

7장

레퍼런스와 포인터, 그리고 함수호출 방법

변영철 교수

(ycb@jejunu.ac.kr)

제1절 변수와 함수, 주소 이야기

- 컴퓨터 = [CPU + 메인 메모리 + HDD + ...]
- CPU = CU + PU + 레지스터들(작은 메모리들)
- 몇 비트 컴퓨터? 32비트? 64비트?

제1절 변수와 함수, 주소 이야기

- 컴퓨터 가상 메모리 주소는 4G
 - 0000 0000 ~ FFFF FFFF
- 주소는 메모리에서 변수(배열, 함수)가 만들어진 곳 = 숫자
- 주소값(address)
 - '&변수명'은 그 변수가 위치한 메모리 주소값(숫자)
 - '배열명'은 그 배열이 위치한 메모리 주소값(숫자)
 - '함수명'은 함수 코드가 있는 메모리 주소값(숫자)

제1절 변수와 함수, 주소 이야기

- 변수, 배열, 함수가 있는 영역
 - 스택 영역 : 지역 변수(배열)
 - 전역 영역 : 전역 변수(배열)와 상수
 - 코드 영역 : 함수
 - 힙(heap) 영역 : 스택과 전역 공간 사이에 있는 비어있는 공간 -> new 동적 할당 연산자

제2절 포인터 이야기

- 포인터(변수)는 주소값을 저장하는 변수
- 포인터를 사용하는 이유 3가지
 - 포인터를 이용하면 편리한 경우가 있어서 (함수 포인터)
 - 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)

제2절 포인터 이야기

- 변수 포인터, 배열 포인터, 함수 포인터
- C 혹은 C++ 언어에서 배열의 첨자가 항상 0부터 시작하는 이유 -> 포인터를 배열처럼 사용하기 위하여

array

0	1	2
3	4	5

- 소중압출 소제거의 원칙

포인터 상수(숫자) 3가지

```
int a; -> &a  
int array[3];  
void say() {  
}
```



* p;

```
int a
char b
float c
int array[3]
void say()
```


 * p;

int a * p;

char b * p;

float c * p;

int array[3] * p;

void say() * p;



* p;

int * p;

char * p;

float * p;

int [3] * p;

void () * p;

 * p;

int * p;

char * p;

float * p;

int (* p)[3];

void (* p)();

포인터

1차원 배열 포인터

```
int a=7;  
int * p;  
p = &a;
```

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

```
int array[2]={7,8};  
int * p;  
p = a;
```

```
printf("%d\n", *p);  
printf("%d\n", *(p+0));  
printf("%d\n", *(p+1));
```

2차원 배열 포인터

0	1	2
3	4	5

```
int array[2][3]={{0,1,2},{3,4,5}};
```

```
int (* p)[3];
```

```
p = a;
```

```
*p
```

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

제3절 레퍼런스 이야기

`int& babo = a;`

- 레퍼런스 = 별명(alias) = 변수의 또 다른 이름
- 레퍼런스를 선언할 때는 선언과 동시에 초기화를 해야
- 레퍼런스 앞에 const라는 키워드를 붙이면 레퍼런스 상수 : 오로지 읽을 수만 있음
- 레퍼런스에 의한 호출(call-by-reference)

제4절 함수 호출 이야기

- 함수 호출 방법
 - 함수 호출 시 값을 넘겨주면 '값에 의한 호출'
 - 주소를 넘겨주면 '주소에 의한 호출'
 - 레퍼런스로 받으면 '레퍼런스에 의한 호출'

Call by value

```
#include <stdio.h>
void Swap(int i, int j) {
    int tmp;
    tmp = i;
    i = j;
    j = tmp;
}
void main() {
    int a = 2;
    int b = 3;
    printf("%d, %d\n", a, b);
    Swap(a, b);
    printf("%d, %d\n", a, b);
    getchar();
}
```