시스템 프로그래밍을 위한 C언어

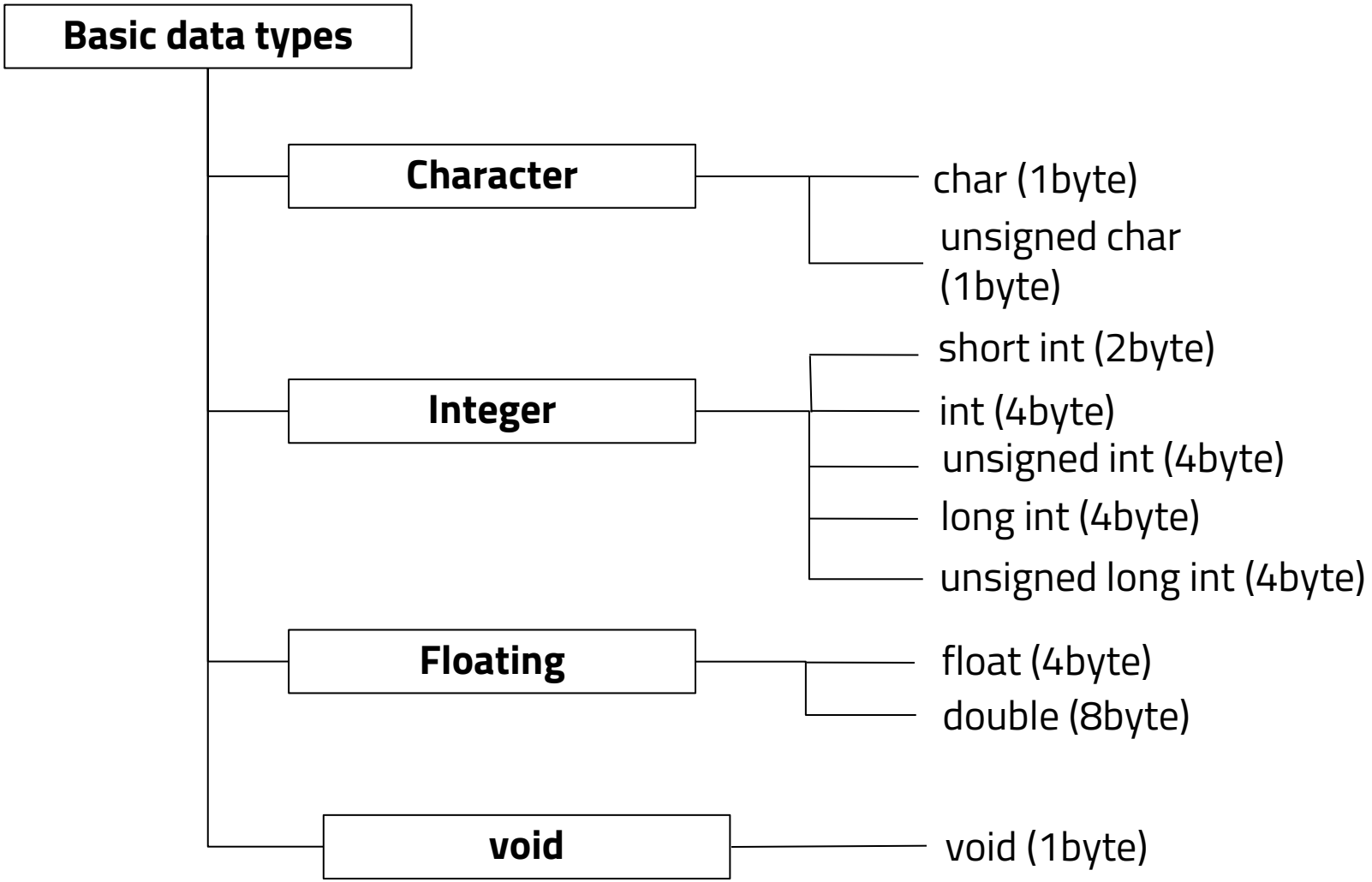
Variable & Address & Memory Allocation

현대자동차 입문교육
박대진 교수

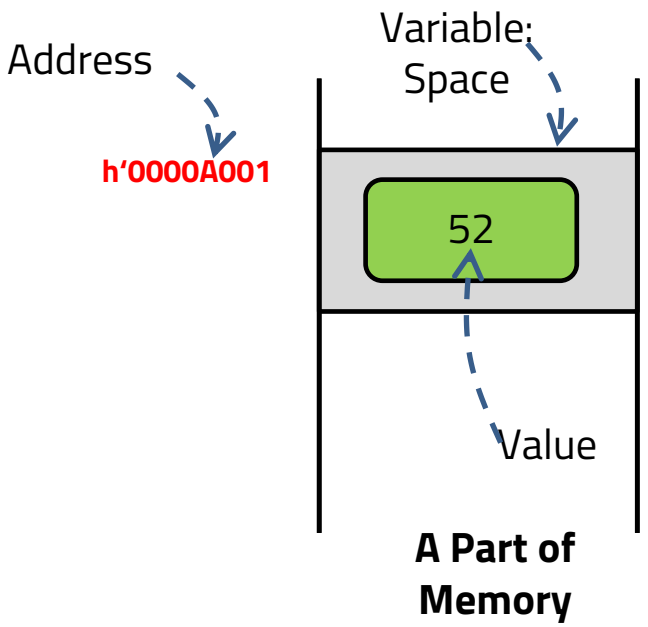
Lecture Lessoned

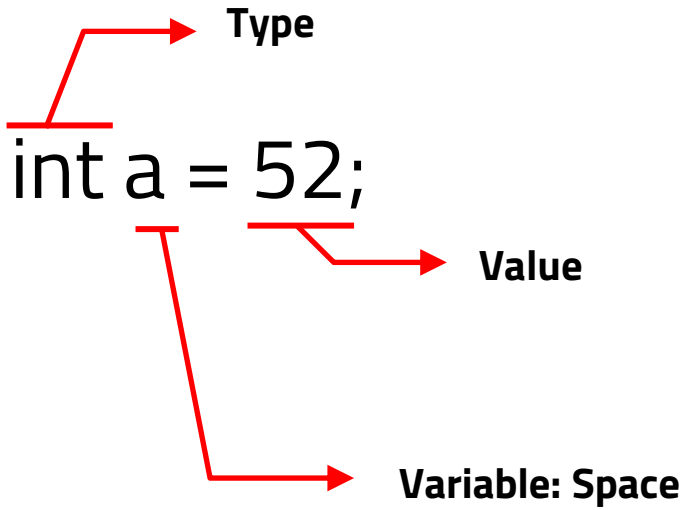
- C언어 복습
- 변수와 값
- 변수의 타입과 주소
- 포인터 변수
- 지역변수, 전역변수
- static, extern
- 구조체
- 복수의 파일로 구현

Basic Data Types (typical size)



Type vs. Address vs. Variable vs. Value

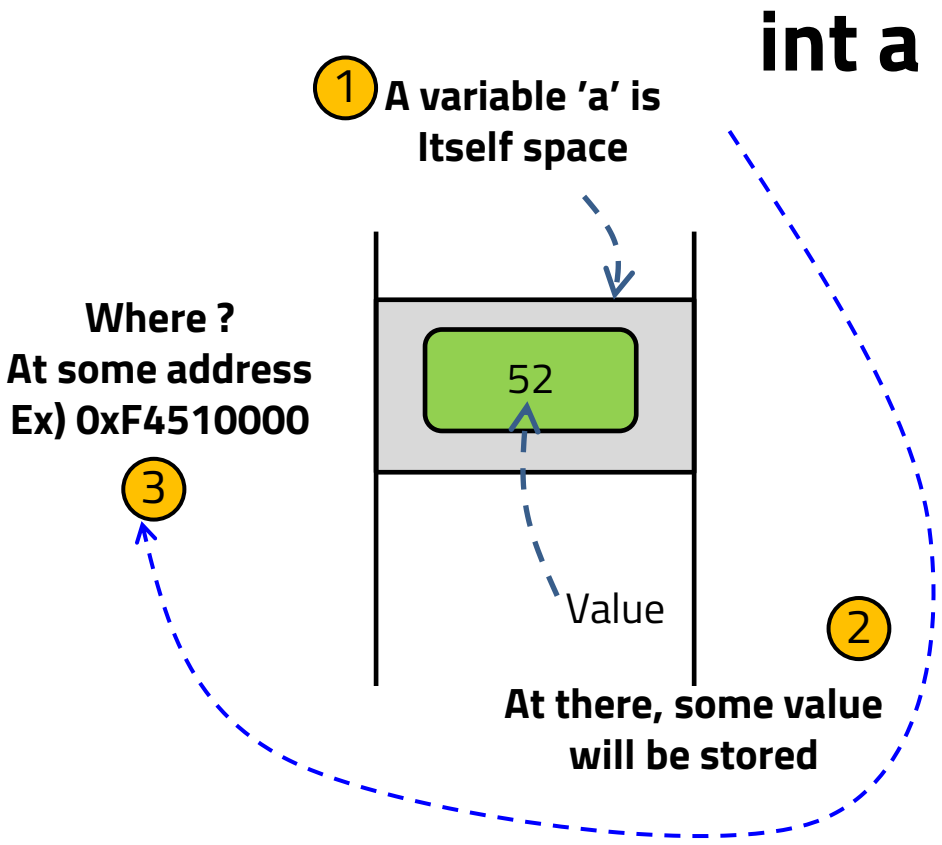




Address of variable: &a
Space of variable: a
Value of variable: 'd52

Pointer: Address of Variable

- All Variable will be allocated in address space somewhere



Do you remember the address 0xF4510000 ?

Just &a is same to 0xF4510000

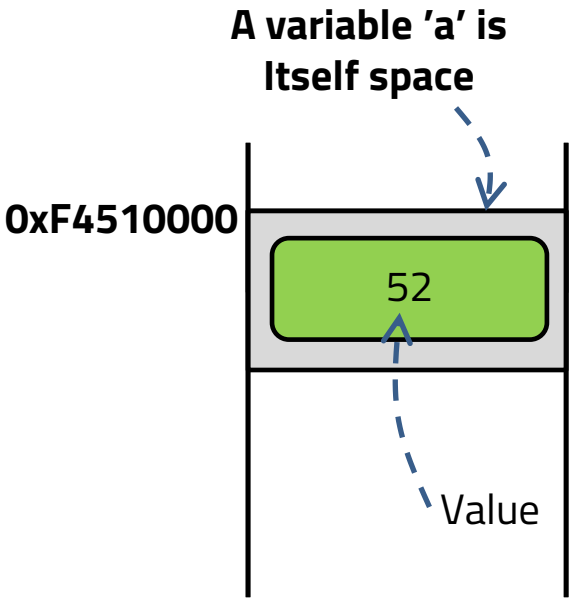
⇩

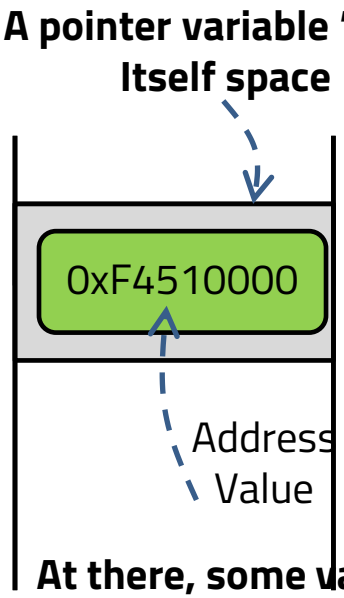
Pointer Value for Variable 'a'

Pointer Type's Variable

- A variable holding an address value of variable

```
int a = 52;
int* ap;
ap = &a;
```





At there, some value will be stored

But value is specially address

Accessing Original Variable by using Pointer Variable

- Dereferencing

```
int a = 52;
int* ap;
ap = &a;
```

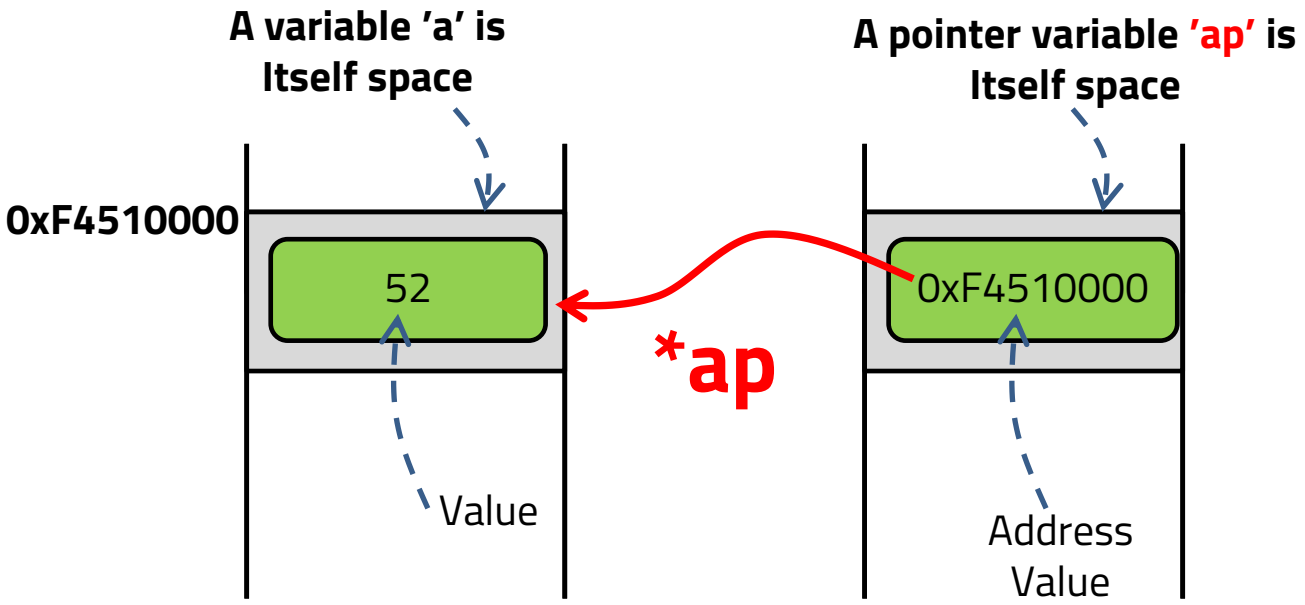
***ap = 100;**

same

***(&a) = 100;**

same

a =100



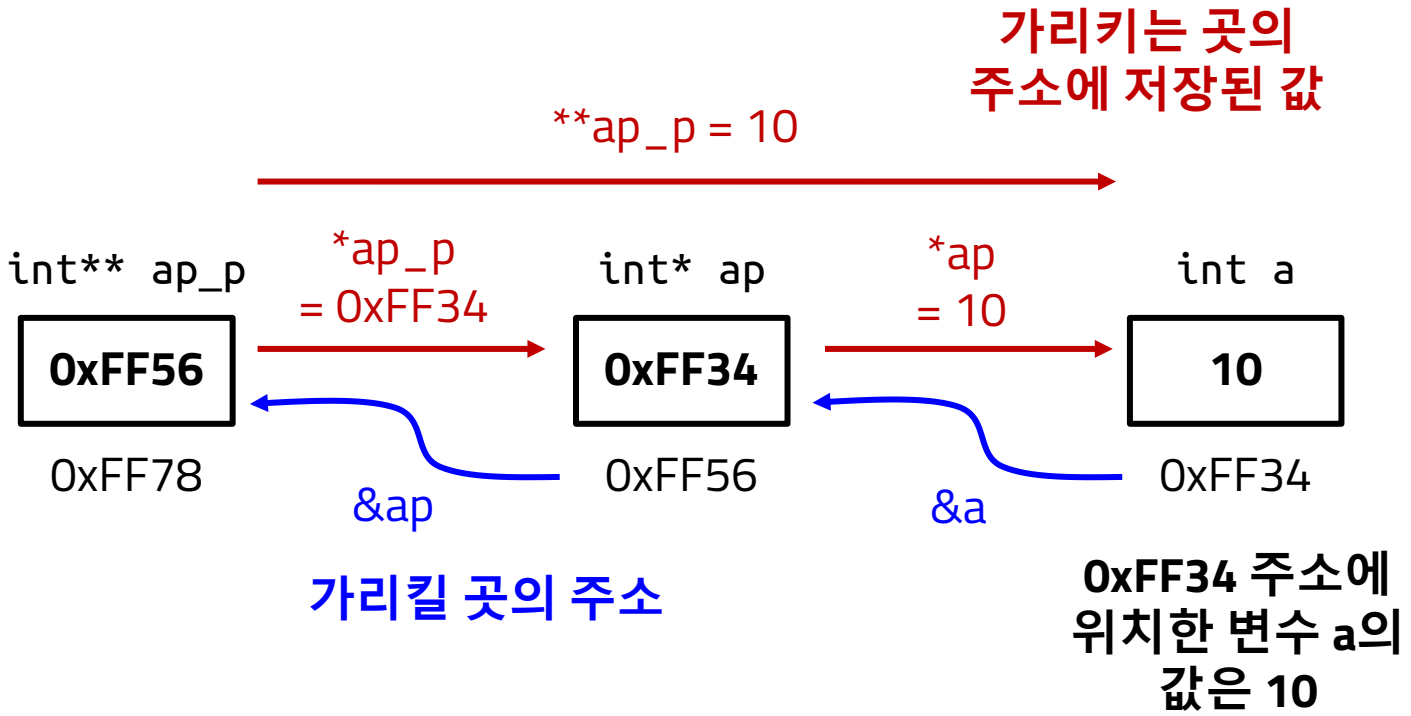
이중 포인터

주소를 통해 간접적으로 변수에 접근

```
int a = 10;

int* ap = &a;

int** ap_p = &ap;
```



Constant

- Integer constants value

```
int decimal = 123;                          ← decimal
int octal = 0123;                           ← octal
int hex = 0x123;                            ← hexadecimal
unsigned int unsigned_decimal = 123U;       ← unsigned          2 bytes
long int long_decimal = 123L;               ← long              4 bytes
```

- Floating constants value

```
float pi1 = 3.14F;          4 bytes
double pi2 = 3.14;          8 bytes
long double pi3 = 3.14L;    16 bytes
```

Size of vars is depend on processors

- const keyword in variable declaration

```
const float pi = 3.14;          float pi = 3.14;
pi = 3.1415;                    pi = 3.1415;
```

Error – constant variable can not be changed by assignment

Constant

char ch = ‘a’;

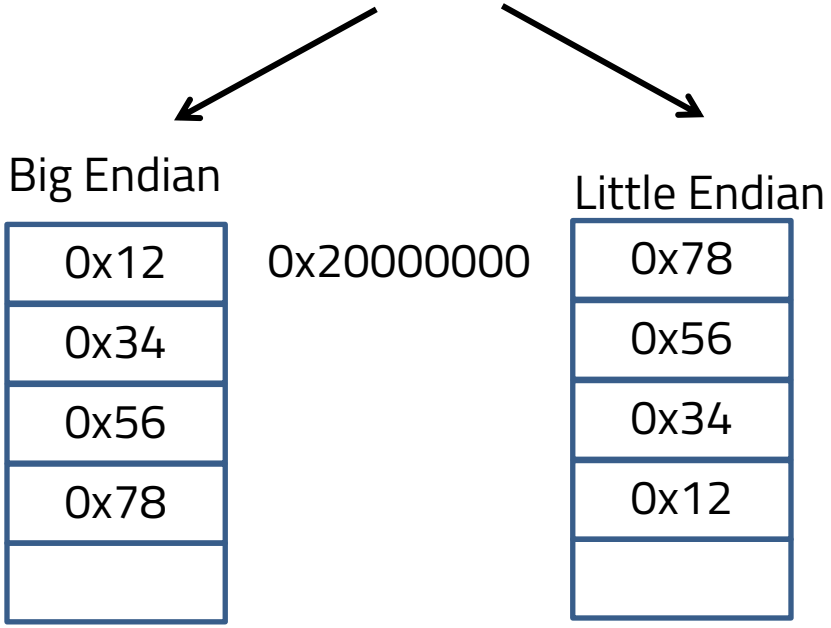
- Character constants
  - A character constant is 1 character enclosed in single quotes ‘ ’ (exception : escape sequence)
  - The value of character constant is the numeric value of the character in the machine’s character set.
    - Ex. ASCII character set
      the vaule of ‘0’ = 48
  - Escape Sequence

\a	alert (bell) character	\\	backslash
\b	backspace	\?	question mark
\f	formfeed	\’	single quote
\n	newline	\”	double quote
\r	carriage return		
\t	horizontal tab		
\v	vertical tab		

Endian을 고려해야할 때 vs. 고려할필요없을때

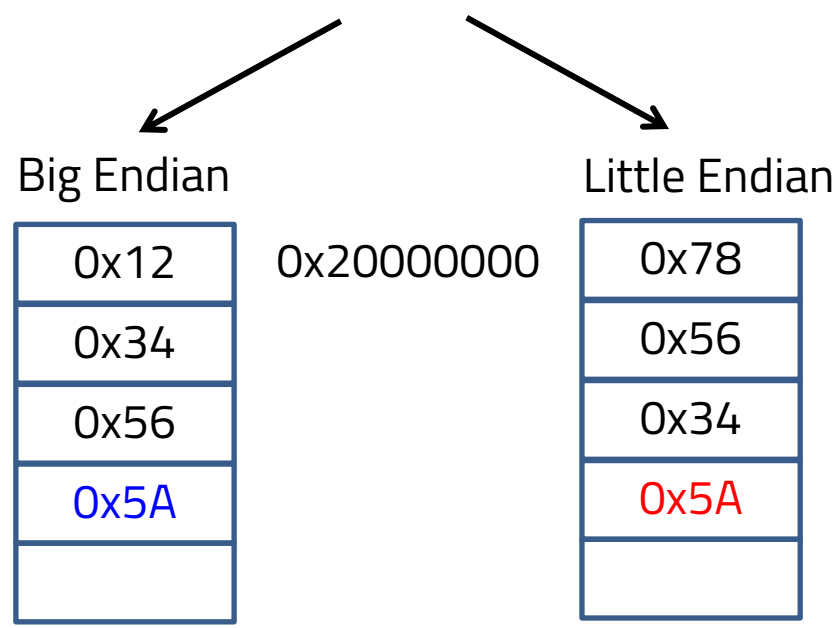
정의한 타입으로 사용시 메모리 레이아웃(endian)은 고려할 필요없다. 그러나 byte단위로 접근시 endian고려해야 함.

unsigned int x = 0x12345678;



- 컴파일러는 타겟 프로세서의 endian 타입을 알고 있고
- 자동으로 배치해준다
- 값을 읽을 때 원래 타입으로 사용시 endian고려해서 읽히고
- 사용자는 그냥 눈에 보이는 그대로 값이 있다고 보면 된다.

unsigned int x = 0x12345678;
unsigned char* p = &a
*(p+3) = 0x5A;



- MSB를 수정하기 위해 *(p+3)을 이용했으나, (little endian)을 가정해서
- 타겟 시스템이 Big endian으로 바뀌면, 코드가 무용지물
- 만약 하려면 *(p+0) = 0x5A로 바꾸어야 한다.