2010-1학기 프로그래밍입문(1)

chapter 06-2 참고자료

포인터

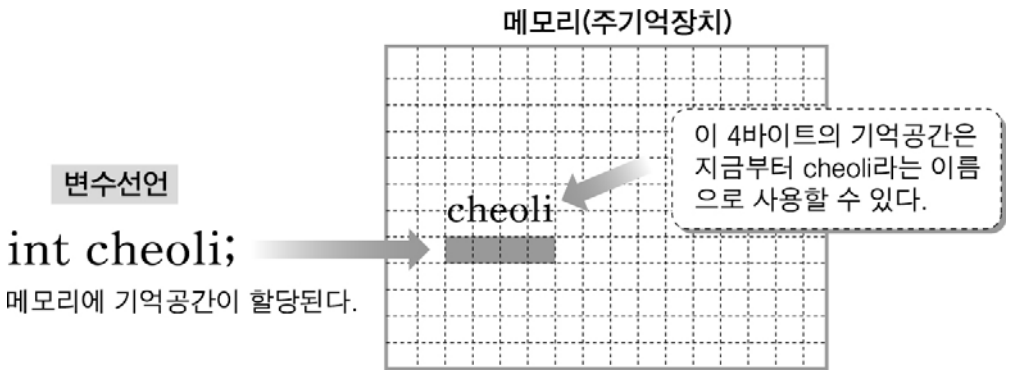
박 종 혁

Tel: 970-6702
Email: jhpark1@snut.ac.kr

출처: 뇌를 자극하는 C프로그래밍, 한빛미디어

❑ 포인터의 정의와 사용

- 변수를 선언하는 것은 메모리에 기억공간을 할당하는 것이며 할당된 이후에는 변수명으로 그 기억공간을 사용한다.



변수명으로 기억공간을 사용한다.

- cheoli = 10; → 할당된 기억공간에 10을 저장한다.
- metel = cheoli; → cheoli의 값을 다른 기억공간에 복사한다.

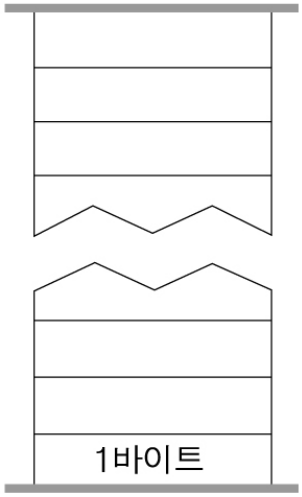
- 할당된 기억공간을 사용하는 방법에는 변수명 외에 메모리의 실제 주소값을 사용하는 것이다. 이 주소값을 포인터라고 한다.

▶ 포인터란 무엇인가?

- 메모리에는 바이트(byte)단위로 그 위치를 식별할 수 있는 물리적인 주소값이 있다.
  - 메모리의 용량이 64kb라면 주소값은 0번지부터 65535번지까지 존재한다.

$$64\ \text{kbyte} = 64 \times 1024 = 65536\ \text{byte} \quad (1k = 2^{10} = 1024)$$

메모리에는 바이트 단위로 위치를 식별할 수 있는 주소값이 있다.

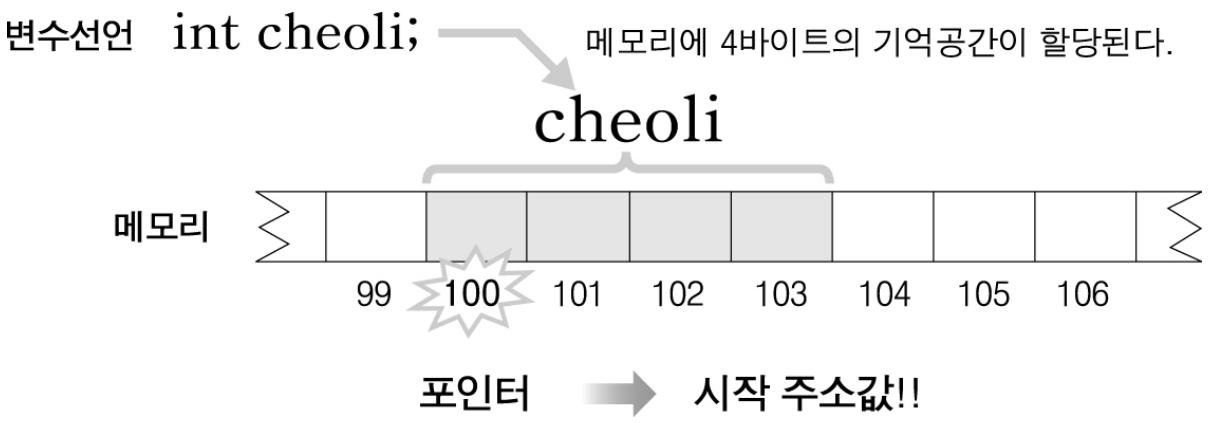


65535 번지	0xffff
65534 번지	0xfffe
65533 번지	0xfffd
⋮	⋮
2 번지	0x0002
1 번지	0x0001
0 번지	0x0000

16진수 주소값

▶ 포인터란 무엇인가?

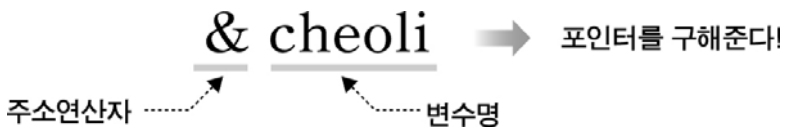
- 변수를 선언하면 그 자료형의 크기만큼 메모리에 연속된 바이트의 기억공간이 할당되는데 그 첫번째 바이트의 주소값이 포인터이다.



- 이 포인터를 사용하여 4바이트의 기억공간에 값을 저장하거나 저장된 값을 꺼내어 쓸 수 있다.

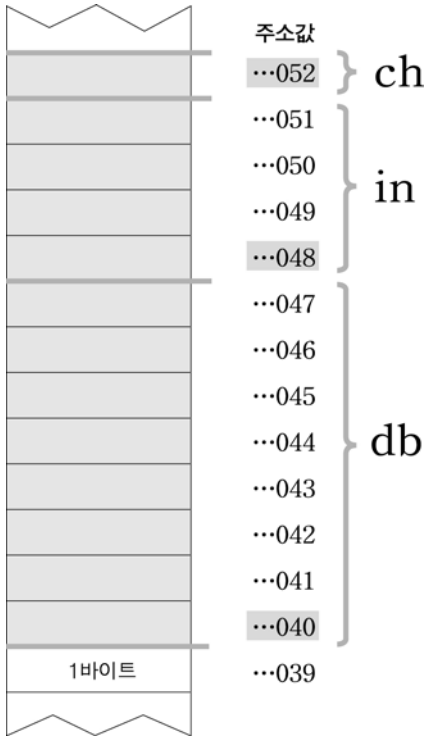
▶ 포인터를 구하자(주소연산자)

- 특정 변수의 포인터를 구하기 위해서는 주소연산자(&)를 사용한다.



- 포인터를 구하여 출력해 보자.

```
char ch;
int in;
double db;

printf("ch의 포인터 : %u₩n", &ch);
printf("in의 포인터 : %u₩n", &in);
printf("db의 포인터 : %u₩n", &db);
```




```
ch의 포인터 : 1245052    // char형 변수의 주소값
in의 포인터 : 1245048    // int형 변수의 시작 주소값
db의 포인터 : 1245040    // double형 변수의 시작 주소값
```

▶ 포인터는 특정 자료형을 가리킨다.

- 포인터에는 자신이 어떤 자료형으로부터 만들어졌는지에 대한 정보를 가지고 있다.

포인터	값	혈통에 대한 정보
&ch	1245052	“나는 char형 기억공간의 주소값이다!”
&in	1245048	“나는 int형 기억공간의 시작 주소값이다!”
&db	1245040	“나는 double형 기억공간의 시작 주소값이다!”

- 포인터가 특정 자료형에 대한 정보를 가지고 있다는 것을 간단히 표현할 때 “가리킨다”고 하고 화살표를 사용하여 그림을 그린다.

▶ 포인터를 사용하자(참조연산자)

- 포인터를 통해서 기억공간을 사용하기 위해서는 참조연산자(*)를 사용한다.

```
char ch;
int in;
double db;
*&ch = 'P';          // 포인터 &ch가 가리키는 기억공간에 'P'를 저장한다.
*&in = 100;          // 포인터 &in이 가리키는 기억공간에 100을 저장한다.
*&db = 3.14;         // 포인터 &db가 가리키는 기억공간에 3.14를 저장한다.
printf("변수 ch에 저장된 문자 : %c₩n", ch);
printf("변수 in에 저장된 값 : %d₩n", in);
printf("변수 db에 저장된 값 : %lf₩n", db);
```

```
변수 ch에 저장된 문자 : P
변수 in에 저장된 값 : 100
변수 db에 저장된 값 : 3.140000
```

▶ 포인터를 사용하자(참조연산자)

- "참조"는 기억공간 뿐만 아니라 기억공간에 저장된 값도 사용한다.

```
int a=100, b=0;
b = *&a;        // 포인터 &a가 가리키는 기억공간의 값을 b에 대입한다.
printf("b의 값 : %d₩n", b);
```

- 기억공간을 사용하는 것과 값을 사용하는 것은 대입연산자의 어디에 위치하느냐에 따라 결정된다.

```
int a=10, b=20;
*&a = *&b;          // 변수 b에 저장된 값을 변수 a의 기억공간에 저장한다.
printf("a의 값 : %d₩n", a);       // a의 값은 20이 출력된다.
```

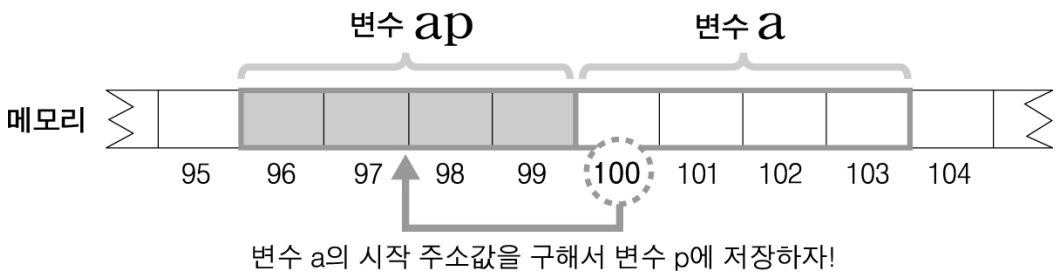
기억공간을 사용 (왼쪽) *&a = *&b ; 기억공간의 값을 사용 (오른쪽)

오른쪽의 값을 왼쪽의 기억공간에 저장한다.

▶ 포인터를 저장하자(포인터변수)

- 포인터의 값 자체는 정수값이지만 가리키는 자료형에 대한 정보를 가지고 있으므로 정수형 변수에 저장할 수 없다.

```
int a;          // 포인터를 구할 변수
int ap;         // 포인터를 저장할 변수
ap = &a;        // a의 포인터를 구해서 ap에 저장한다.
```



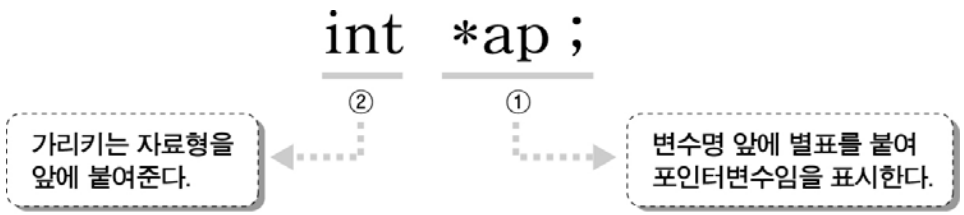
- 직관적으로는 충분히 가능할 듯 하지만 컴파일에러가 발생한다.

error C2440: '=' : cannot convert from 'int *' to 'int'

▶ 포인터를 저장하자(포인터변수)

- 포인터는 포인터가 가진 정보를 그대로 보존할 수 있도록 포인터변수에 저장해야 한다.

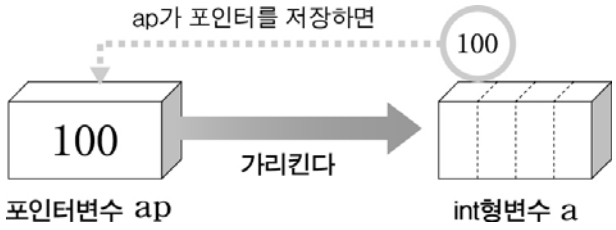
- 포인터변수는 변수명 앞에 '*'을 붙이고 가리키는 자료형을 앞에 적어준다.
  - int형 변수의 포인터를 저장하는 포인터변수의 선언



(제 이름은 ap이고요 포인터변수입니다. 저는 int형 변수의 시작주소값만을 저장할 수 있습니다. 그래서 int형 변수를 가리킨다고 말하지요.)

- 포인터변수가 포인터를 저장하면 포인터와 마찬가지로 기억공간을 가리킨다.

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



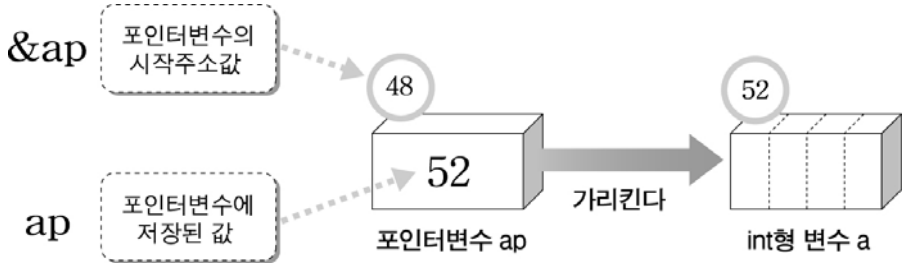
▶ 포인터변수를 사용한 참조

- 포인터를 저장한 포인터변수도 참조연산자로 그 것이 가리키는 기억공간 또는 그 기억공간의 값을 사용할 수 있다.

```
int a;          // int형 변수의 선언
int ap = &a;    // 포인터변수의 선언과 동시에 초기화, ap는 변수 a를 가리킨다.
*ap = 10;       // 포인터변수가 가리키는 기억공간에 10을 저장한다.
```

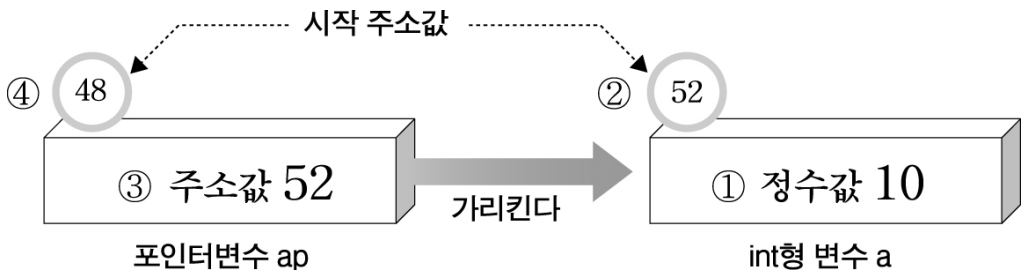
- 포인터변수도 하나의 변수이므로 주소연산자로 메모리에서의 위치를 구할 수 있다.

```
int a;          // int형 변수의 선언
int ap = &a;    // 포인터변수의 선언과 동시에 초기화, ap는 변수 a를 가리킨다.
printf("ap에 저장된 값 : %u₩n", ap);        // 변수 a의 시작주소값 출력
printf("ap자체의 주소값 : %u₩n", &ap);      // 포인터변수 ap의 시작주소값 출력
```



▶ 포인터 정리

```
int a = 10;        // int형 변수 선언, 정수값 10으로 초기화
int *ap = &a;      // int 포인터변수 선언, a의 시작주소값으로 초기화
```



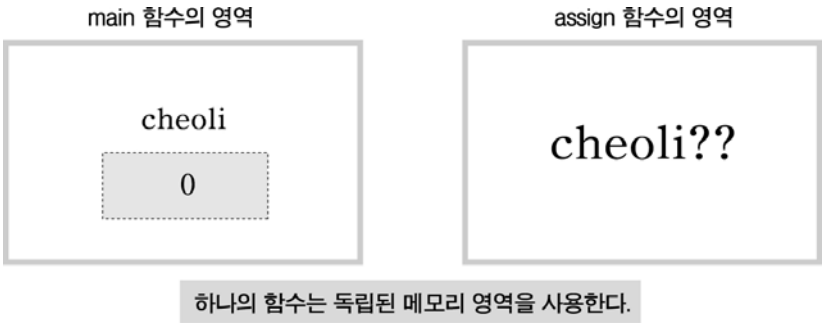

```
printf("%d", a);        // ①번 출력, a에 저장된 정수값 10
printf("%d", *ap);      // ①번 출력, ap가 가리키는 곳에 저장된 값 10
printf("%u", &a);       // ②번 출력, a의 시작주소값 52번지
printf("%u", ap);       // ③번 출력, ap에 저장된 주소값 52번지
printf("%u", &ap);      // ④번 출력, 포인터변수 ap의 시작주소값 48번지
```

❑ 포인터의 필요성

- 함수들은 독립된 기억공간을 가지므로 다른 함수에 선언된 변수를 사용할 수 없다.

  - assign함수를 호출하여 메인함수에 있는 cheoli변수에 값을 할당하는 예

```
#include <stdio.h>

void assign();

int main()
{
    int cheoli=0;
    assign();
    printf("함수가 호출된 후에 cheoli에 저장된 값 : %d₩n", cheoli);
    return 0;
}

void assign()
{
    cheoli=100;
}
```



함수가 호출된 후에 cheoli에 저장된 값 : 0

▶ 포인터로 다른 함수의 기억공간을 사용한다.

- assign함수가 main함수의 cheoli변수를 사용하기 위해서는 메모리에서의 위치(포인터)를 알아야 한다.

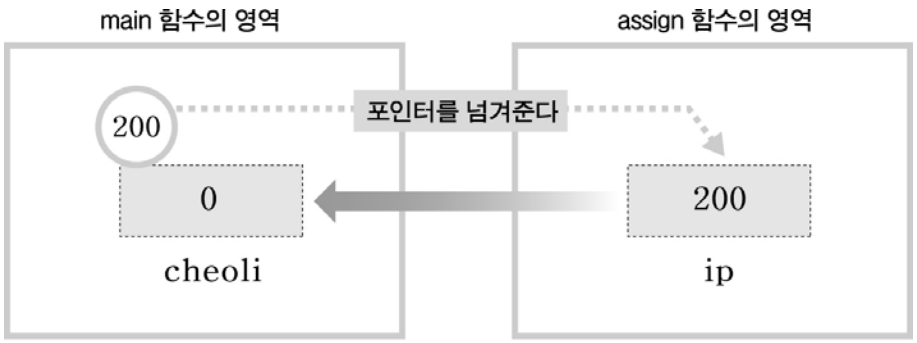
```
#include <stdio.h>

void assign(int *ip);

int main()
{
    int cheoli=0;
    assign(&cheoli);
    printf("함수가 호출된 후에 cheoli에 저장된 값 : %d₩n", cheoli);  // 100출력
    return 0;
}

void assign(int *ip)
{
    *ip=100;
}
```

main 함수에서 호출할 때

```
assign( &cheoli );        // 포인터를 구해서 전달인자로 넘겨준다.
```

assign 함수를 선언할 때

```
void assign( int *ip );   // 포인터변수를 선언하여 포인터를 받는다.
```



ip가 포인터를 저장하여 cheoli를 가리킨다.

(main 함수에 있는 cheoli 변수의 시작주소값은 200이라고 가정합니다.)

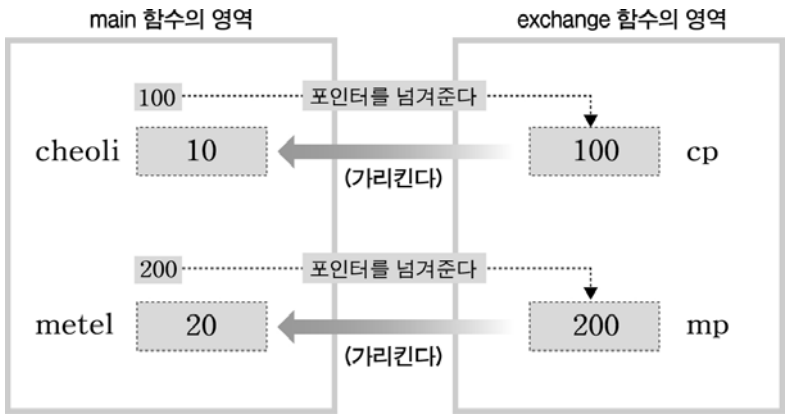
▶ 함수의 한계를 극복한다.

- 함수는 전달인자가 많아도 리턴되는 값은 오직 하나이다. 따라서 메인함수에 있는 두 변수의 값을 바꾸는 함수는 포인터를 사용해야 한다.

```
#include <stdio.h>

void exchange(int *, int *);

int main()
{
    int cheoli=10, metel=20;
    exchange(&cheoli, &metel); //  [1]
    return 0;
}

void exchange(int *cp, int *mp)
{
    int temp;
    temp=*cp;   // [2]
    *cp=*mp;    // [3]
    *mp=temp;   // [4]
}
```

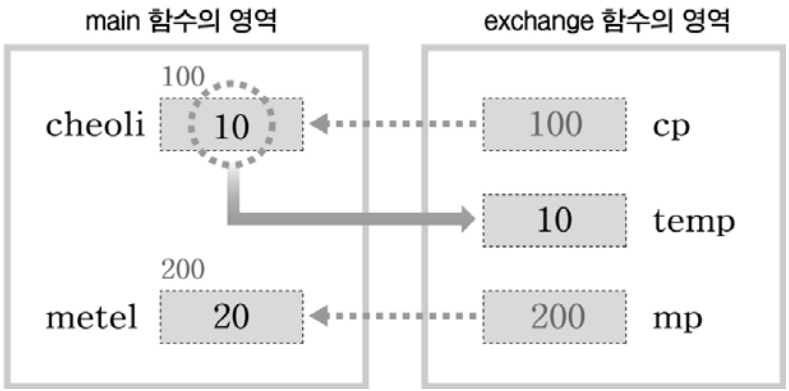
▶ 함수의 한계를 극복한다.

[1]



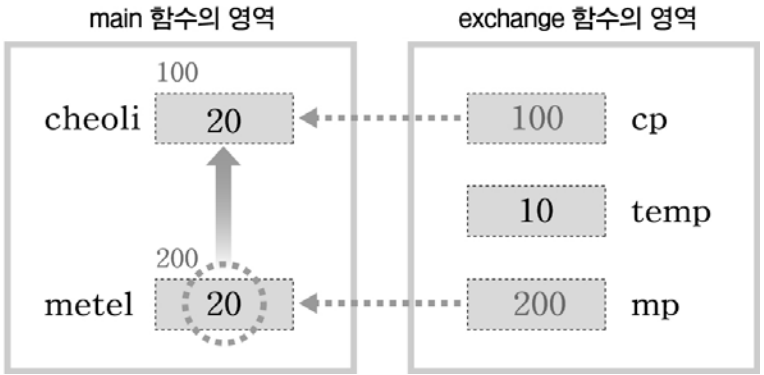
[2]

```
temp = *cp;
```



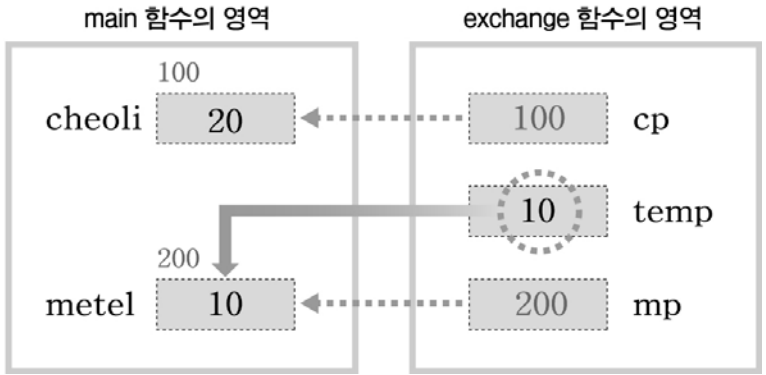
[3]

```
*cp = *mp;
```



[4]

```
*mp = temp;
```



❑ 포인터에 관한 중요한 질문

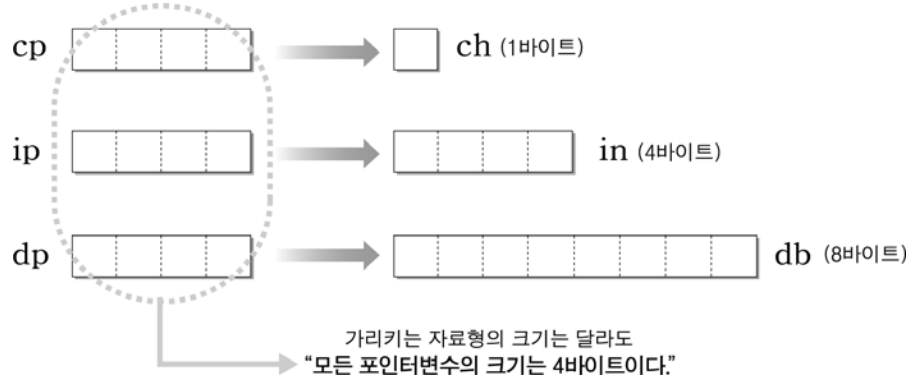
- 포인터와 포인터변수의 크기는 몇 바이트인가?

  - 컴파일러마다 다를 수 있으므로 sizeof연산자로 계산한다.

```
int in;
int *ip = &in;
printf("int형 변수의 포인터의 크기 : %d₩n", sizeof(&in));
printf("int형 변수를 가리키는 포인터변수의 크기 : %d₩n", sizeof(ip));
```

  - 포인터와 포인터변수의 크기는 가리키는 자료형과는 관계없이 항상 같다.

```
char ch;
int in;
double db;
char *cp = &ch;
int *ip = &in;
double *dp = &db;
printf("%d, %d, %d₩n",
    sizeof(cp), sizeof(ip), sizeof(dp));        // 결과는 4, 4, 4
printf("%d, %d, %d₩n",
    sizeof((*cp), sizeof(*ip), sizeof(*dp));    // 결과는 1, 4, 8
```



❑ 포인터에 관한 중요한 질문

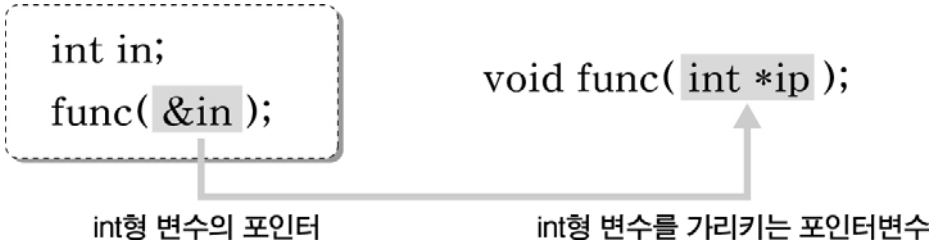
- 포인터와 포인터변수는 자동 형변환이 가능한가?

  – 기본자료형은 자동형변환이 가능하나 포인터(변수)는 불가능하다.

  (형변환 연산자를 사용한 명시적 형변환은 가능하다.)

```
int *ip;            // int형 변수를 가리키는 포인터변수
double db=6.5;      // double형 변수
ip=&db;             // double형 변수의 포인터를 int형 포인터변수에 대입
```

컴파일 에러 ⇨ error C2440: '=' : cannot convert from 'double *' to 'int *'

  – 자동 형변환이 가능하다면 정수값이 저장된 기억공간에서 실수값을 참조하는 오류를 범하게 될 것이다.

  – 포인터의 형변환 규칙은 함수의 전달인자와 매개변수 사이에도 적용된다.