A	j	122	0x7A	z
75	0x4B	K	91	0x5B	[	107	0x6B	k	123	0x7B	{
76	0x4C	L	92	0x5C	₩	108	0x6C	l	124	0x7C	|
77	0x4D	M	93	0x5D	]	109	0x6D	m	125	0x7D	}
78	0x4E	N	94	0x5E	^	110	0x6E	n	126	0x7E	~
79	0x4F	O	95	0x5F	_	111	0x6F	o	127	0x7F	DEL

출처:
http://blog.naver.com/PostView.nhn?blogId=ouwukwfy&logNo=220248439711&parentCategoryNo=16&categoryNo=&viewDate=&isShowPopularPosts=true&from=search

byte	byte	byte	byte	byte	byte	byte	byte

휘발성

비휘발성

8bits = 1byte

1000bytes = 1Kbyte

1000Kbytes = 1Mbyte

1000Mbytes = 1Gbyte

1000Gbytes = 1Tbyte

1-10

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

문자 출력을 위해 ASCII 코드를 사용

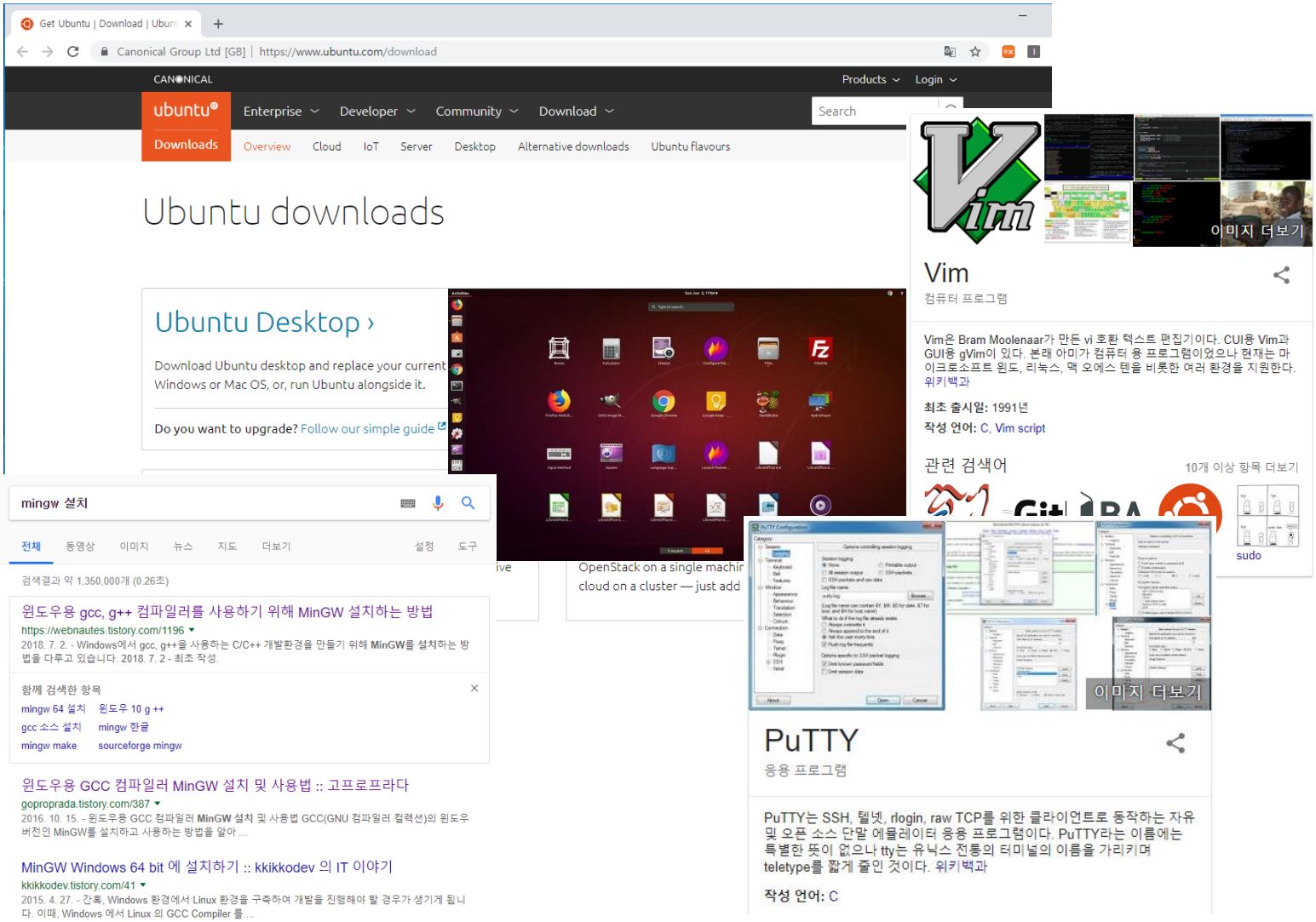
2장 C언어 데이터형

: 바이트에 데이터를 저장하는 방법

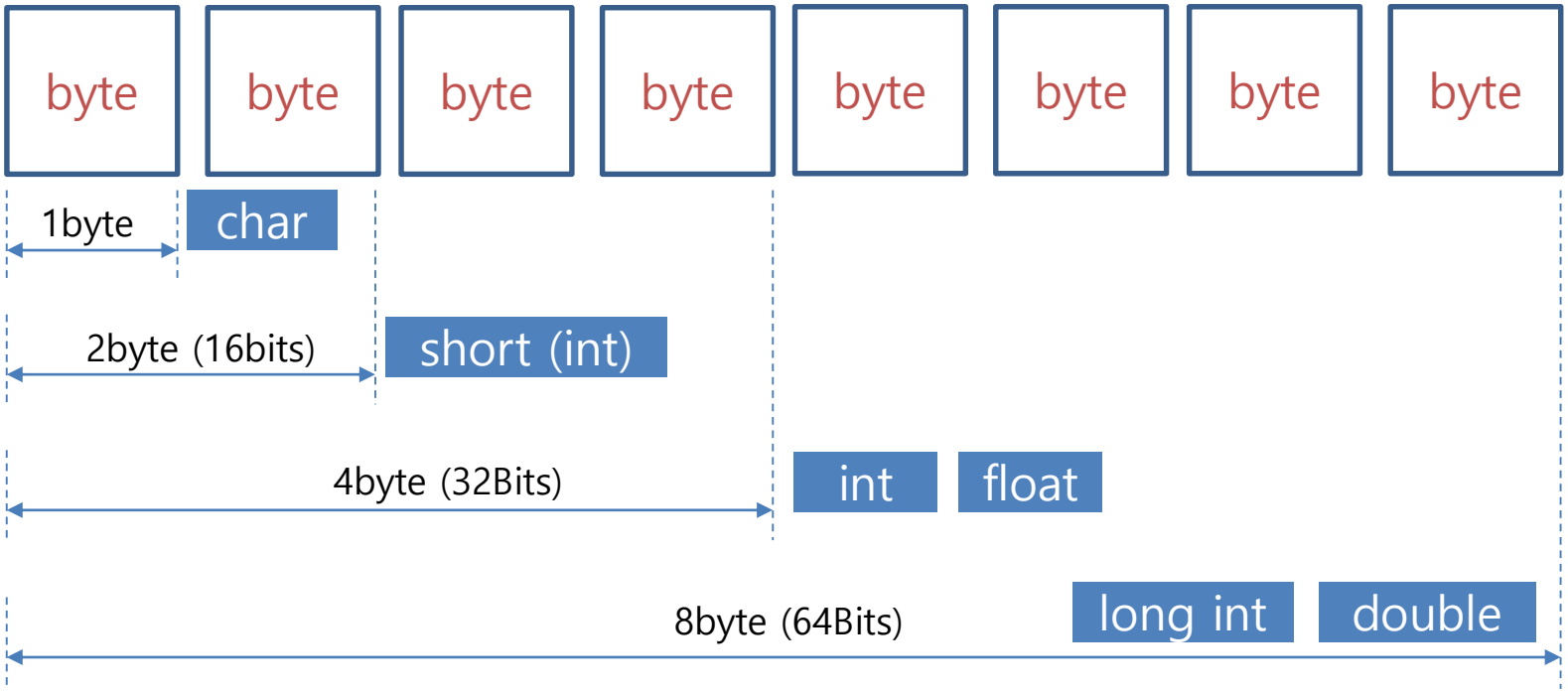
ps : PPT 무료 템플릿 (https://minheeblog.tistory.com/)

CONTENTS

2-1

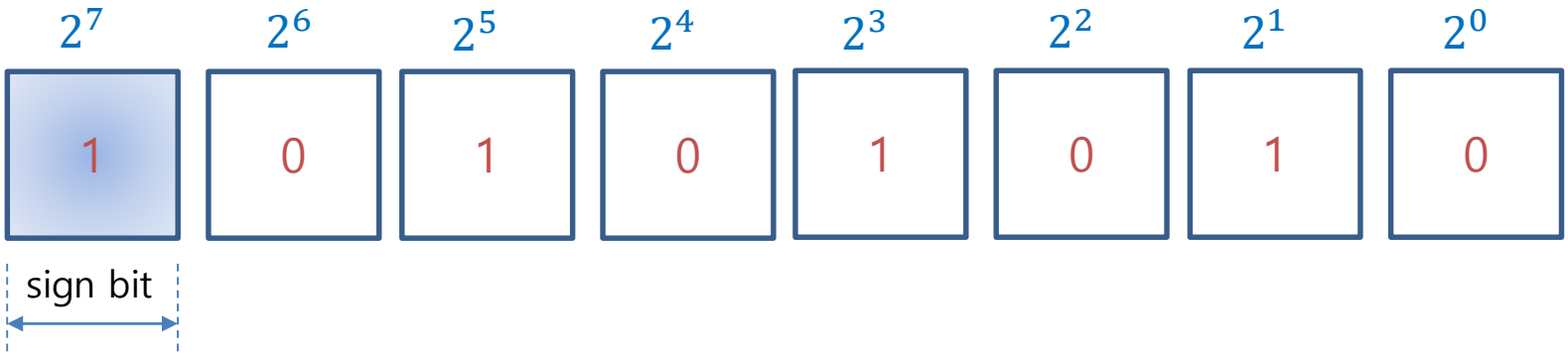

Get Ubuntu | Download | Ubunt
Canonical Group Ltd [GB] | https://www.ubuntu.com/download
CANONICAL
Products
Login
ubuntu
Enterprise
Developer
Community
Download
Search
Downloads
Overview
Cloud
IoT
Server
Desktop
Alternative downloads
Ubuntu flavours
Ubuntu downloads
Ubuntu Desktop ›
Download Ubuntu desktop and replace your current Windows or Mac OS, or, run Ubuntu alongside it.
Do you want to upgrade? Follow our simple guide
OpenStack on a single machir
cloud on a cluster — just add
이미지 더보기
Vim
컴퓨터 프로그램
Vim은 Bram Moolenaar가 만든 vi 호환 텍스트 편집기이다. CUI용 Vim과 GUI용 gVim이 있다. 본래 아미가 컴퓨터 용 프로그램이었으나 현재는 마이크로소프트 윈도, 리눅스, 맥 오에스 텐을 비롯한 여러 환경을 지원한다.
위키백과
최초 출시일: 1991년
작성 언어: C, Vim script
관련 검색어
10개 이상 항목 더보기
sudo
Google
mingw 설치
전체
동영상
이미지
뉴스
지도
더보기
설정
도구
검색결과 약 1,350,000개 (0.26초)
윈도우용 gcc, g++ 컴파일러를 사용하기 위해 MinGW 설치하는 방법
https://webnautes.tistory.com/1196
2018. 7. 2. - Windows에서 gcc, g++을 사용하는 C/C++ 개발환경을 만들기 위해 MinGW를 설치하는 방법을 다루고 있습니다. 2018. 7. 2 - 최초 작성.
함께 검색한 항목
mingw 64 설치
윈도우 10 g ++
gcc 소스 설치
mingw 한글
mingw make
sourceforge mingw
윈도우용 GCC 컴파일러 MinGW 설치 및 사용법 :: 고프로프라다
goproprada.tistory.com/387
2016. 10. 15. - 윈도우용 GCC 컴파일러 MinGW 설치 및 사용법 GCC(GNU 컴파일러 컬렉션)의 윈도우 버전인 MinGW를 설치하고 사용하는 방법을 알아 ...
MinGW Windows 64 bit 에 설치하기 :: kkikkodev 의 IT 이야기
kkikkodev.tistory.com/41
2015. 4. 27. - 간혹, Windows 환경에서 Linux 환경을 구축하여 개발을 진행해야 할 경우가 생기게 됩니다. 이때, Windows 에서 Linux 의 GCC Compiler 를 ...
이미지 더보기
PuTTY
응용 프로그램
PuTTY는 SSH, 텔넷, rlogin, raw TCP를 위한 클라이언트로 동작하는 자유 및 오픈 소스 단말 에뮬레이터 응용 프로그램이다. PuTTY라는 이름에는 특별한 뜻이 없으나 tty는 유닉스 전통의 터미널의 이름을 가리키며 teletype를 짧게 줄인 것이다. 위키백과
작성 언어: C

2-2



* int/float의 사이즈는 컴파일러와 OS(32bits/64bits)에 따라 달라질 수 있다.

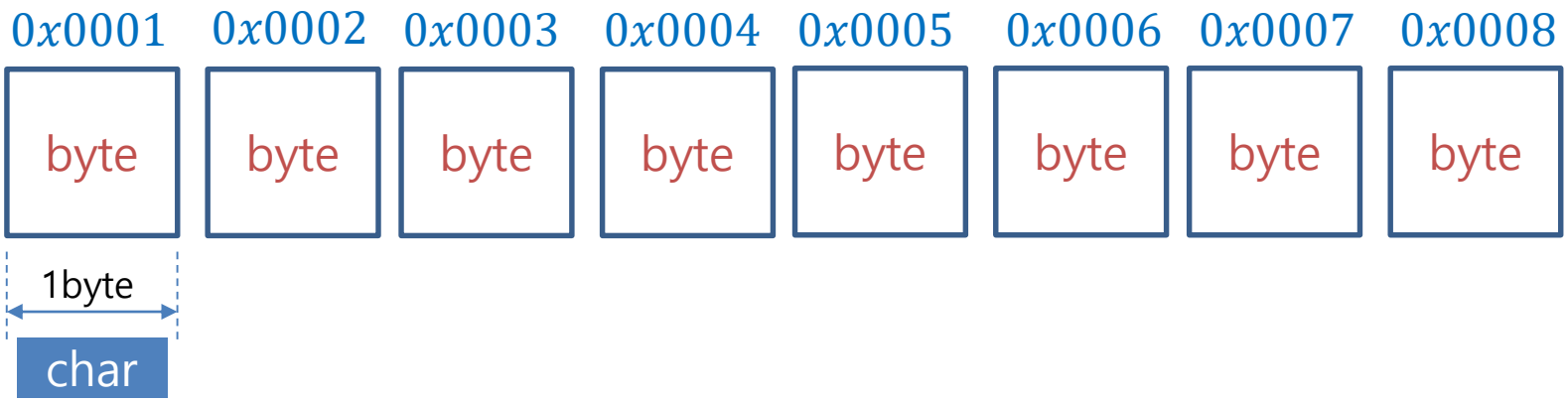
2-3



sign bit ON	sign bit OFF
char == signed char	unsigned char
short == signed short == short int == signed short int	unsigned short unsigned short int
int = signed signed	unsigned int

* float / double은 signed bit를 선택할 수 없다.

2-4

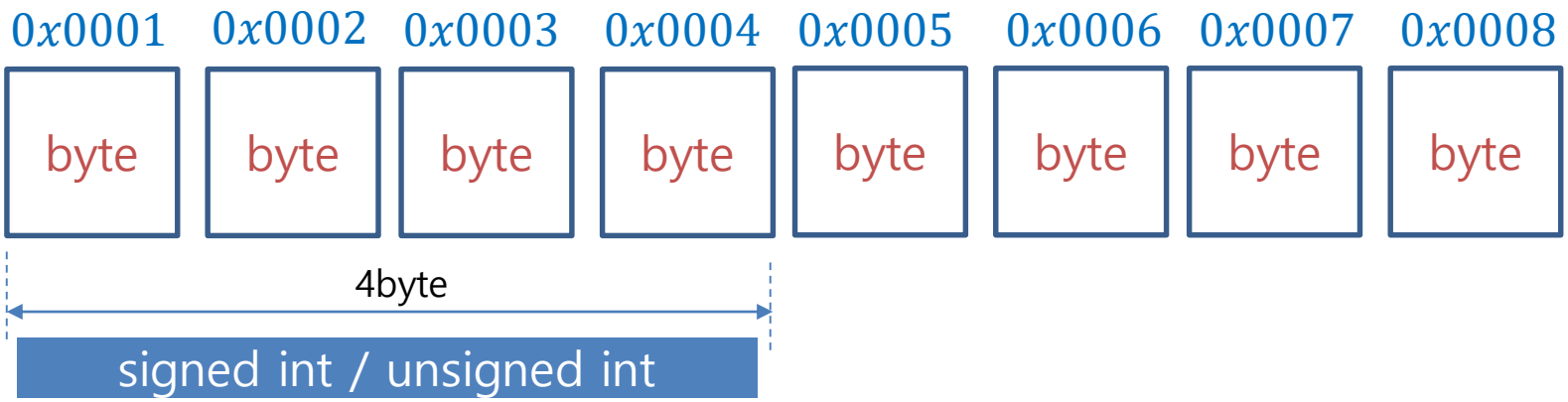


char 변수명 = 데이터(값);

ex) char myFirstInitial = 'J';

ex) char myCats = 3;

2-5

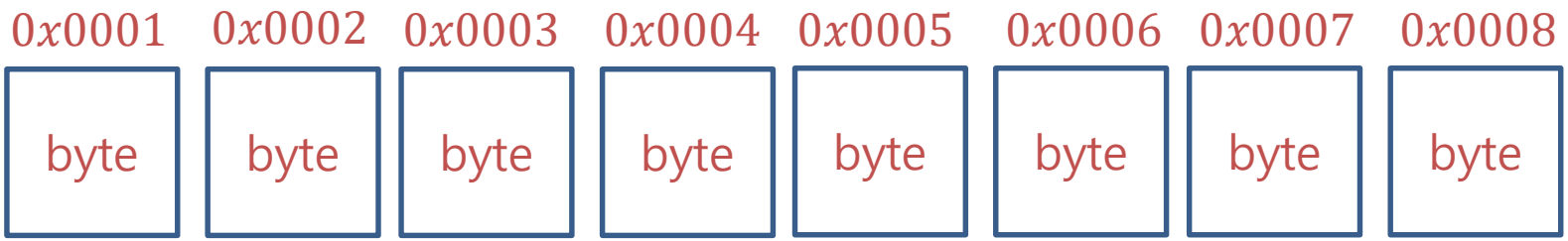


int 변수명 = 데이터(값);

int를 short 데이터형으로 바꾸면!?

ex) int chickenHouseNum = 50000;

ex) int classCount = 30;



const 데이터타입 변수명 = 데이터(값);

ex) const char myFirstInit = 'J';

ex) const int taxRate = 10;

ex) #define MALE 1

#define FEMALE 2

```
// C언어는 절차지향적 언어
// C언어는 함수형 언어

int   main()
{
        return 0;
}
```

2-8

```
#include <stdio.h>     // C 표준 라이브러리 헤더

int  main()
{
        printf("Hello world₩n");
        return 0;
}
```

2-9

```
#include <stdio.h>     // C 표준 라이브러리 헤더

int   main()
{
        char   myFirstInitial = 'J';
        printf("my first initial %c₩n", myFirstInitial);
        return 0;
}
```

2-10

```
#include <stdio.h>                  // C 표준 라이브러리 헤더
#define  OIL_PRICE   950            // 2019. 1월 기준 등유 가격
int  main()
{
        int liter = 200;            // 1 Drum
        int result = OIL_PRICE * liter;
        printf("It is %d won₩n", result);
        return 0;
}
```

Source (main.c)

=> Compile (gcc main.c)

=> Binary (a.out)

=> Process (RUN!)

CONTENTS

3-1

우선 순위	연산자	결합성
1	() [] -> . ++(후위) --(후위)	→ (왼쪽에서 오른쪽)
2	sizeof &(주소) ++(전위) --(전위) ~ ! *(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)

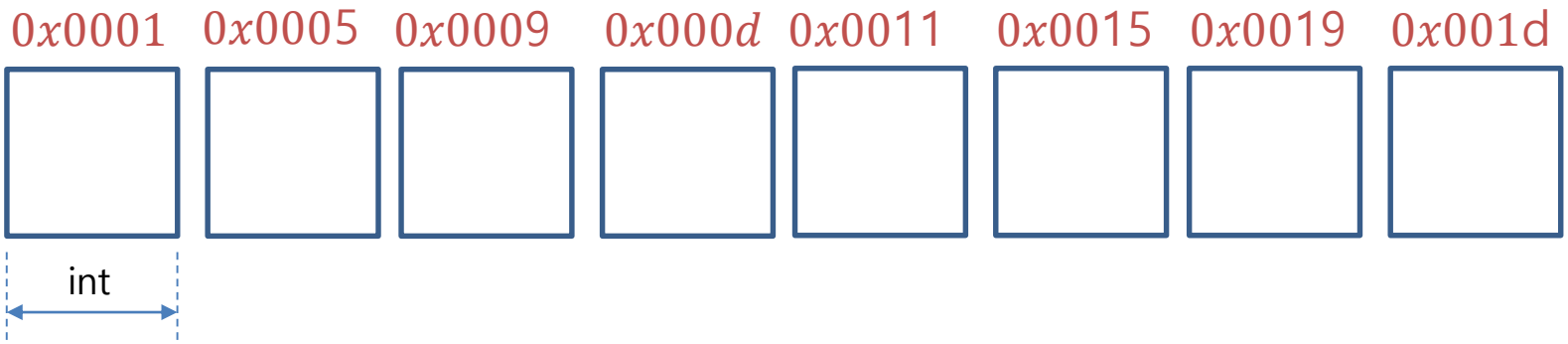
출처:
http://blog.naver.com/PostView.nhn?blogId=rnsu2011&logNo=220653971062&parentCategoryNo=&categoryNo=11&viewDate=&isShowPopularPosts=false&from=postView

3-2

result = a * b + 3 + 7 / 2

result = (a * (b + 3)) + (7 / 2)

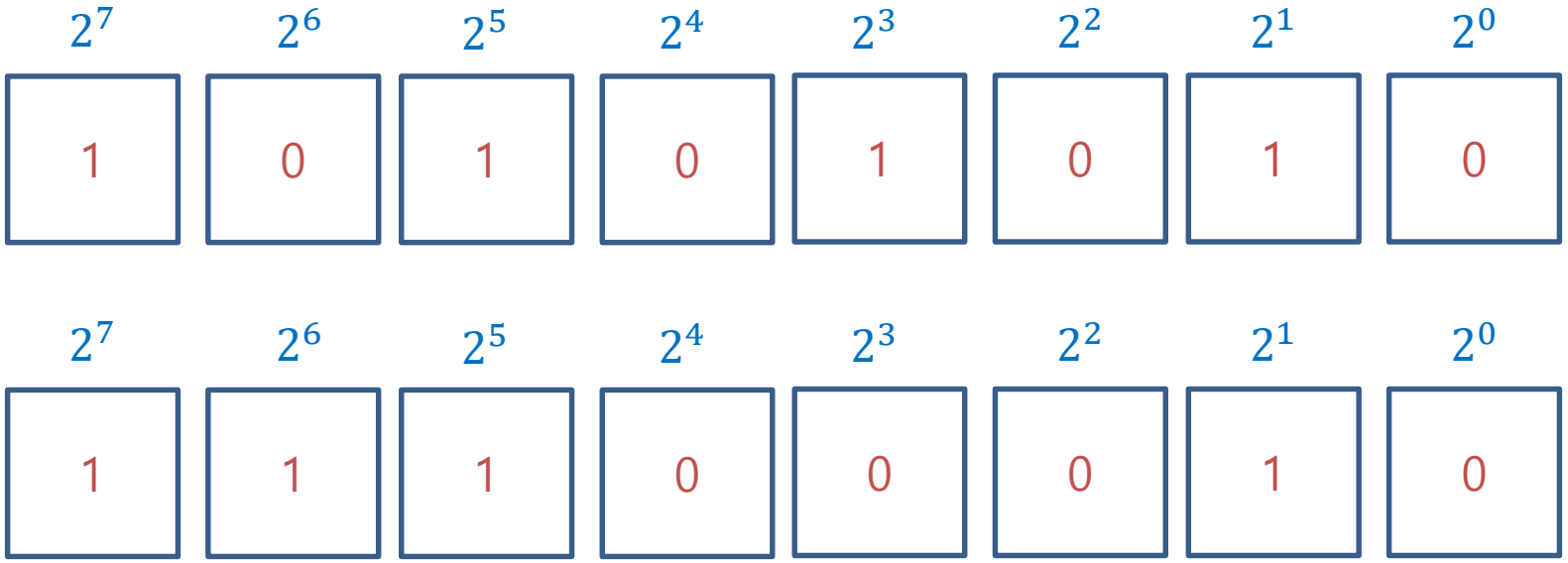
3-3




```
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;
```

3-4

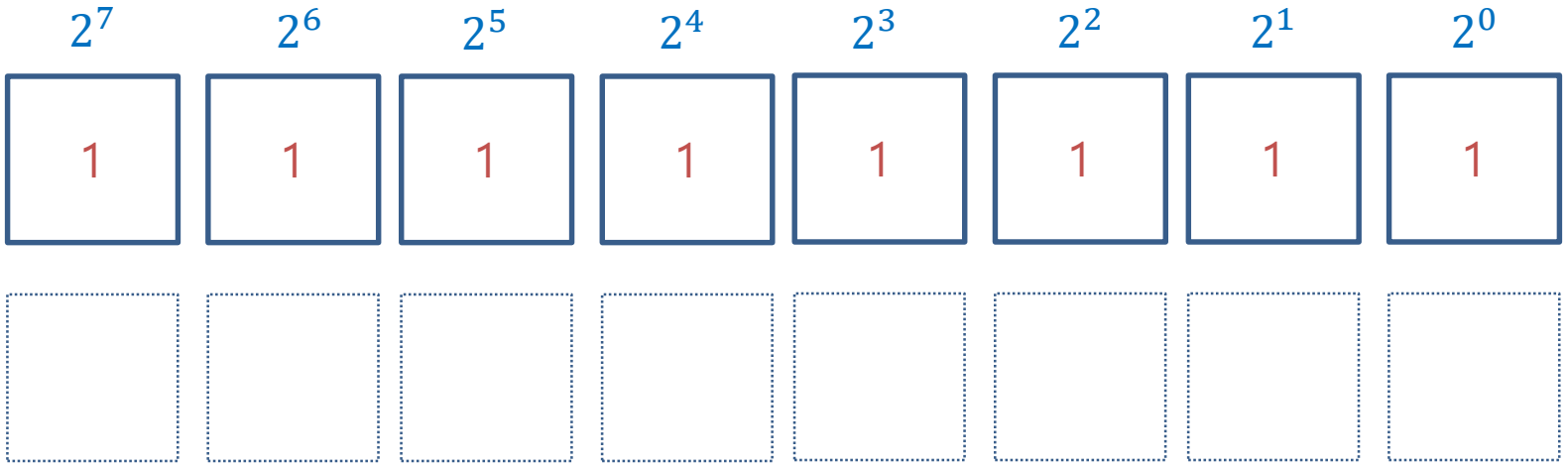


&연산 : 모두 참(1)인 경우

| 연산 : 하나라도 참(1)인 경우

~연산 : 참과 거짓을 반대로 변경 (Switch On/Off)

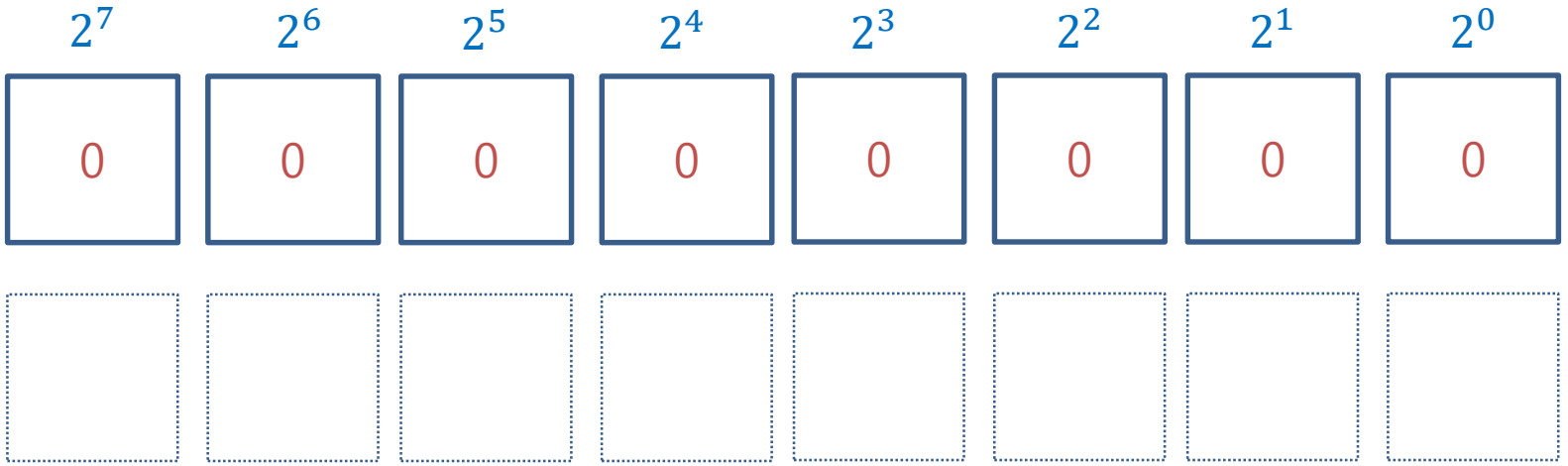
3-5




```
char data = 0b11111111;
data = data >> 1;
data = data << 1;
```

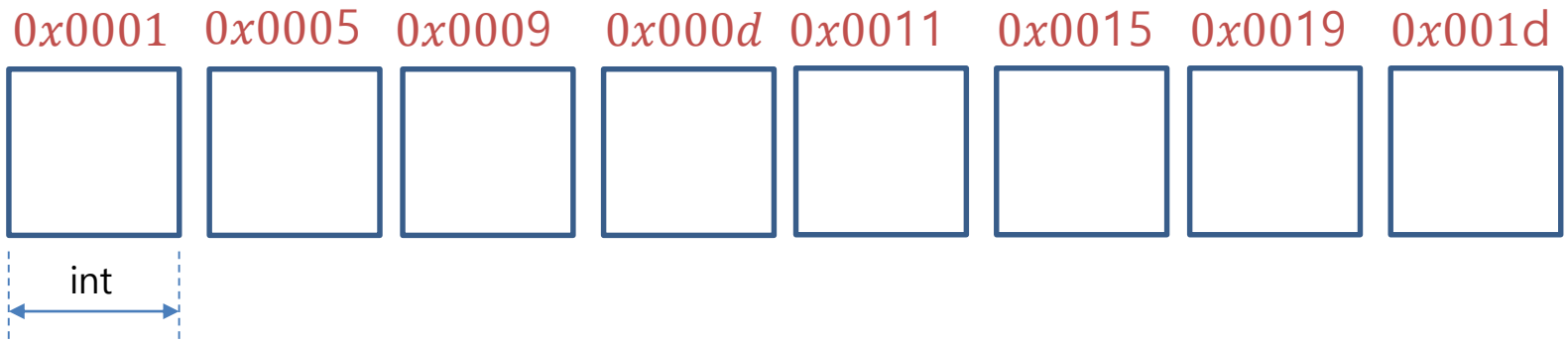
>> : 비트를 오른쪽으로 이동
<< : 비트를 왼쪽으로 이동

3-6




```
char data = 0;

data = data | (1 << 2);      // bit set
data = data & (1 << 2);      // bit get
data = data & ~(1 << 2);     // bit clear
```

3-7




```
int a = 3;


int result = !a;

int *result = &a;
```

3-8

```
int a = 3;


int result = ++a;   // 전위 증가
int result = a++;   // 후위 증가
int result = --a;   // 전위 감소
int result = a--;   // 후위 감사


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

3-9

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

a > b : a가 b보다 크다면 1(참), 아니면 0(거짓)

a < b : b가 a보다 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)

3-10

(논리연산) ? 참 : 거짓 ;

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

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

3-11

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)

CONTENTS

4-1

```
if (조건문 = true or false)
{ …
}
else if (조건 = true or false)
{ …
}
else
{ …
}
```

4-1

```
switch (정수값)
{
        case value1:
        {       ...
                break;
        }
        case value2:
        {       ...
                break;
        }
        default:
        {       ...
                break;
        }
}
```

CONTENTS

5-1

```
for (초기화; 조건문; 증감문)
{
        …
}


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

5-1

```
while (조건문)
{
        …
}

// while문으로 ASCII 코드 출력
char i=0;
while (i < 128)
{
        printf("%d (%#x) : %c", i, i, i);
        i++;
}
```

5-1

```
// while문으로 ASCII 코드 출력
char i=0;
while (1)
{
        printf("%d (%#x) : %c", i, i, i);
        i++;

        if ((i % 10) == 0) {
                printf("₩n");
                continue;
        }

        if (i == 127) break;
}
```

5-1

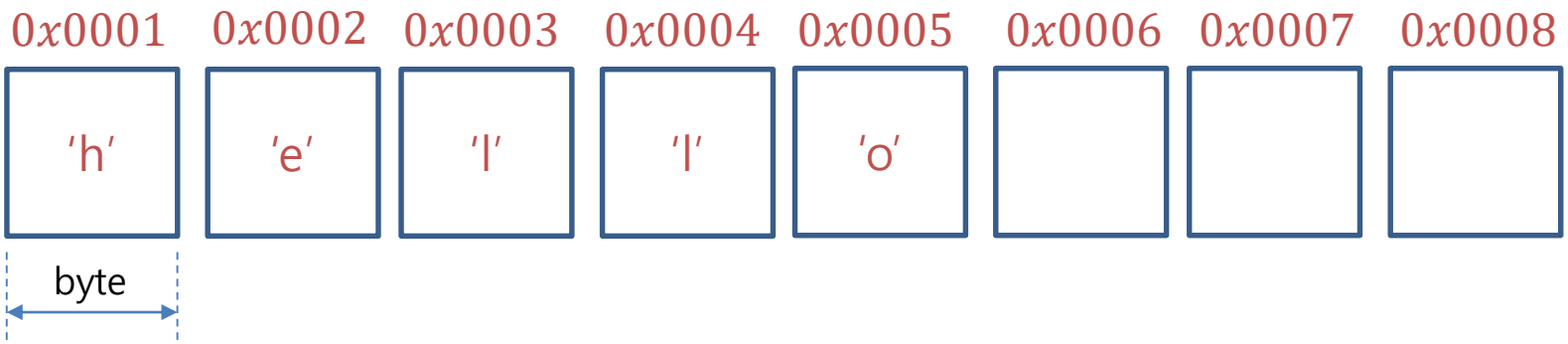
```
do
{
        …
} while (조건문);

// while문으로 ASCII 코드 출력
int i=1;
do
{
        printf(“%d”, i);
        i++;
} while (i < 128);
```

5-1

```
int i=10;

while (i) {
        printf("i = %d₩n", i);
        i--;

        if (i == 5) goto G_ERR;
}

G_ERR:
        printf("i value 5!₩n");
        return 0;
```

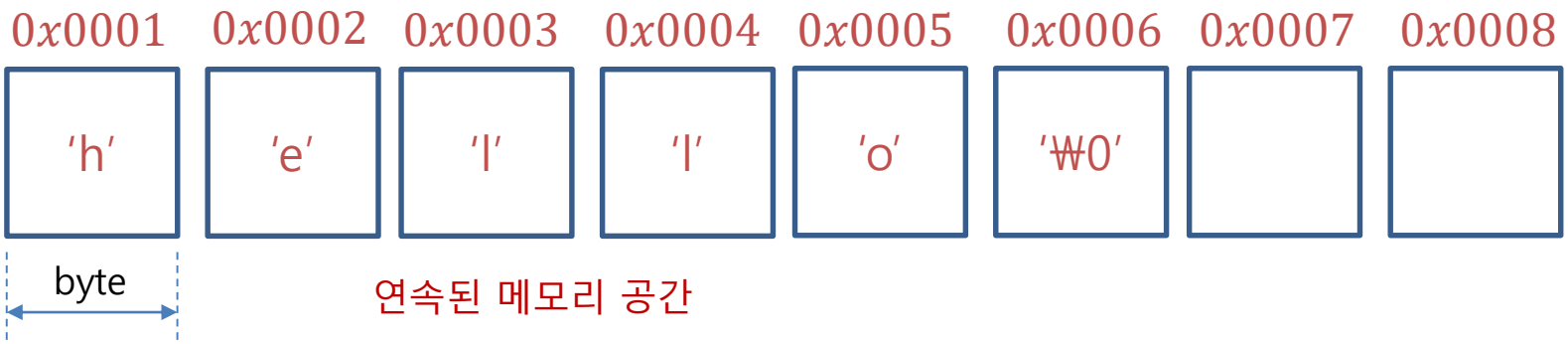
CONTENTS

6-1




```
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);
```

6-1



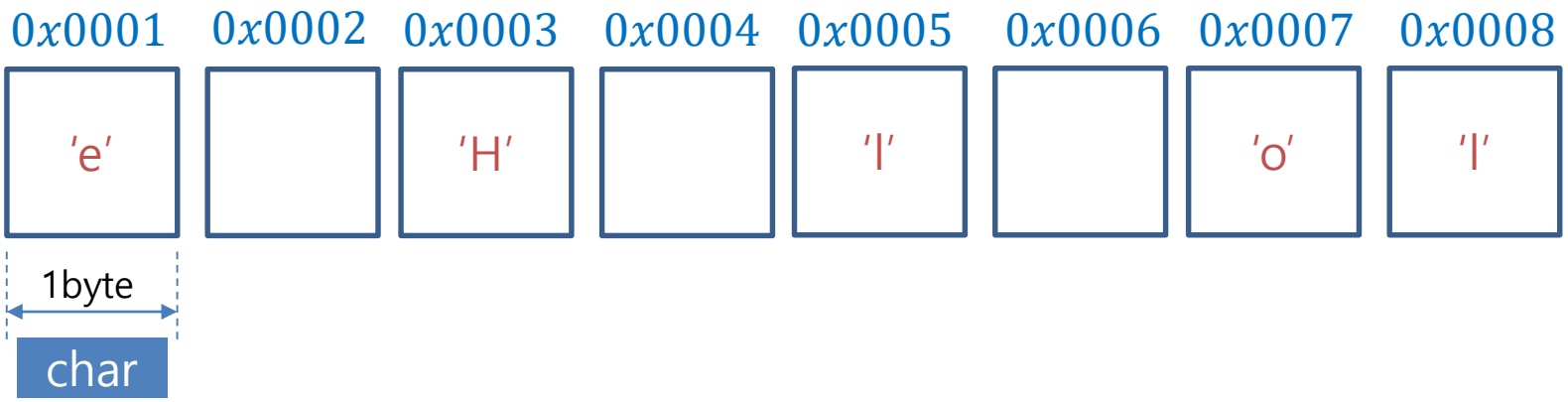

```
char *data1 = "hello";
char data2[10] = "world";

printf("%s %s₩n", data1, data2);
```

6-1

```
#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
```

CONTENTS

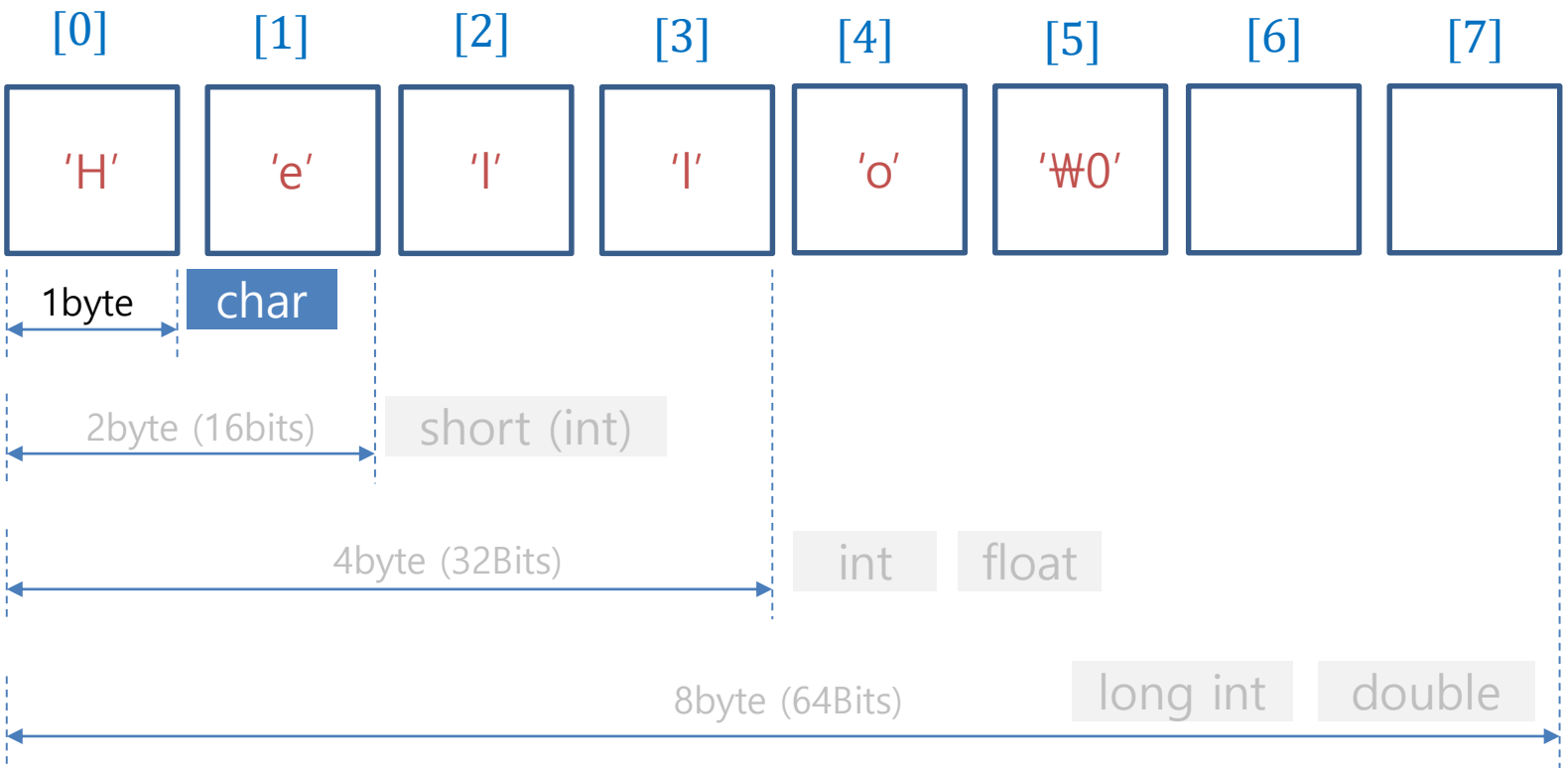
7-4



char 변수명 = 데이터(값);

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

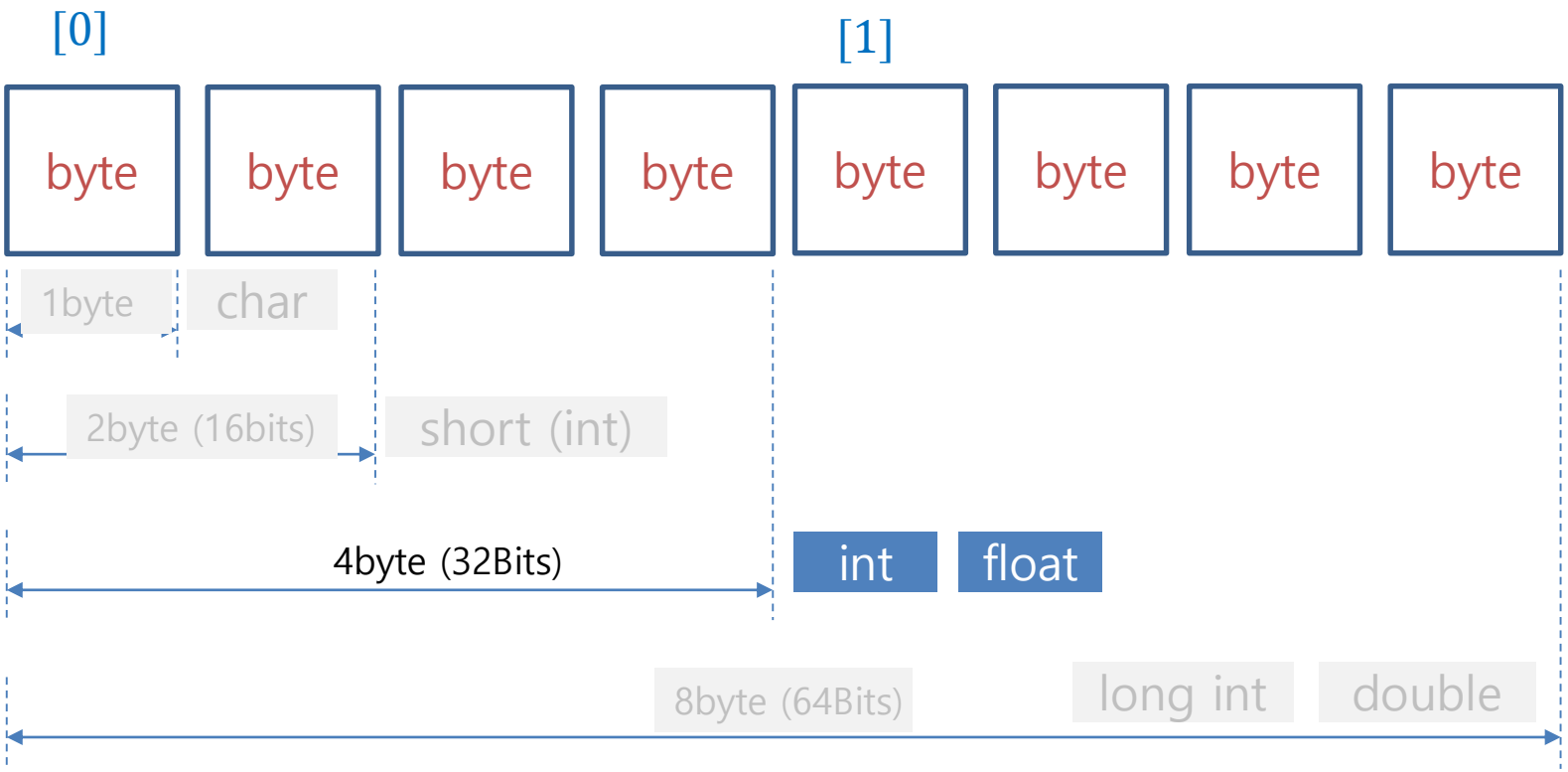
7-2



원하는 데이터형의 개수를 변수명의 대괄호([ ]) 사이에 표기하여 사용

```
char data[8];          // 선언시 문자열 또는 strcpy() 함수를 이용
data[0] = 'H';         data[1] = 'e';          data[2] = 'l'
data[3] = 'l';         data[4] = 'o';
```

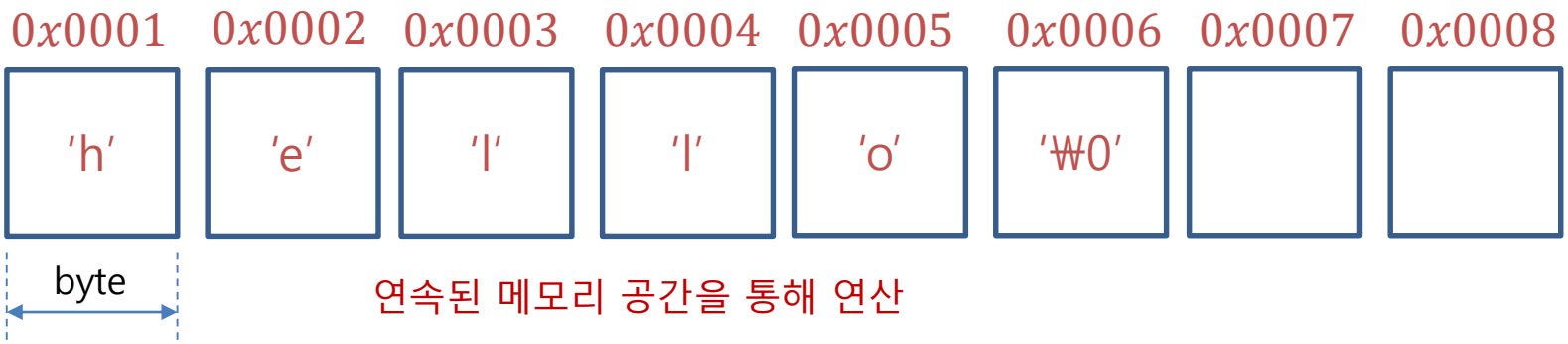
7-3



원하는 데이터형의 개수를 변수명의 대괄호([ ]) 사이에 표기하여 사용

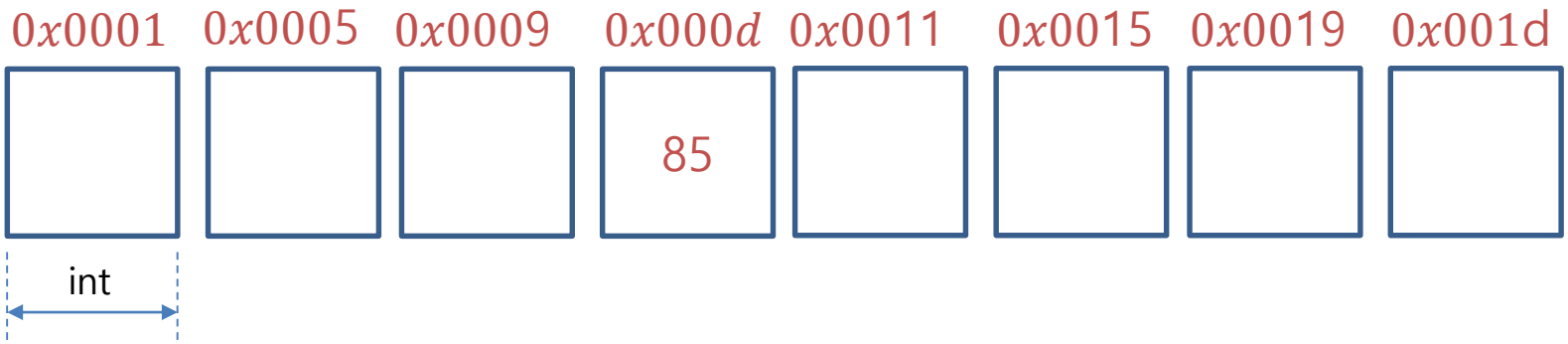
```
int data[2];
data[0] = 123;
data[2] = 10;
```

7-5



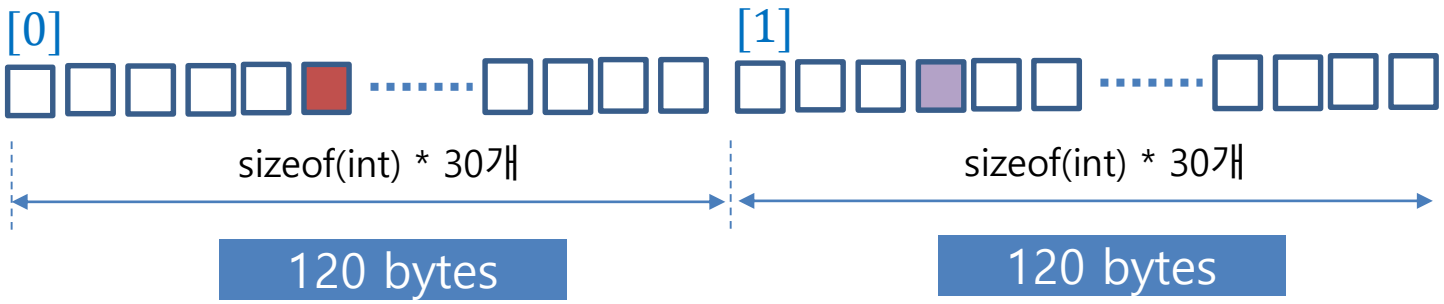
연속된 메모리 공간을 통해 연산

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



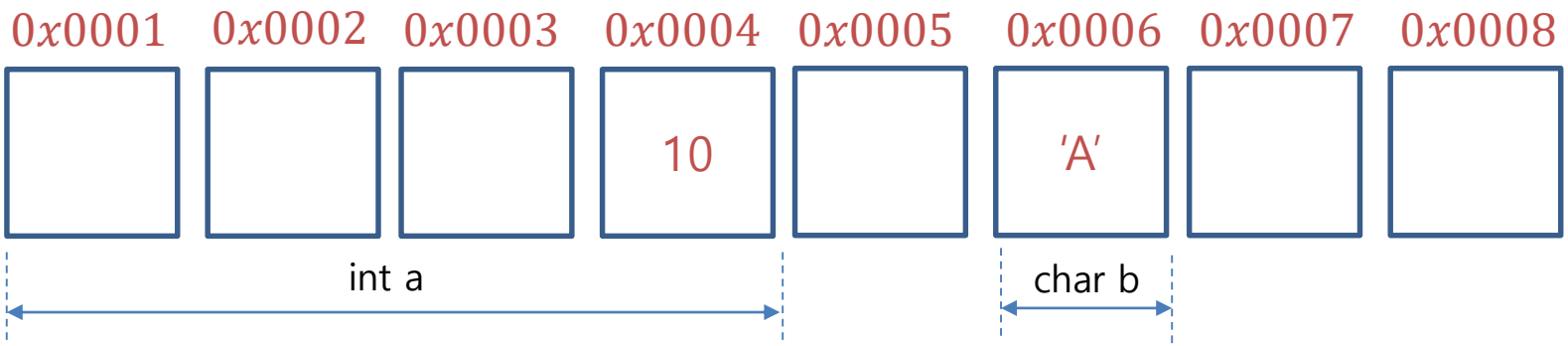

```
int student[30];
student[3] = 85;
```

7-7




```
int student[10][30];
student[0][5] = 90;
student[1][3] = 80;
```

7-8



1. 일반적인 연산자에서는 곱하기

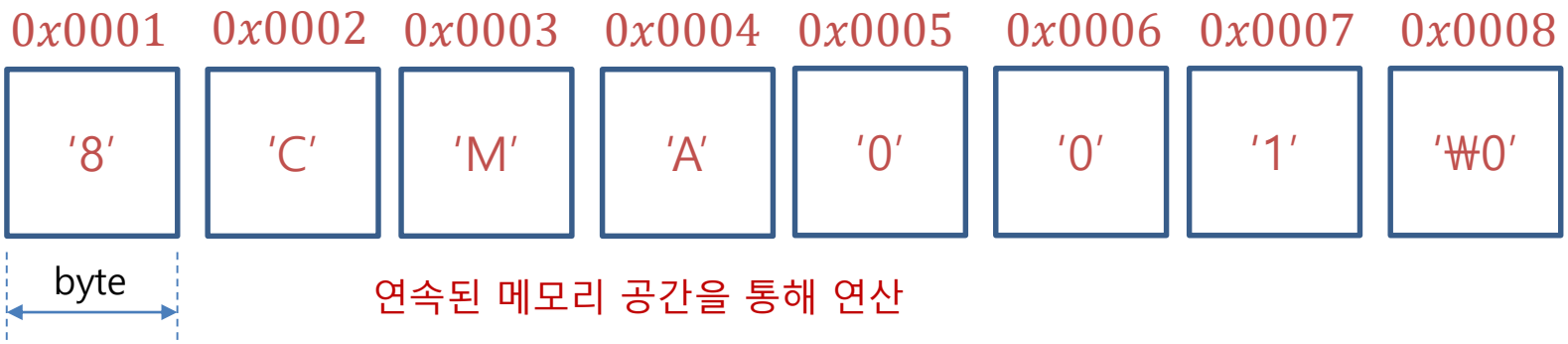
   ex) 10 * 2

2. 변수 선언시 주소를 저장하는 포인터 변수

   ex) int *ptr = &a; char *ptr = &b;

3. 표현식에서는 포인터 변수에 저장된 값

   ex) a + *ptr;



배열의 이름은 할당된 메모리의 시작 주소이다!

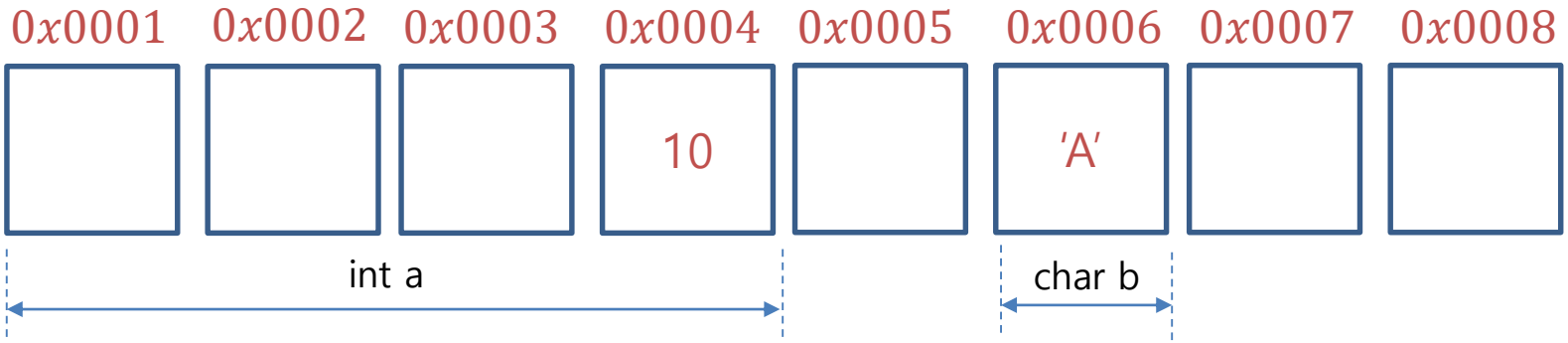
```
char serial[8];
serial == 0x0001;
serial[0] == *(serial + 0);     // *(serial + (sizeof(char) * 0));
serial[1] == *(serial + 1);     // *(serial + (sizeof(char) * 1));
```

CONTENTS

8-1

1. 연산을 편하고 빠르게 하기 위해
2. 참조에 의한 연산을 하기 위해
3. 동적 메모리를 할당하기 위해

8-1



1. 일반적인 연산자에서는 곱하기

   ex) 10 * 2

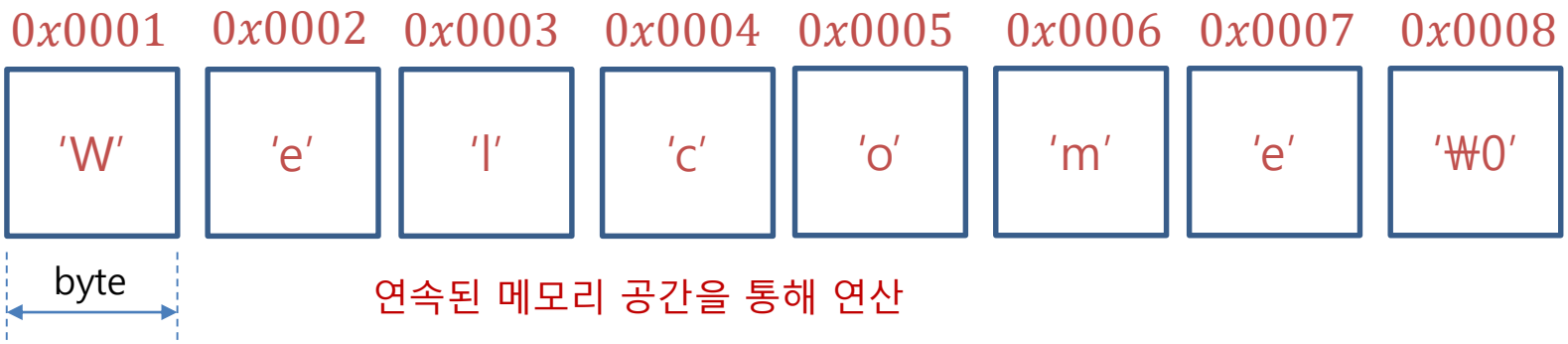
2. 변수 선언시 주소를 저장하는 포인터 변수

   ex) int *ptr = &a; char *ptr = &b;

3. 표현식에서는 포인터 변수에 저장된 값

   ex) a + *ptr;

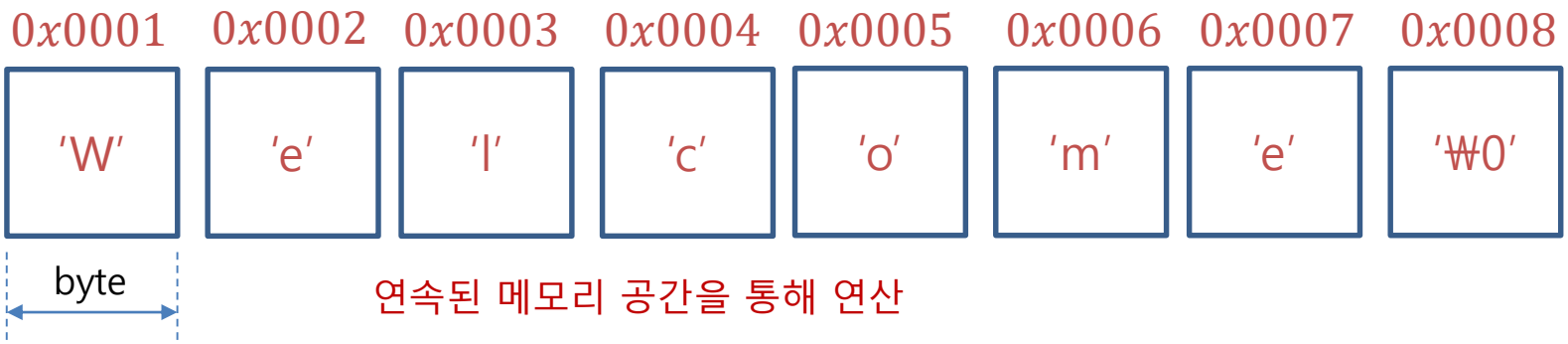
8-2



배열의 이름은 할당된 메모리의 시작 주소이다!

```
char serial[8];
serial == 0x0001;
serial[0] == *(serial + 0);     // *(serial + (sizeof(char) * 0));
serial[1] == *(serial + 1);     // *(serial + (sizeof(char) * 1));
```

8-3



포인터와 배열은 동작 방식이 똑같다!

```
char array[8];
char *ptr = array;
/* ----------------------------------------------------
ptr == array == &array[0]
*ptr == *(ptr+0) = *array == array[0] == ptr[0]
----------------------------------------------------*/
```

4-1

```
int a=3;        // 전역 변수


int main()
{
        int a=5, b=10;        // 지역 변수
        static int c = 1;     // 지역내 전역 변수
}
```

4-1

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

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

8-4

```
char *ptr = (char*)malloc(100);
memset(ptr, 0, 100);
free(ptr);

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

CONTENTS

7-1

고객 데이터 베이스 구축

{

고객 이름

고객 전화번호

고객 주소

고객 포인트

}

7-1

```
typedef struct _customer_data_t;
{
        char name[20];              // 고객 이름
        char phoneNumber[15];       // 고객 전화번호
        char *address;              // 고객 주소
        int customerPoint;          // 고객 포인트
} customer_data_t;
```

7-1

```
typedef struct _customer_data_t;
{
        char name[20];                  // 고객 이름
        char phoneNumber[15];           // 고객 전화번호
        char *address;                  // 고객 주소
        int customerPoint;              // 고객 포인트
} customer_data_t;

int main()
{
        customer_data_t  customerData;
        strcpy(customerData.name, "홍길동");
        customerData.customerPoint = 100;
        ...
}
```

CONTENTS

10-1

```
// C언어는 절차지향적 언어
// C언어는 함수형 언어 (모듈화)


int   main()
{
        return 0;
}
```

10-1

```
반환데이터형     함수이름(매개변수, ...)
{
        return 반환데이터;
}

int   proc_sum(int a, int b)
{
        return a+b;
}
```

10-1

1. 함수도 메모리 공간의 주소를 할당한다.
2. 함수의 메모리 주소를 알면 변수처럼 활용이 가능하다.
3. 함수의 인자로 함수 포인터 전달이 가능하다.

데이터타입 (*함수이름)();

```
int (*proc_func)();
```

CONTENTS

THANK YOU

유튜브 채널 : 시골사는 개발자