

# 재미있는 포인터 상수/변수

제주대학교 컴퓨터공학과

변영철 교수

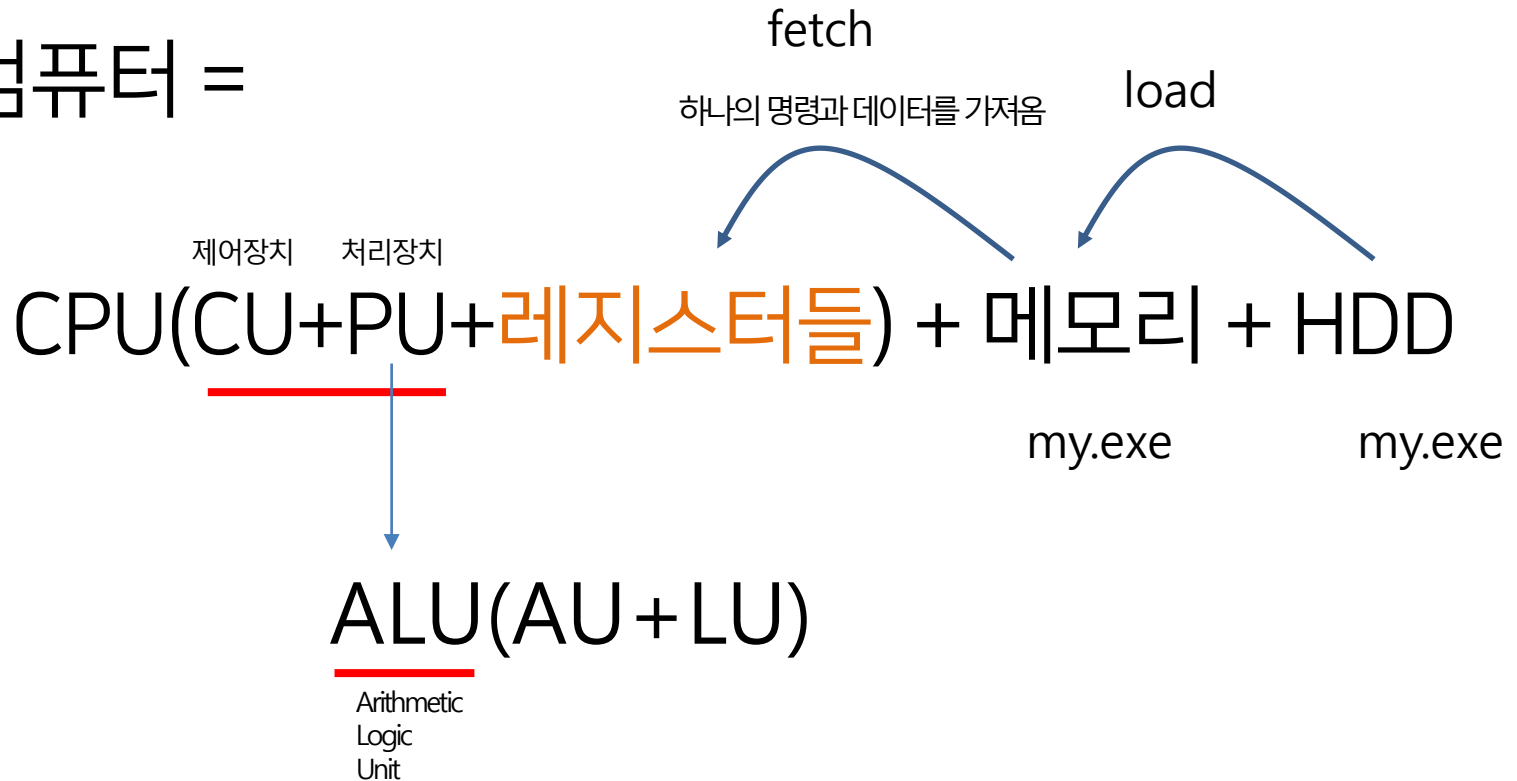
(ycb@jejunu.ac.kr)

# 이 장을 공부하면

- 주소 값, 포인터 상수를 이해할 수 있다.
- 포인터 변수를 이해할 수 있다.
- 포인터를 이용하여 변수, 배열, 함수를 접근할 수 있다.
- 레퍼런스를 이해할 수 있다.

# 주소 이야기

- 컴퓨터 =



# 주소 이야기

- 2비트로 표현할 수 있는 주소 크기는?
  - 2비트 = 00, 01, 10, 11 = 4개 =  $2^2(0 \sim 2^2-1)$
- CPU에 레지스터 중 주소를 저장하는 레지스터 크기는 32비트
- 32비트로 표현할 수 있는 주소 크기는?
  - 32비트 =  $2^{32}(0 \sim 2^{32}-1) = 4G$
  - 0 서른 두개(0) ~ 1 서른 두개( $2^{32}-1$ ) 주소
  - 16진수로 000000000 ~ FFFFFFFF

# 주소 이야기

- 정만선알, 정의는 만들고(어디에? 메모리에) 선언은 알린다(누구에게? 컴파일러에게)
- 따라서 변수 정의는? 변수를 메모리(어떤 주소)에 만드는 것
- 함수 정의는? 함수를 메모리(어떤 주소)에 만드는 것
- 메모리 어디에 만들어질까?

# 포인터 상수 이야기

- 변수가 어디에 만들어졌는지 알고 싶으면 **&변수명**을 출력해보라.
- &a는 변수 a가 들어있는 주소값(숫자, 상수)
- &a는 **주소값**, **포인터 상수**이다.

```
#include <stdio.h>

void main() {
    int a = 2;

    printf("%d", a);
    printf("%08X", &a);

    getchar();
}
```

숫자, 숫자, 숫자, 숫자, 숫자....

# 포인터 상수 이야기

- 배열이 어디에 만들어졌는지 알고 싶으면 **배열명**을 출력해 보라.
- 배열명은 배열이 만들어진 주소값(숫자, 상수)
- 배열명은 **주소값**, **포인터 상수**이다.

```
#include <stdio.h>

void main() {
    int array[3];
    array[0] = 2;

    printf("%d", array[0]);
    printf("%08X", array);

    getchar();
}
```



# 포인터 상수 이야기

- 함수가 어디에 만들어졌는지 알고 싶으면 **함수명**을 출력해보라.
- 함수명은 함수가 들어있는 주소값(숫자, 상수)
- 함수명은 **주소값**, **포인터 상수**이다.

```
#include <stdio.h>

void xxx() {
    printf("Hello,World!\n");
}

void main() {
    xxx();
    printf("%08X", xxx);

    getchar();
}
```

# 포인터 상수 이야기

- &변수명 (지역변수, 전역변수)
- 배열명 (지역배열, 전역배열)
- 함수명



위 3개는 모두 주소값(숫자, 포인터 상수)이다.

# 포인터 상수 이야기

- 지역변수(배열)가 만들어지는 곳 -> 스택 영역
- 전역변수(배열)가 만들어지는 곳 -> 전역 영역
- 함수가 만들어지는 곳 -> 코드 영역
- 지역변수(스택)와 전역변수 사이에 비어 있는 영역 -> 힙(heap) 영역 (new 동적할당 연산자로 힙 영역에 변수를 만들 수 있음)

# 포인터 상수 이야기

힙에 변수 만들고 주소  
출력해보기

00893BA0

00894658

```
#include <stdio.h>

void main() {
    int* pi = new int;
    char* pc = new char;

    printf("%0X\n", pi);
    printf("%0X\n", pc);

    getchar();
}
```

# 포인터 변수 이야기



제주시 용담1동 30번지

포인터 변수



제주시 용담1동 30번지

주소값

# 포인터 변수 이야기

주소를 적으면 넣으면 수첩이  
그 주소를 가리키는 것이 됨.

주소값을 저장하면 ‘가리키는  
(pointing)’ 것과 같아서  
포인터 변수라고 함.

- 머시기를 가리키는 포인터 p
- 머시기 \* p;
- 사람 \* p;
- 개 \* p;
- 동물 \* p;
- 가리키고 싶은 것 \* p;

# 포인터 변수 이야기

- char a를 가리키는 (포인터) 변수 pb
  - char a \* pb;
  - char \* pb;
- 
- int b를 가리키는 (포인터) 변수 pa
  - int b \* pa;
  - char \* pa;

# 포인터 변수 이야기

- 함수 void xxx(int a, int b)를 가리키는 포인터 변수 pd;
- void xxx(int a, int b) \* pd;
  - > void (int a, int b) \* pd;
  - > void (int, int) \* pd;
  - > void (\* pd) (int, int);

쉽게 이해하려면?  
“소대앞출, 소제거의  
원칙“



**함수 포인터를  
쉽게 이해하려면?**  
**“소대앞출, 소제거의 원칙”**

# 포인터 변수 이야기

- 문자 'A'를 저장하는 변수 a
  - 그 변수 주소를 저장하는(가리키는) 변수 pa
- 숫자 3을 저장하는 변수 b
  - 그 변수 주소를 저장하는 (가리키는) 변수 pb

**왜 가리킬까? 왜, 왜, 왜?  
(정답) 읽거나 쓰려고...**

# 포인터 변수 이야기

- 변수 포인터 변수
- \* 의미 : '가리키는'
  - int a \* pa;
  - int \* pa;
- \* 의 또 다른 의미 : '읽어라', '써라' 라는 의미

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

```
void main() {
```

```
    char a = 'A';
```

```
    char* pa;
```

```
    pa = &a;
```

```
    int b = 3;
```

```
    int* pb;
```

```
    pb = &a;
```

```
    printf("%c\n", *pa);
```

```
    printf("%d\n", *pb);
```

```
    getchar();
```

```
}
```

'가리키는'이라는  
의미

pa가 가리키는  
것을 '읽어라' (\*)라는  
의미

# 포인터 변수 이야기

- 써라(=의 왼쪽에 \*가 있을 때)
- 그 외 : 읽어라

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

```
void main() {
```

```
    char a = 'A';
```

```
    char* pa;
```

```
    pa = &a;
```

```
    char tmp;
```

```
    tmp = *pa; //읽어라
```

```
    *pa = 'B'; //써라
```

```
    printf("%c\n", tmp);
```

```
    printf("%c\n", *pa); //읽어라
```

```
    getchar();
```

```
}
```

# 포인터 변수 이야기

- 배열명은 주소값
- 배열에서는 포인터로 무엇을 할까?
- 배열에 있는 요소 하나를 액세스하기 위함.

왜 배열 전체를 가리키지 않고 배열  
요소 하나만 가리키도록 할까?  
[정답] 전부 가리켜서 뭐할건데???  
원소 하나 하나 읽는 것이 목적

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

```
void main() {  
    int array[3];  
    array[0] = 2;  
    int * p;  
    p = array;  
    printf("%d\n", array[0]);  
    printf("%d\n", *p);  
  
    getchar();  
}
```

p가 가리키는  
것을 읽어라(\*)  
라는 의미

# 포인터 변수 이야기

- 일차원 배열 `int array[3]`  
포인터 변수
- `int * p;`
- `p`가 가리키는 것을 읽어라.  
`*p` 혹은 `*(p + 0)`
- 다음 것을 읽어라.  
`*(p + 1)`

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

```
void main() {  
    int array[3];  
    array[0] = 2;  
    int * p;  
    p = array;  
    printf("%d\n", array[0]);  
    printf("%d\n", *p);  
  
    getchar();  
}
```

`p`가 가리키는 것을 읽어라(\*)라는 의미

# 포인터 변수 이야기

```
int * p;
```

\*p 의미 -> "p가 가리키는 것을 읽어라(\*)."

\*p -> \*(p+0) -> p[0]

\* (p+1) -> p[1]    "p가 가리키는 것에서 1번째를(+1) 읽어라(\*)."

\*(p+2) -> p[2]

포인터수식    배열수식

\* ( ? + 1 )

# 포인터 변수 이야기

- 2차원 배열을 읽는 방법

(1) 1줄 전체를 가리키도록 하자.

```
int[4]* p;
```

```
int (* p)[4]; //(O)
```

```
p = array;
```

소중입출소제거  
의 원칙

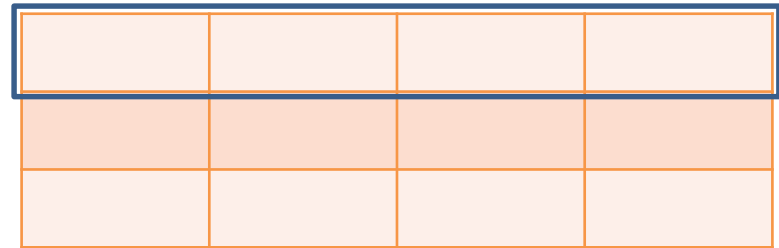
(2) 1줄 전체를 읽어라(\*).

```
*p
```

```
*(p+1)
```

```
*(p+2)
```

```
int array[3][4];
```



우선, 이거 전체를  
가리키도록 한다.

(3) 읽은 것에서 다시 1번째를 읽어라.

```
*( *p+1)
```

```
*( *(p+1)+1)
```

```
*( *(p+2)+1)
```

\* (? + 1)




# 포인터 변수 이야기

쉽게 이해하려면?  
“소대앞출, 소제거의  
원칙”



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

```
void main() {  
    int array[3][4];  
    array[0][0] = 2;  
    int(*p)[4];  
    p = array;  
    printf("%d\n", array[0][0]);  
    printf("%d\n", *( *p ));  
    printf("%d\n", *( * (p+0) ) + 0);  
  
    getchar();  
}
```



해석해보자.

# 포인터 변수 이야기

- 포인터를 사용하는 이유 3가지
  - 포인터를 이용하면 편리한 경우가 있어서 (함수 포인터 등)
  - 포인터를 이용해야 가능한 경우가 있어서 (주소에 의한 호출 및 값 교환)
  - 포인터를 이용하면 더 효율적인 경우가 있어서 (포인터 수식의 코드는 크기가 작고 실행 속도가 빠름)

# 레퍼런스 이야기

- 레퍼런스 = 별명(alias)  
= 변수의 또 다른 이름
- 레퍼런스에 의한 호출  
(call-by-reference)

```
#include <stdio.h>

void main() {
    int a = 2;
    int& babo = a;

    printf("%d\n", a);
    printf("%d\n", babo);

    getchar();
}
```

# 함수 호출 이야기

- 값에 의한 호출
- 주소에 의한 호출
- 레퍼런스에 의한 호출

# 요약

- 주소값, 포인터 상수의 의미를 이해할 수 있다.
- 포인터 변수를 이해할 수 있다.
- 포인터를 이용하여 변수, 배열, 함수를 액세스할 수 있다.
- 레퍼런스를 이해할 수 있다.