C언어 (CSE2035)
(Chap2. Pointers) (2/2)

Sungwon Jung, Ph.D.

Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

Pointers and functions

- **C는 함수를 호출할 때 기본적으로 "값에 의한 호출(call-by-value)" 메커니즘을 사용한다**
  - caller(호출하는 쪽)에서 callee(호출되는 쪽)에 parameter를 넘길 때 변수의 값을 넘겨주는 방식

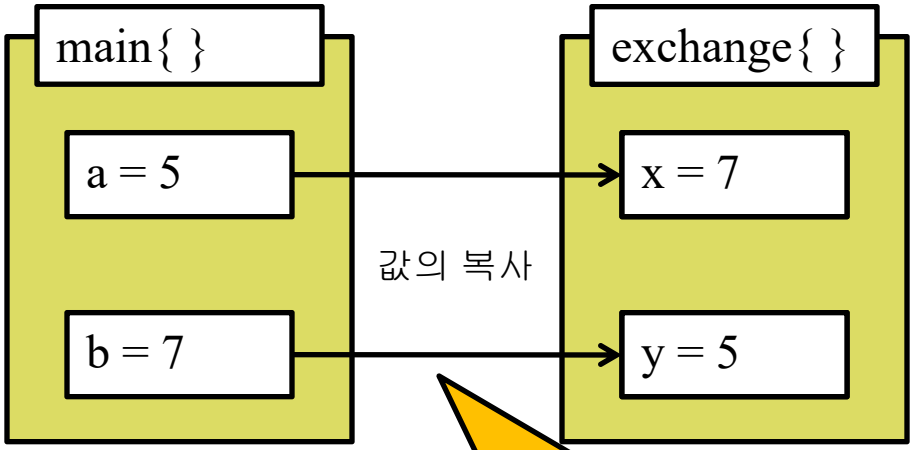
- **하지만 이 방법은 callee가 넘겨받은 parameter의 값을 변경하더라도 caller에 영향을 미치지 못한다.**
  - 그래서 callee에서 caller의 변수를 변경해야 할 필요가 있을 때에는 적절히 동작하지 않는다.

Pointers and functions

- 값에 의한 호출(Call by Value) 로 인해 발생하는 문제

```
/* Prototype Declarations */
void exchange (int x, int y);

int main (void)
{
  int  a = 5;
  int  b = 7;
  exchange (a, b);
  printf("%d %d\n", a, b);
  return 0;
} /* main */
```

a, b의 값을 서로 바꿀 것을 기대



exchange( )함수를 호출한 후에도 a와 b의 변수값이 바뀌지 않는다.

```
void exchange (int x,
               int y)
{
  int temp;

  temp = x;
  x    = y;
  y    = temp;
  return;
} /* exchange */
```

x와 y의 값을 서로 바꿈

Pointers and functions

- 참조에 의한 호출(Call-by-reference)
  - 참조에 의한 호출은 caller에서 callee에 parameter를 넘길 때
    - 변수의 값을 넘겨주는 대신 변수의 주소를 넘겨주는 방식
  - callee에서 caller의 변수에 대한 주소를 가지고 있기 때문에 callee에서 caller의 변수의 값을 변경할 수 있다.
  - 참조에 의한 호출을 사용하면 앞에서와 같은 문제를 피할 수 있다.

- "참조에 의한 호출"의 효과를 얻는 방법
  - 함수(callee)의 paramener를 포인터형으로 선언
  - Caller에서 함수(callee)를 호출할 때 parameter로 주소를 전달
  - 함수(callee) 내부에서 parameter 사용시 역참조 연산자(*) 사용
    - 넘겨받은 값이 주소값이므로 그 값을 사용할 때는 당연히 역참조연산자를 사용해야 한다.

Pointers and functions

- **Call by reference** 방법을 이용하여 해결한 방법

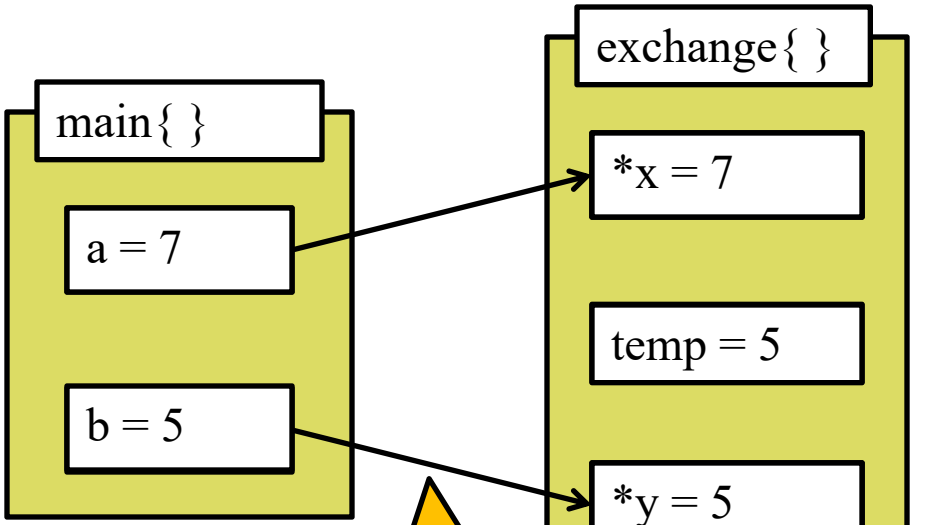
```
/* Prototype Declarations
void exchange (int *, int *);

int main (void)
{
  int  a = 5;
  int  b = 7;
  exchange (&a, &b);
  printf("%d %d\n", a, b);
  return 0;
} /*
```

a, b의 주소를 넘겨준다

포인터 변수를 parameter로 사용

```
void exchange (int *x,
               int *y)
{
  int temp;

  temp = *x;
  *x   = *y;
  *y   = temp;
  return;
} /* exchan
```

역참조 연산자(*)를 사용



callee의 변수 x, y가 각각 caller의 변수 a, b의 주소를 가지고 있기 때문에 x가 가리키는 변수와 y가 가리키는 변수의 값이 바뀔 때, a, b의 값도 바뀐다.

Pointers and functions

- 예제 프로그램 - Call by reference

```c
#include <stdio.h>

void exchange(int *, int *);

int main(void)
{
        int a = 5;
        int b = 7;
        printf("<<Before>>\n");
        printf("Value    : a=%d b=%d\n", a, b);
        printf("Address : a=%d b=%d\n\n", &a, &b);
        exchange(&a, &b);
        printf("<<After>>\n");
        printf("Value    : a=%d b=%d\n", a, b);
        return 0;
}

void exchange(int *x, int *y)
{
        int temp;

        printf("<<In Function>>\n");
        printf("Value    : x=%d y=%d temp=%d\n", *x, *y);
        printf("Address : x=%d y=%d\n\n", x, y);
        temp = *x;
        printf("Step1(temp=*x) : *x=%d *y=%d temp=%d\n", *x, *y, temp);
        *x = *y;
        printf("Step2(*x=*y)    : *x=%d *y=%d temp=%d\n", *x, *y, temp);
        *y = temp;
        printf("Step3(*y=temp) : *x=%d *y=%d temp=%d\n\n", *x, *y, temp);

        return;
}
```

```
<<Before>>
Value    : a=5 b=7
Address : a=-1076611136 b=-1076611140

<<In Function>>
Value    : x=5 y=7 temp=-1076611176
Address : x=-1076611136 y=-1076611140

Step1(temp=*x) : *x=5 *y=7 temp=5
Step2(*x=*y)    : *x=7 *y=7 temp=5
Step3(*y=temp) : *x=7 *y=5 temp=5

<<After>>
Value    : a=7 b=5
```

주소를 그대로 넘겨받는다.

함수 수행 후 a, b 의 값이 바뀐다.

Pointers and functions

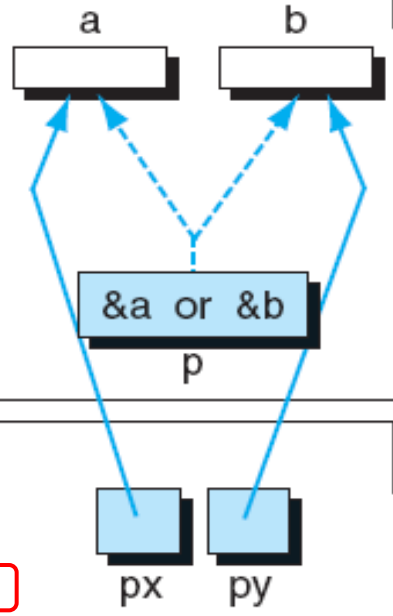
- **Functions returning pointer**
  - 다음 프로그램과 같이 함수가 pointer 변수를 리턴할 수도 있다.
  - 이 경우에는 함수의 헤더 부분에서 return type을 pointer형으로 해주어야 한다.
  - 지역 변수에 대한 참조를 리턴하는 것은 심각한 오류를 불러올 수 있다.

Return type이 pointer형이다.

Pointer type을 return한다. (조건식의 결과에 따라 작은 값을 가진 변수에 대한 참조를 return한다.)

```
/* Prototype Declarations */
int *smaller (int *p1, int *p2);

int main (void)
  …
  int    a;
  int    b;
  int   *p;
  …
  scanf ( "%d %d", &a, &b );
  p = smaller (&a, &b);
  …
```

```
int *smaller (int *px,
              int *py)
{
  return (*px < *py ? px : py);
} /* smaller */
```



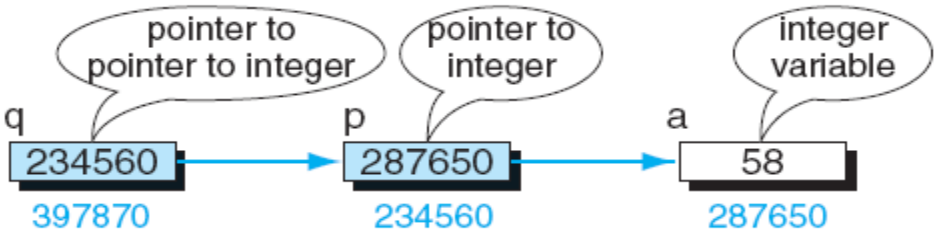
Pointers to pointer

- 포인터 변수가 다른 포인터 변수를 참조할 수도 있다.
  - 다음 프로그램은 int형 변수 a를 참조하는 포인터 변수 p와, 포인터변수 p를 참조하는 포인터변수 q에 대한 예제이다.

Pointer변수를 참조하는 pointer 변수의 선언

p는 a를, q는 p를 참조한다.

```
/* Local Declarations */
int      a;
int     *p;
int    **q;
/* Statements */
a = 58;
p = &a;
q = &p;
printf(" %3d",    a);
printf(" %3d",   *p);
printf(" %3d", **q);
```

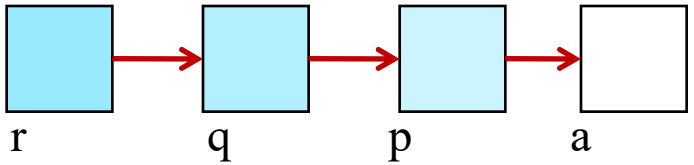


- integer형 포인터 변수를 참조하는 포인터 변수 q의 선언 방법은 다음과 같다.
  - int **q;
- q는 p를 통해 a를 참조할 수 있다.
- q를 이용하여 a를 참조하기 위해서는 두 단계를 거쳐야 하므로 **q와 같이 역참조 연산자를 두 번 사용한다.
- 따라서 a, *p, **q에 의해 출력되는 결과는 모두 58 이다.

Pointers to pointer

- 예제 프로그램 - 포인터의 포인터를 사용한 예

```c
#include <stdio.h>

int main(void)
{
        int a;
        int *p;
        int **q;
        int ***r;

        p = &a;
        q = &p;
        r = &q;

        printf("Enter a number : ");
        scanf("%d", &a);
        printf("\nThe number is %d\n", a);

        printf("\nEnter a number : ");
        scanf("%d", p);
        printf("\nThe number is %d\n", a);

        printf("\nEnter a number : ");
        scanf("%d", *q);
        printf("\nThe number is %d\n", a);

        printf("\nEnter a number is : ");
        scanf("%d\n", **r);
        printf("\nThe number is %d\n", a);

        return 0;
}
```



scanf의 입력 값은 모두 a에 저장된다.

```
[root@mclab chap9]# vi chap9-3.c
[root@mclab chap9]# gcc -o chap9-3 chap9-3.c
[root@mclab chap9]# ./chap9-3
Enter a number : 1
The number is 1

Enter a number : 2
The number is 2

Enter a number : 3
The number is 3

Enter a number : 4
The number is 4
[root@mclab chap9]#
```

Compatibility

- 포인터 변수는 선언될 때 어떤 **type**의 변수를 참조할 지가 미리 정해진다. 즉, 포인터 변수도 **type**을 갖는다.
  - 예제 프로그램 - char, int, double type에 대한 포인터 변수의 size를 출력

```c
#include <stdio.h>

int main(void)
{
        char c;
        char *pc;
        int sizeofc      = sizeof(c);
        int sizeofpc     = sizeof(pc);
        int sizeofStarpc = sizeof(*pc);

        int a;
        int *pa;
        int sizeofa      = sizeof(a);
        int sizeofpa     = sizeof(pa);
        int sizeofStatpa = sizeof(*pa);

        double x;
        double *px;
        int sizeofx      = sizeof(x);
        int sizeofpx     = sizeof(px);
        int sizeofStarpx = sizeof(*px);

        printf("sizeof(c)  : %3d | ", sizeofc);
        printf("sizeof(pc) : %3d | ", sizeofpc);
        printf("sizeof(*pc): %3d\n", sizeofStarpc);

        printf("sizeof(a)  : %3d | ", sizeofa);
        printf("sizeof(pa) : %3d | ", sizeofpa);
        printf("sizeof(*pa): %3d\n", sizeofStarpa);

        printf("sizeof(x)  : %3d | ", sizeofx);
        printf("sizeof(px) : %3d | ", sizeofpx);
        printf("sizeof(*px): %3d\n", sizeofStarpx);

        return 0;
}
```

```
[root@mclab chap9]# vi chap9-4.c
[root@mclab chap9]# gcc -o chap9-4 chap9-4.c
[root@mclab chap9]# ./chap9-4
sizeof(c)  :   1 | sizeof(pc) :   4 | sizeof(*pc):   1
sizeof(a)  :   4 | sizeof(pa) :   4 | sizeof(*pa):   4
sizeof(x)  :   8 | sizeof(px) :   4 | sizeof(*px):   8
[root@mclab chap9]#
```

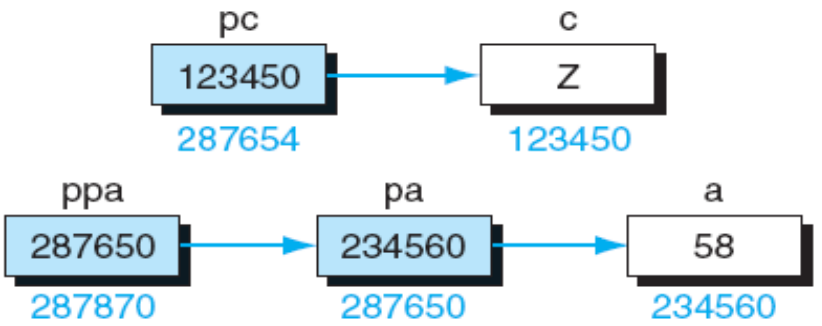
Compatibility

- **Pointer 변수는 참조하는 type에 상관 없이 모두 4byte이다.**
  - 참조하는 변수에 상관 없이 주소를 저장할 크기를 갖는다.

- **앞의 프로그램은 각각의 포인터를 초기화하지 않았다.**
  - 각각의 포인터가 참조하는 곳의 size는 서로 다르다(1, 2, 12).
  - 이유는 각각의 포인터가 선언될 때, 참조할 변수의 type이 정해져 있기 때문이다.

Compatibility

- 포인터 변수에 다른 타입의 주소를 저장하면 **error**가 발생한다.



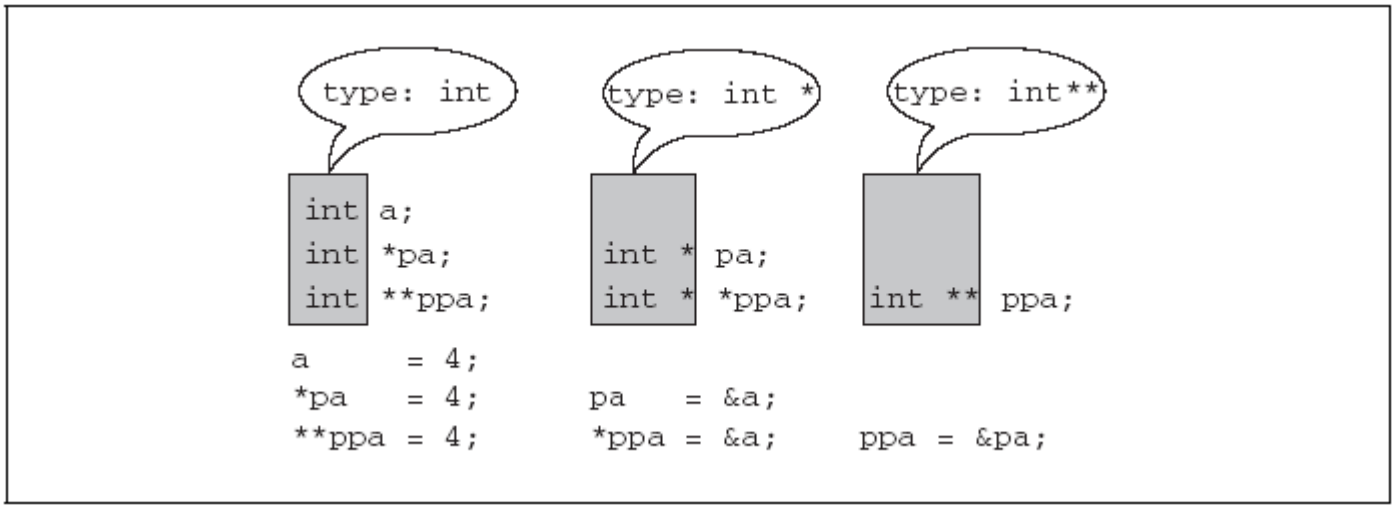

```
char c;
char *pc;

int   a;
int  *pa;
int **ppa;

pc  = &c;                  /* Good and valid    */
pa  = &a;                  /* Good and valid    */
ppa = &pa;                 /* Good and valid    */

/* Invalid pointers will generate errors     */
pc  = &a;                  /* Different types   */
ppa = &a;                  /* Different levels  */
```

Char 타입 포인터에 int형 변수의 주소를 저장 → error!

int * 타입 포인터에 int형 변수의 주소를 저장 → error!

Compatibility

- **Pointer types must match !!**



Compatibility

- **Void pointer**
  - 포인터 변수도 type을 가지고 있기 때문에 다른 type의 변수를 참조하도록 한다면 compile error가 발생한다.
  - Void pointer는 임의의 type을 갖지 않기 때문에 어떤 type이든 참조할 수 있다.
  - 선언 방법 : void *pVoid;

- **Casting pointer**
  - 선언된 포인터 변수가 다른 타입의 변수를 참조할 수 있도록 강제적인 형 변환이 가능하다.
  - Ex) int a;
    char *p;
    p = (char *)&a;
  - 이러한 형 변환은 메모리의 낭비를 불러올 수 있다.

Compatibility

- **Casting pointer**
  - 다음은 void pointer를 이용한 변수의 참조와 casting을 통한 참조의 예제이다.

```
/* Local Definitions */
   void *pVoid;
   char *pChar;
   int  *pInt;
/* Statements */
   pVoid  = pChar;
   pInt   = pVoid;
   pInt   = (int *) pChar;
```

  - 둘 다 pChar을 pInt에 저장하고 있다.
    - 첫 번째 경우는 다른 type의 pointer 변수라도 참조가 가능한 void 타입 포인터를 이용했다.
    - 두 번째 경우는 type casting을 이용했다.

Example program

- 다음은 초를 시간으로 바꾸는 함수이다.

```
int secToHours(long time, int *hours, int *minutes, int *seconds)
{
        long localTime;

        localTime = time;
        *seconds  = localTime % 60;
        localTime = localTime / 60;

        *minutes  = localTime % 60;
        *hours    = localTime / 60;

        if(*hours > 24)
                return 0;
        else
                return 1;
}
```

시간, 분, 초 3개의 값을 caller에게 알려줘야 하는데, Return문은 하나의 값만을 반환할 수 있으므로 call-by-reference를 이용한다.

Parameter인 역참조 연산자를 이용하여 hours, minutes, seconds가 참조하는 곳에 저장하면 caller쪽의 변수에 저장 된다.