C 프로그래밍 프로젝트

Chap 14. 포인터와 함수에 대한 이해

2013.10.09.

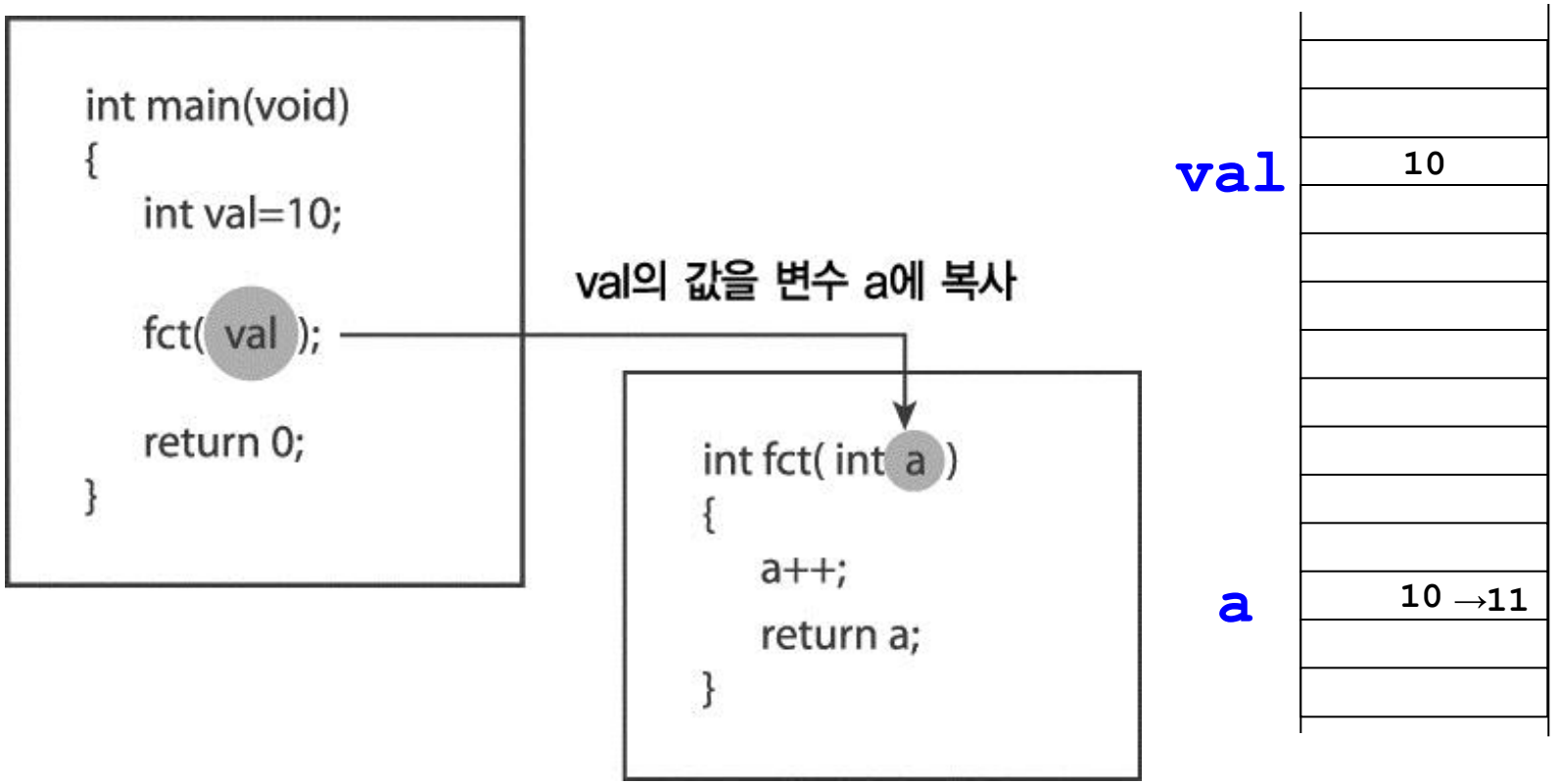
오 병 우

컴퓨터공학과

kit 국립금오공과대학교
Kumoh National Institute of Technology

14-1 함수의 인자로 배열 전달

- 기본적인 인자의 전달 방식
  - ◈ 값의 복사에 의한 전달



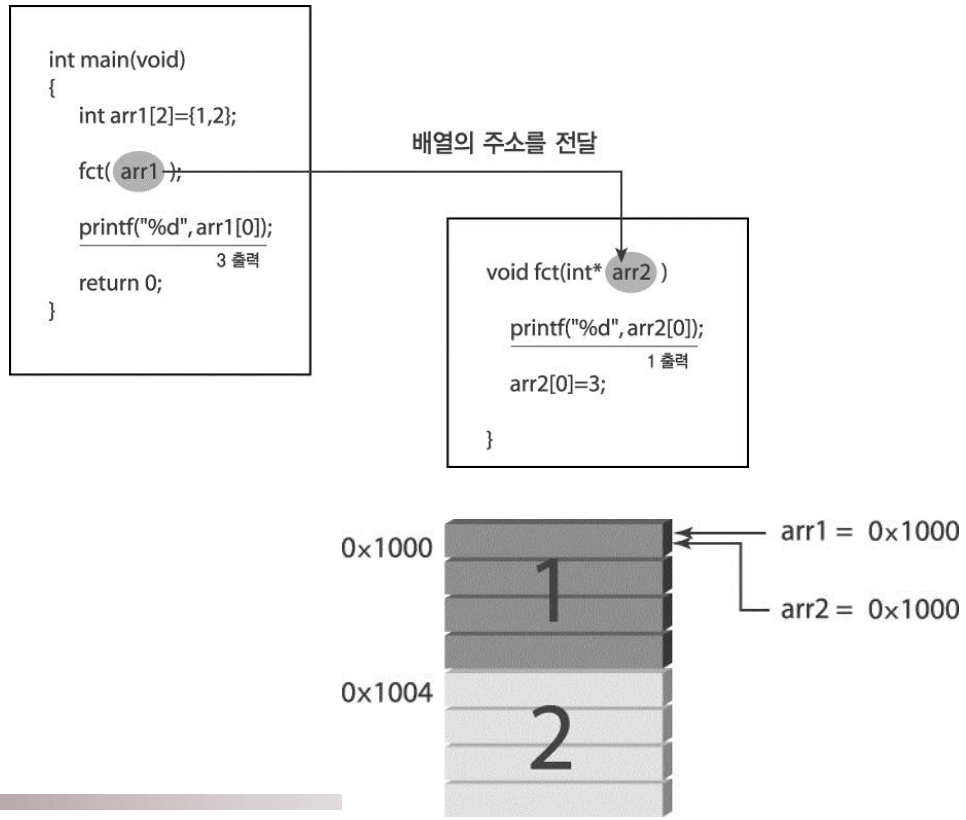
14-1 함수의 인자로 배열 전달

- 배열의 함수 인자 전달 방식
  - ◈ 배열 이름(배열 주소, 포인터)에 의한 전달

```
#include <stdio.h>
void fct(int *arr2);

int main(void)
{
    int arr1[2]={1, 2};

    fct(arr1);
    printf("%d \n", arr1[0]);
    return 0;
}
void fct(int *arr2)
{
    printf("%d \n", arr2[0]);
    arr2[0]=3;
}
```



14-1 함수의 인자로 배열 전달

- 배열 이름, 포인터의 sizeof 연산
  - 배열 이름 : 배열 전체 크기를 바이트 단위로 반환
  - 포인터 : 포인터의 크기(4)를 바이트 단위로 반환

```
#include <stdio.h>

int main(void)
{
   int arr[5];
   int* pArr=arr;

   printf("%d \n", sizeof(arr) );
   printf("%d \n", sizeof(pArr) );
   return 0;
}
```

14-1 함수의 인자로 배열 전달

```
#include <stdio.h>
int ArrAdder(int *, int);

int main()
{
    int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int SumOfArr;
    SumOfArr = ArrAdder(arr, sizeof(arr)/sizeof(int));
    printf("배열의 총 합: %d\n", SumOfArr);

    return 0;
}
int ArrAdder (int * pArr, int n)
{
    int sum = 0;
    for (int i=0; i<n; ++i)
        sum += pArr[i];
    return sum;
}
```

14-1 함수의 인자로 배열 전달

- "int * pArr" vs. "int pArr[ ]"
  - 둘 다 같은 의미를 지닌다.
  - 선언 "int pArr[ ]"은 함수의 매개 변수 선언 시에만 사용 가능

```
int function(int pArr[])
{
   int a=10;
   pArr=&a; // pArr이 다른 값을 지니게 되는 순간
   return *pArr;
}
```

14-2 호출 방식

- Call-By-Value
  - 값의 복사에 의한 함수의 호출
  - 가장 일반적인 함수 호출 형태

C 동영상 강좌 중에서 포인터 부분을 CD에 구워서 준다.

```
#include <stdio.h>
int add(int a, int b);

int main(void)
{
    int val1=10;
    int val2=20;
    printf(" 결 과 : ", add(val1, val2));

    return 0;
}
int add(int a, int b)
{
    return a+b;
}
```

14-2 호출 방식

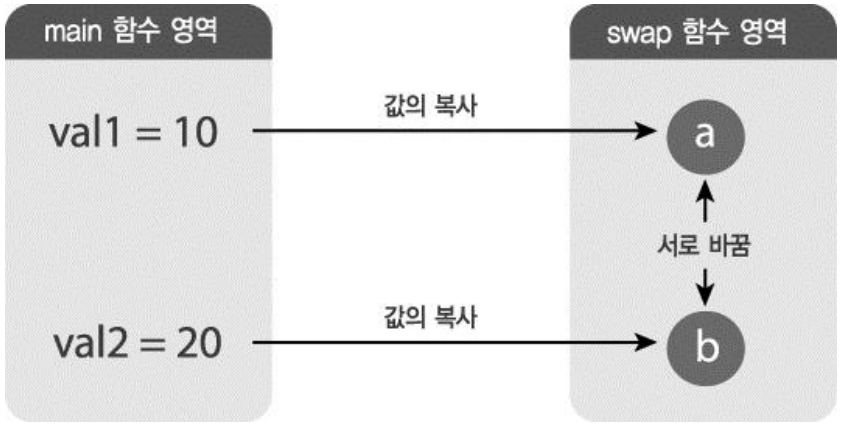
- Call-By-Value에 의한 swap

```
int main(void)
{
    int val1=10;
    int val2=20;
    swap(val1, val2);

    printf("val1 : %d \n", val1);
    printf("val2 : %d \n", val2);
    return 0;
}
void swap(int a, int b)
{
    int temp=a;
    a=b;
    b=temp;

    printf("a : %d \n", a);
    printf("b : %d \n", b);
}
```

여기에선
안바뀜

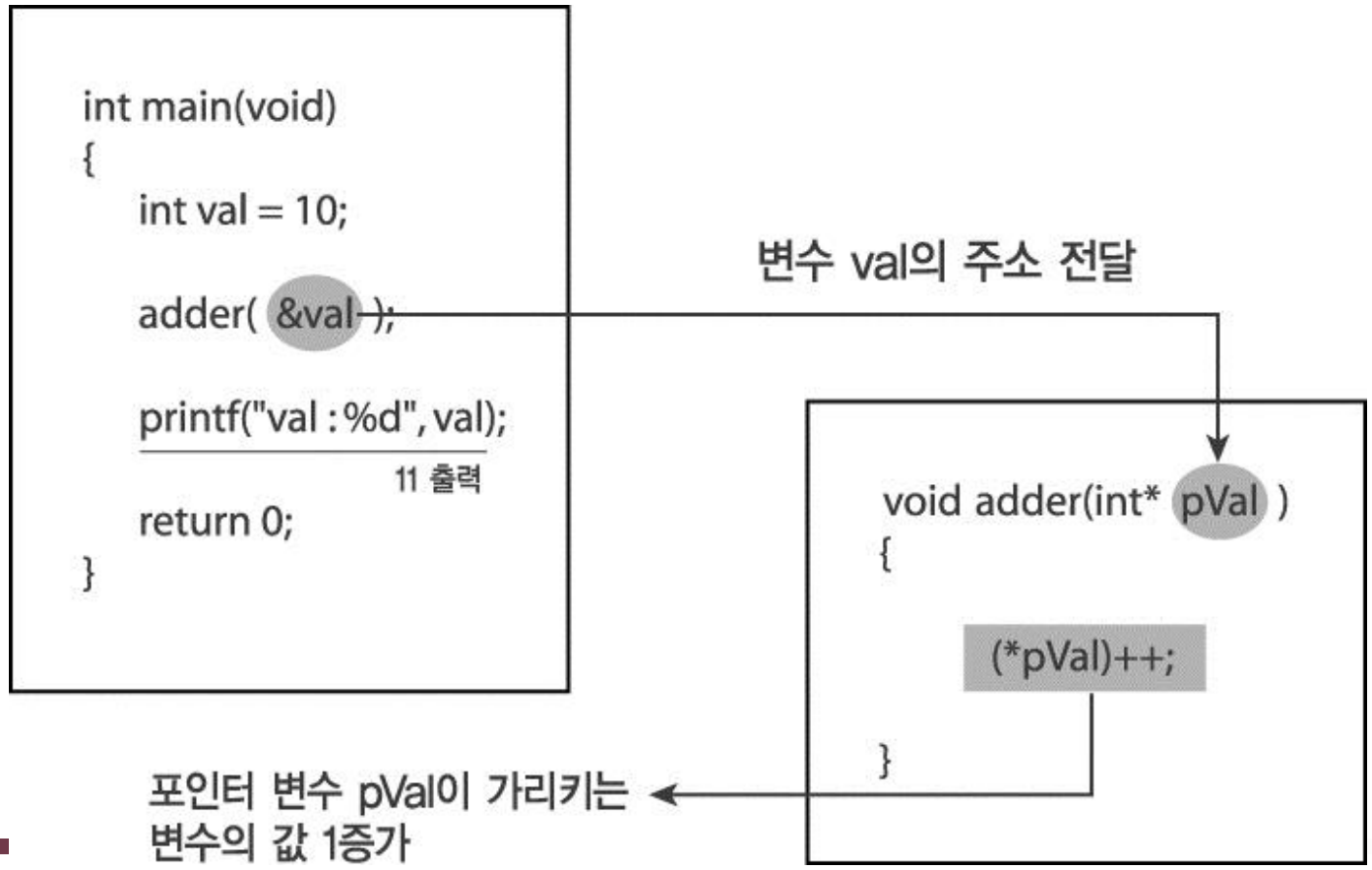


14-2 호출 방식

C 동영상 강좌의 네트워크 주소를 알려준다.

Call-By-Reference

- 참조(참조를 가능케 하는 주소 값)를 인자로 전달하는 형태의 함수 호출



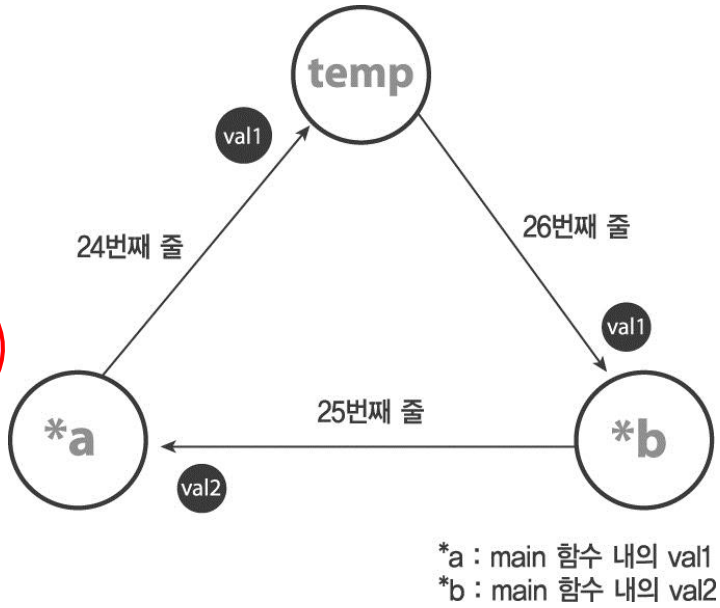
14-2 호출 방식

- Call-by-Reference에 의한 swap

```
int main(void)
{
    int val1=10;
    int val2=20;
    printf("Before val1 : %d \n", val1);
    printf("Before val2 : %d \n", val2);
    swap(&val1, &val2);      //val1, val2 주소 전달

    printf("After val1 : %d \n", val1);
    printf("After val2 : %d \n", val2);
    return 0;
}
void swap(int* a, int* b)
{
    int temp=*a;
    *a=*b;
    *b=temp;
}
```

따라 바뀜



14-2 호출 방식

- scanf 함수 호출 시 &를 붙이는 이유
  - case 1

```
int main(void)
{
    int val;
    scanf("%d", &val);
    . . . . .
```

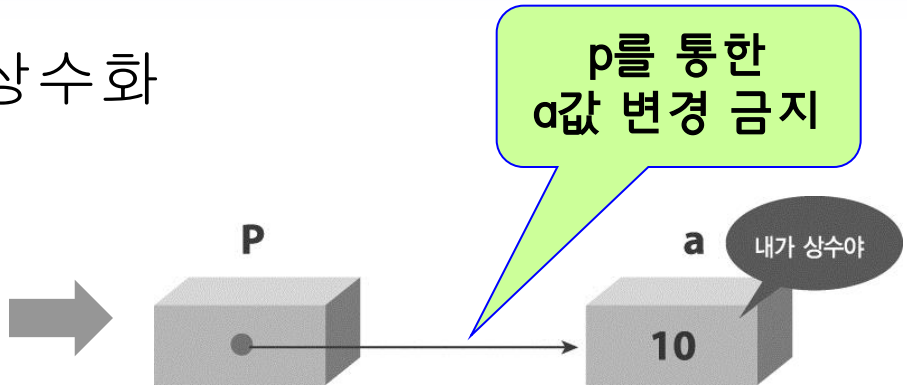
  - case 2

```
int main(void)
{
    char str[100];
    printf("문자열 입력 : ");
    scanf("%s", str);
    . . . . .
```

14-3 포인터와 const

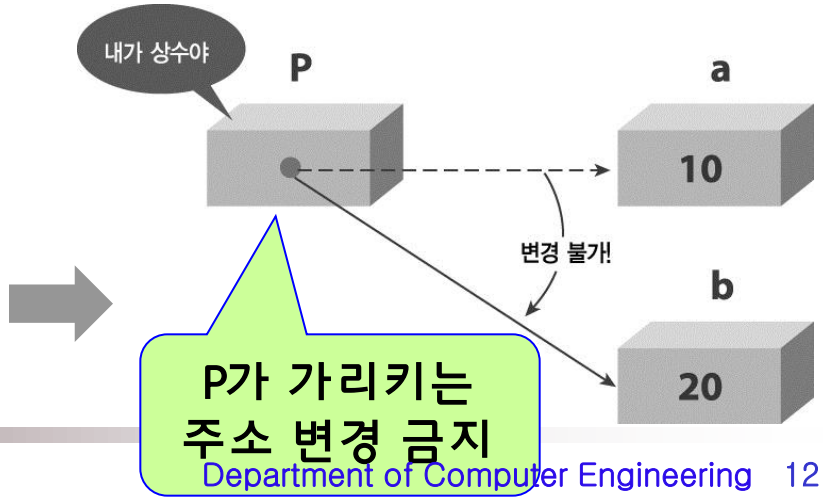
- 포인터가 가리키는 변수의 상수화

```
int a = 10;
const int * p = &a;
*p=30          // Error!
a=30           // OK!
```



- 포인터 상수화

```
int a=10;
int b=20;
int * const p = &a;
p=&b           // Error!
*p=30          // OK!
```



14-3 포인터와 const

- const 키워드를 사용하는 이유
  - 컴파일 시 잘못된 연산에 대한 에러 메시지
  - 프로그램을 안정적으로 구성

```
#include <stdio.h>
float PI=3.14;

int main(void)
{
    float rad;
    PI=3.07;   // 분명히 실수!!

    scanf("%f", &rad);
    printf("원의 넓이는 %f \n",
                        rad*rad*PI);
    return 0;
}
```

```
#include <stdio.h>
const float PI=3.14;

int main(void)
{
    float rad;
    PI=3.07;   // Compile Error 발생!

    scanf("%f", &rad);
    printf("원의 넓이는 %f \n",
                        rad*rad*PI);
    return 0;
}
```

연습문제

- 다음 함수에서 const의 역할은?

```
void print (const int* arr, int size)
{
        int i;
        for (i=0; i<size; ++i)
                printf("%d ", arr[i]);
}
```

arr을 통한 배열 요소 변경 금지

- 문제점을 찾아 보자.

```
void print (const int* ptr)
{
        int *p = ptr;
        *p = 20;
        printf("%d ", *ptr);
}
```

const int * 타입은 int *로 대입이 허용되지 않음

const int *로는 대입이 허용됨

Summary

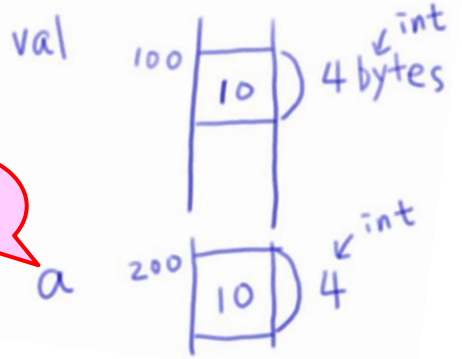
가능하면 변경된 값은 리턴값으로 받으려고 노력할 것. 넘겨준 값을 함수 내에서 바꾸는 Side Effect(부작용)는 좋지 않음

여러 개의 값을 함수로부터 받을 때는 사용해야 함

- 함수 내에서 값을 바꾸고 그 값을 호출한 놈이 써야 한다면 포인터를 써라.
  - Call-By-Reference라고 한다.

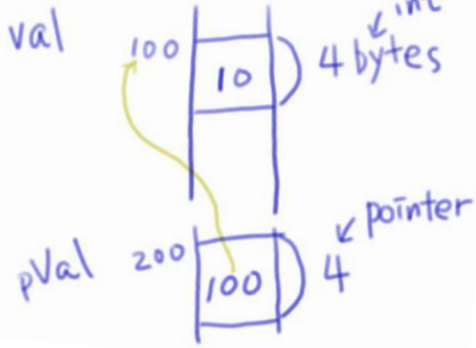
Call-By-Value

```
{
    int val = 10;
    fct(val);
}
void fct(int a)
{
}
```



Call-By-Reference

```
{
    int val = 10;
    fct(&val);
}
void fct(int *pVal)
{
}
```



- 값을 변경할 수 없도록 하려면 const 키워드를 사용한다.