-Part2-

제3장 포인터란 무엇인가

학습목차

3.1 포인터란

'포인터'를 공부하기 전에...

▶ **택배 아저씨가 하는 일**

✓ 고객의 **주소를 저장**하고 있다가 해당 **주소로 물건을 전달**하는 일

✓ 고객의 **주소를 저장**하고 있다가 해당 **주소로 물건을 받아**가는 일

✓ '우리에게 **'간접 접근'** 서비스를 제공한다.'

▶ **택배 아저씨가 있어서 우리가 편리한 점**

✓ '물품을 수령하기 위해 직접 구매처를 방문 하지 않아도 된다.'

✓ '반품을 위해 직접 구매처를 방문 하지 않아도 된다.'

▶ **컴퓨팅 세계에서 택배 아저씨와 같은 일을 하는 변수**

✓ **포인터 변수** (포인터라고도 부름)

- 메모리의 **주소를 저장**하고 있다가 해당 **주소로 데이터를 전달** 하는 일
- 메모리의 **주소를 저장**하고 있다가 해당 **주소로 데이터를 참조** 하는 일

3.1 포인터란

► '포인터'란?

✓ '주소를 저장하는 변수이다.'

✓ 'C언어의 장점 중에 하나가 바로 포인터(포인터 변수)이다.'

► '포인터'를 사용하면 어떤 장점이 있는가?

✓ 메모리 주소를 참조해서 다양한 자료형 변수들의 접근과 조작 용이

- 현재 '장'에서 배울 예정

✓ 메모리 주소를 참조하여 배열과 같은 연속된 데이터에 접근과 조작 용이

- PART2-4장에서 배움

✓ 동적 할당된 메모리 영역(힙영역)에 접근과 조작 용이

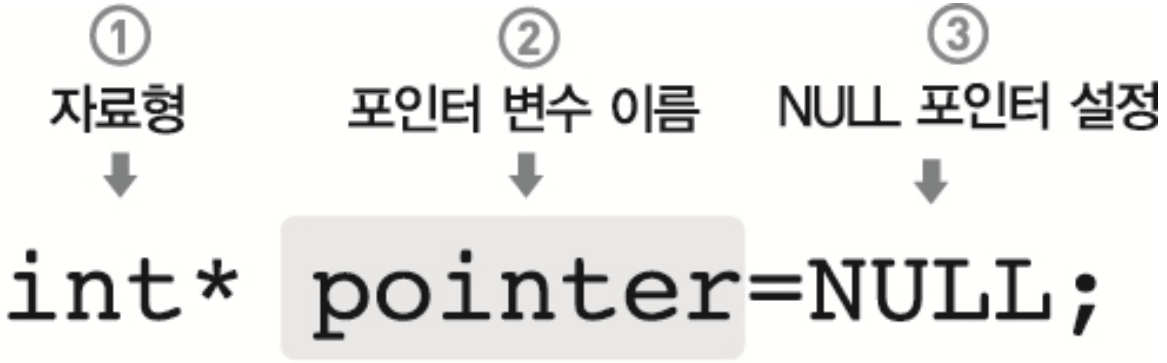
- PART3-4장에서 배움

3.2 포인터 변수의 선언과 사용

3.2 포인터 변수의 선언과 사용 (1/8)

▶ 포인터 변수의 선언

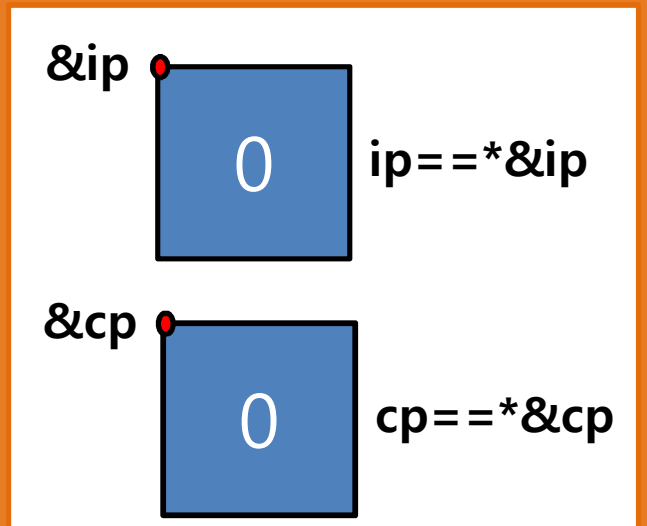
- ✓ **자료형:** 포인터 변수의 자료형 지정, 자료형 다음에 * 연산자를 붙임
- ✓ **포인터 변수 이름:** 주소를 저장할 변수의 이름 지정
- ✓ **NULL 포인터 설정:** 포인터 변수 선언 시 NULL로 초기화




```
int* p1=NULL;       // int형 주소를 저장하는 포인터 변수
char* p2=NULL;      // char형 주소를 저장하는 포인터 변수
double* p3=NULL;    // double형 주소를 저장하는 포인터 변수
```

3.2 포인터 변수의 선언과 사용 (2/8)---[3-2.c 실습]

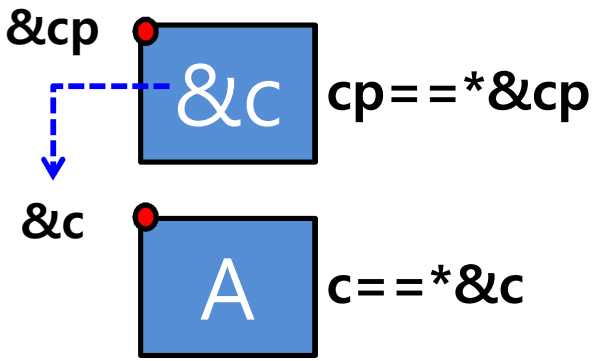
```
#include <stdio.h>
int main(void)
{
   // 포인터 변수 선언
   char* cp=NULL;
     int*  ip=NULL;

   printf("%x %x %x₩n", &cp, cp, *&cp);
   printf("%x %x %x₩n", &ip, ip, *&ip);

   printf("%d %d ₩n", sizeof(char*), sizeof(int*));  // 4, 4 출력
   printf("%d %d ₩n", sizeof(cp), sizeof(ip));       // 4, 4 출력
   return 0;
}
```



'모든 포인터 변수는 4바이트 이다.'

3.2 포인터 변수의 선언과 사용 (3/8)---[3-3.c 실습]

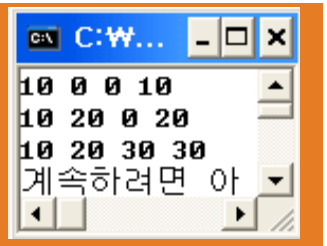
```
#include <stdio.h>
int main( )
{
    char c='A';
    char* cp=NULL;

    cp=&c;  // 주소 저장

    printf("%x %c %c ₩n", &c, c, *&c);
    printf("%x %x %x ₩n", &cp, cp, *&cp);

    printf("%c ₩n", c);      // 직접 접근
    printf("%c ₩n", *cp);  // 간접 접근
    return 0;
}
```

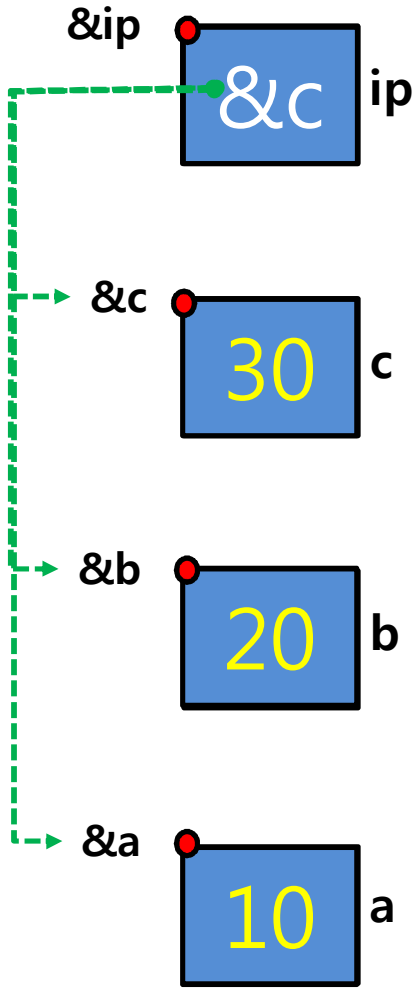


같은 메모리 공간의 이름
c == *&c == *cp

3.2 포인터 변수의 선언과 사용 (4/8)---[3-4.c 실습]

```c
#include <stdio.h>
int main( )
{
    int a=0, b=0, c=0;
    int* ip=NULL;     // 포인터 변수 선언

    ip=&a;            // 주소 저장
    *ip=10;
    printf("%d %d %d %d₩n", a, b, c, *ip);

    ip=&b;            // 주소 저장 변경
    *ip=20;
    printf("%d %d %d %d₩n", a, b, c, *ip);

    ip=&c;            // 주소 저장 변경
    *ip=30;
    printf("%d %d %d %d₩n", a, b, c, *ip);
    return 0;
}
```

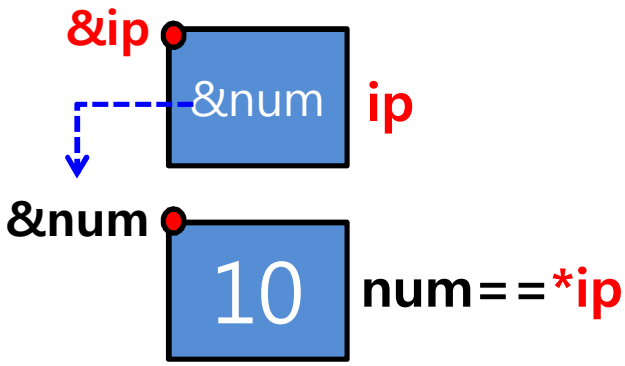


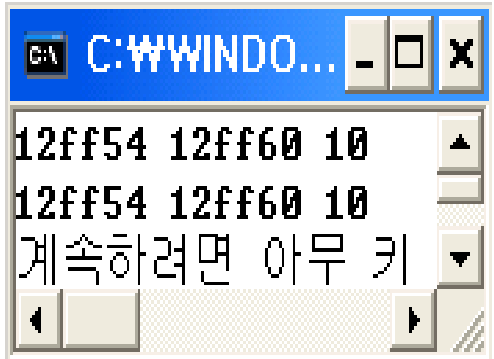


3.2 포인터 변수의 선언과 사용 (5/8)---[3-5.c 실습]

```
#include <stdio.h>
int main( )
{
    int num=10;
    int* ip=NULL;  // 포인터 변수 선언

    ip=&num;       // 주소 저장

    printf("%x %x %d ₩n", &*&ip, *&ip, **&ip);
    printf("%x %x %d ₩n", &ip, ip, *ip);
    return 0;
}
```

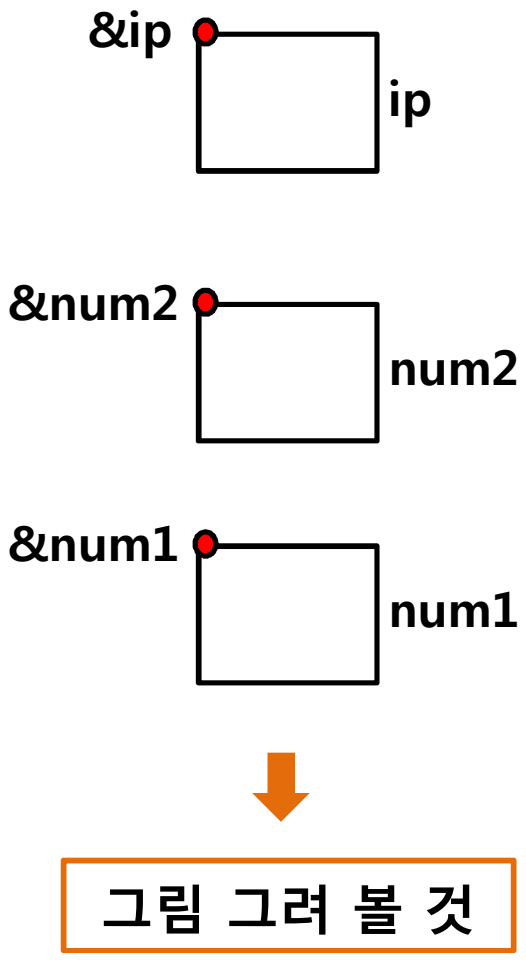
*&는 서로 상쇄





3.2 포인터 변수의 선언과 사용 (6/8)---[3-6.c 실습]

```
#include <stdio.h>
int main( )
{
    int num1=10;
    int num2=0;
    int* ip=NULL;// 포인터 변수 선언

    ip=&num1;   // 주소 저장

    num2=*ip+num1;
    printf("%d %d %d₩n", *ip, num1, num2);
    return 0;
}
```



3.2 포인터 변수의 선언과 사용 (7/8)

▶ 잘못 사용된 포인터

① 포인터 변수에 주소를 저장하지 않은 경우

```
#include <stdio.h>
int main(void)
{
    int* ip=NULL;
    *ip=10000;
    return 0;
}
```

② 포인터 변수에 이상한 주소 저장

```
#include <stdio.h>
int main(void)
{
    int* ip=14592343;
    *ip=1020;
    return 0;
}
```

3.2 포인터 변수의 선언과 사용 (8/8)

► 포인터 변수의 초기화 방법 2 가지

```
#include <stdio.h>
int main(void)
{
    int num=10;
    int* ip=NULL;
    ip=&num;
    return 0;
}
```

포인터 변수의 선언과 초기화를 개별적으로 수행

같은 표현

==

```
#include <stdio.h>
int main(void)
{
    int num=10;
    int* ip=&num;
    return 0;
}
```

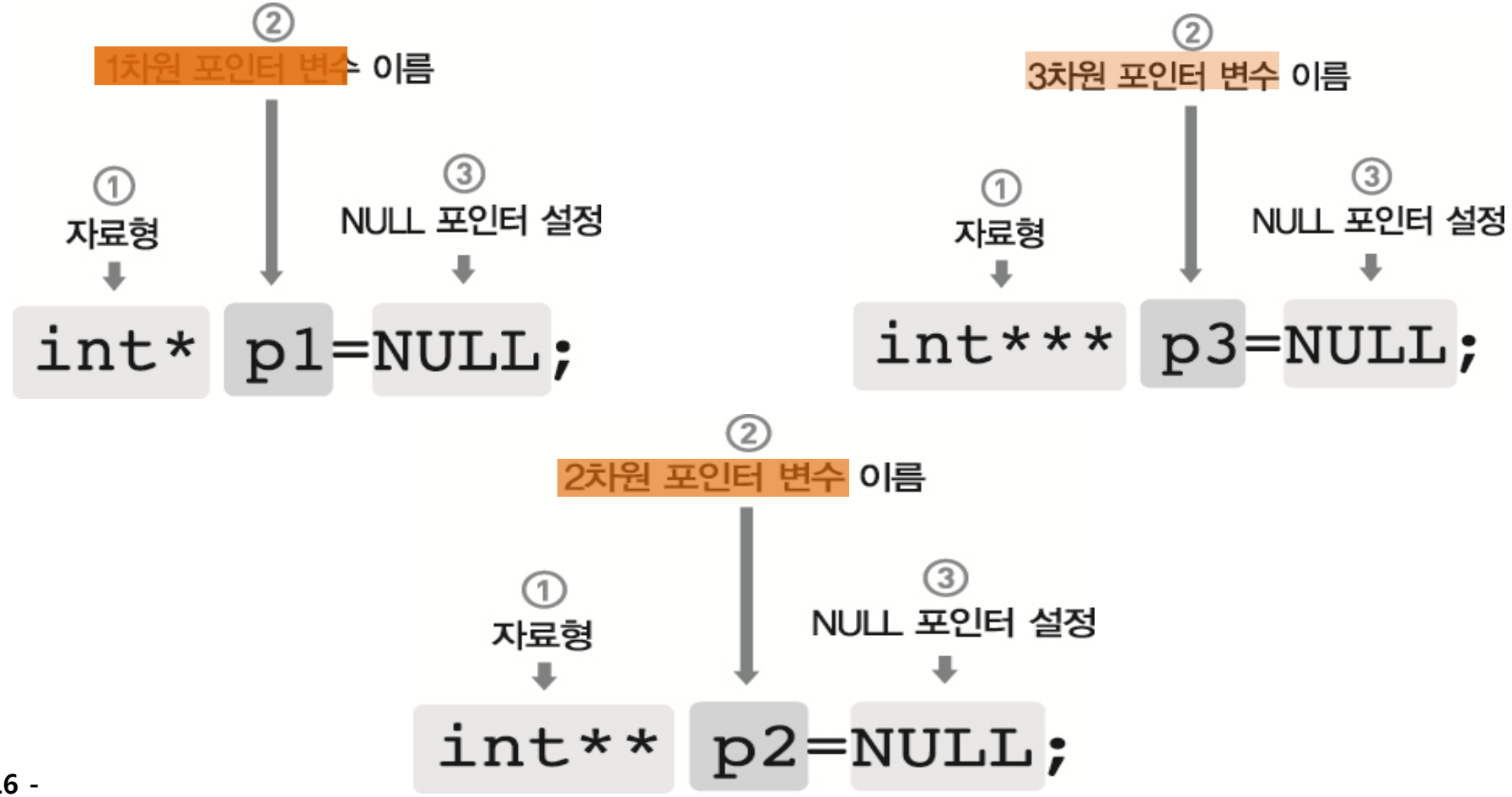
포인터 변수의 선언과 초기화를 동시에 수행

3.3 다차원 포인터 변수의 선언과 사용

3.3 다차원 포인터 변수의 선언과 사용 (1/11)

▶ 다차원 포인터 변수란?

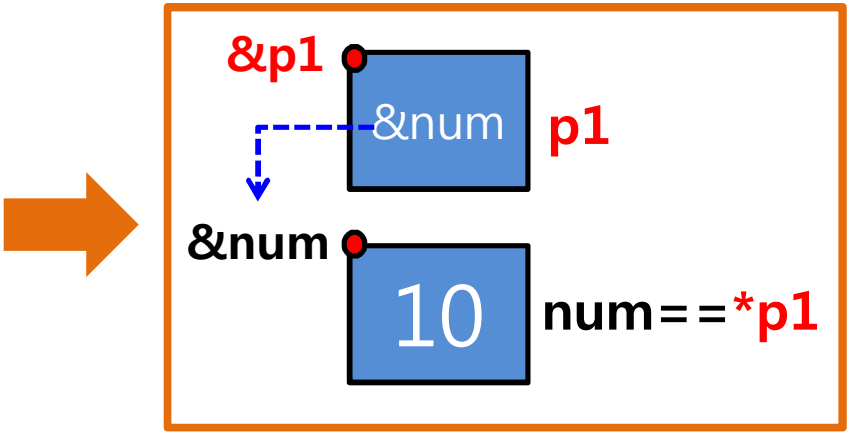
✓ '2차원 이상의 포인터 변수를 의미한다.'



3.3 다차원 포인터 변수의 선언과 사용 (2/11)

▶ 1차원 포인터 변수의 역할: 일반 변수의 주소를 저장

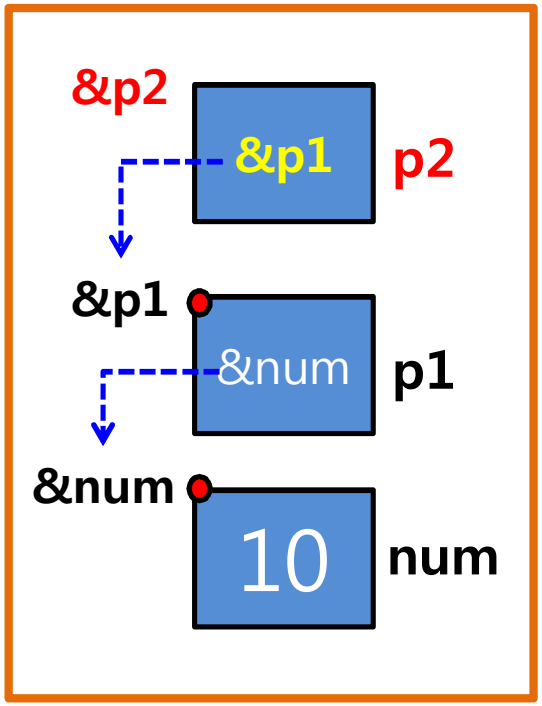
```
#include <stdio.h>
int main(void)
{
    int num=10;
    int* p1=NULL;
    p1=&num;
    return 0;
}
```



3.3 다차원 포인터 변수의 선언과 사용 (3/11)

▶ 2차원 포인터 변수의 역할: 1차원 포인터 변수의 주소를 저장

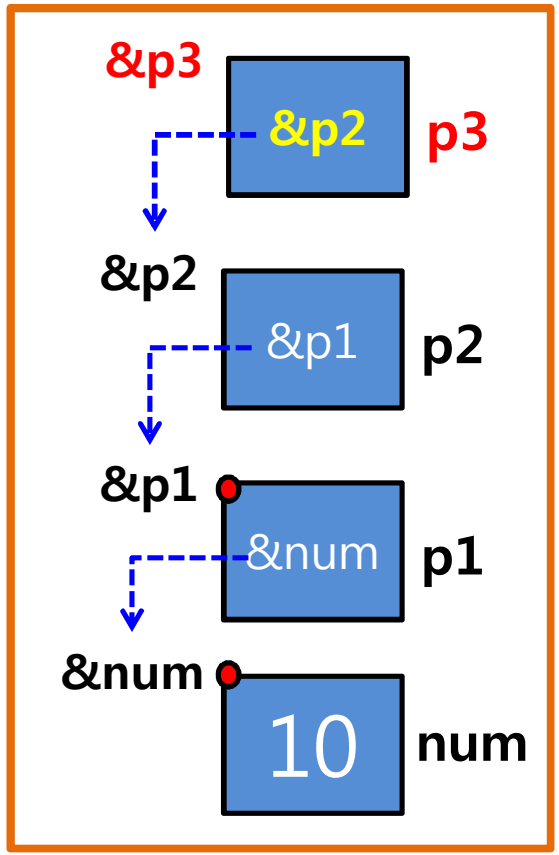
```
#include <stdio.h>
int main(void)
{
    int num=10;
    int*  p1=NULL;
    int** p2=NULL;

    p1=&num;
    p2=&p1;
    return 0;
}
```



3.3 다차원 포인터 변수의 선언과 사용 (4/11)

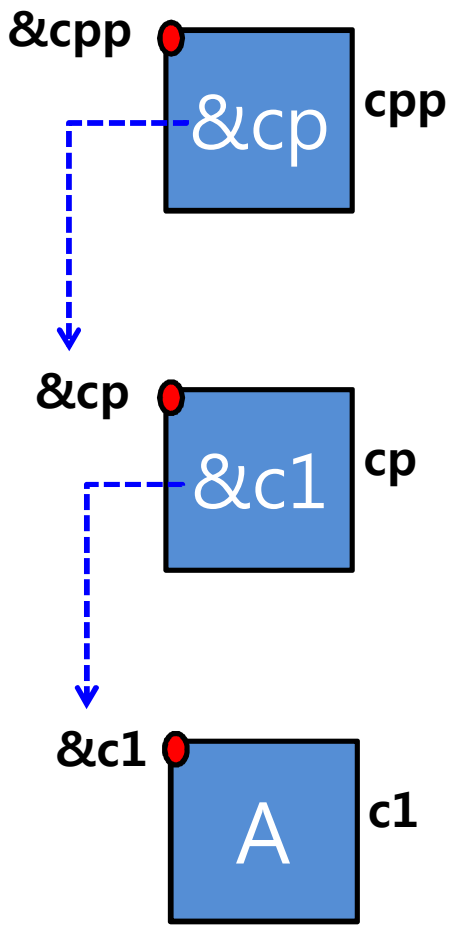
▶ 3차원 포인터 변수의 역할: 2차원 포인터 변수의 주소를 저장

```
#include <stdio.h>
int main(void)
{
    int   num=10;
    int*    p1=NULL;
    int**   p2=NULL;
    int*** p3=NULL;

    p1=&num;
    p2=&p1;
    p3=&p2;
    return 0;
}
```



3.3 다차원 포인터 변수의 선언과 사용 (5/11)---[3-7.c 실습]

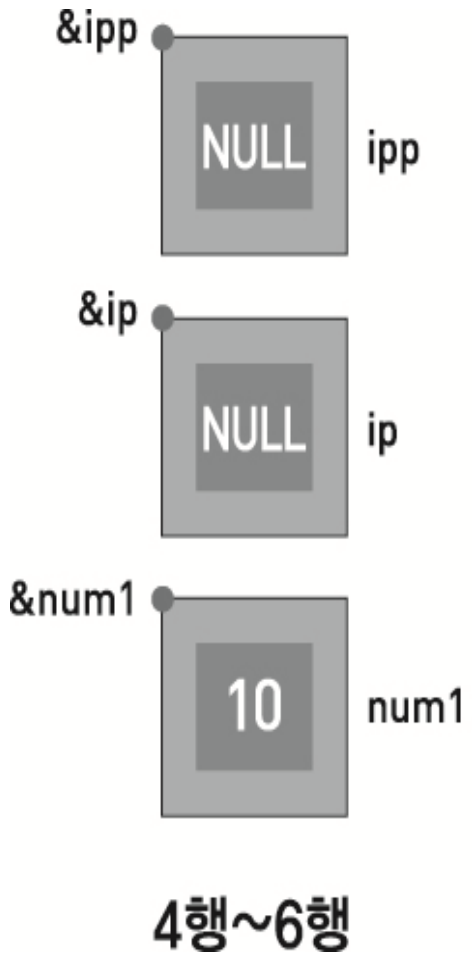
```
#include <stdio.h>
int main( )
{
    char c1='A';
    char* cp=NULL;
    char** cpp=NULL;

    cp=&c1;
    cpp=&cp;

    printf("%c %x %x ₩n", c1, cp, cpp);
    printf("%x %x %x ₩n", &c1, &cp, &cpp);
    printf("%c %c %c ₩n", c1, *cp,**cpp);

    return 0;
}
```



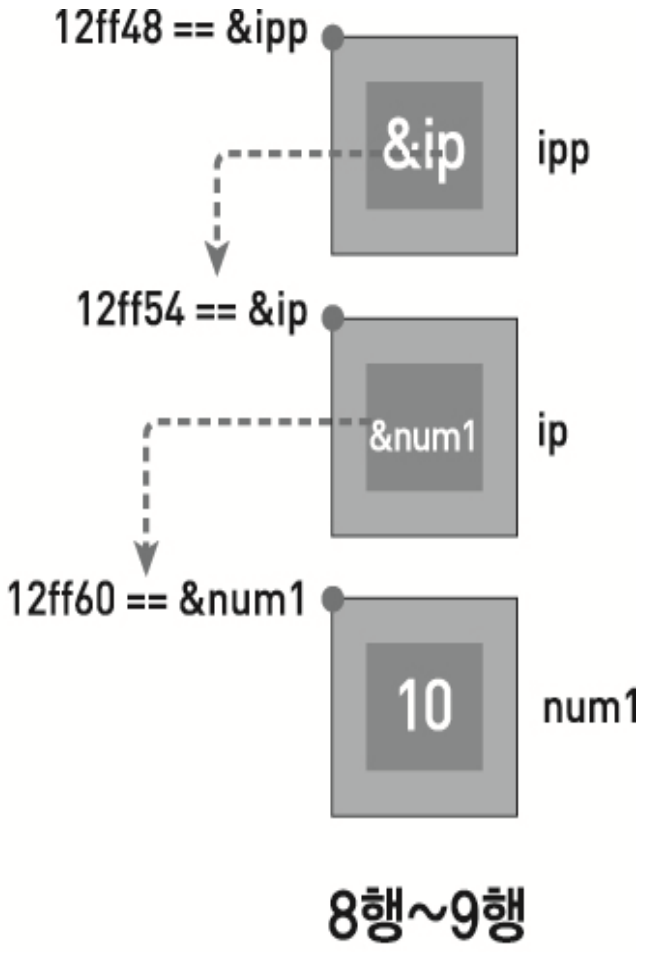
3.3 다차원 포인터 변수의 선언과 사용 (6/11)---[3-8.c 실습]

```
#include <stdio.h>
int main( )
{
    int num1=10;
    int* ip=NULL;
    int** ipp=NULL;

    ip=&num1;
    ipp=&ip;

    printf("%d %x %x ₩n", num1, ip, ipp);
    printf("%x %x %x ₩n", &num1, &ip, &ipp);
    printf("%d %x %x ₩n", *&num1, *&ip, *&ipp);

    printf("%d %d %d ₩n", num1, *ip, **ipp);
    printf("%x %x %x ₩n", &num1, ip, *ipp);
    return 0;
}
```

3.3 다차원 포인터 변수의 선언과 사용 (7/11)---[3-8.c 분석]



4행~6행

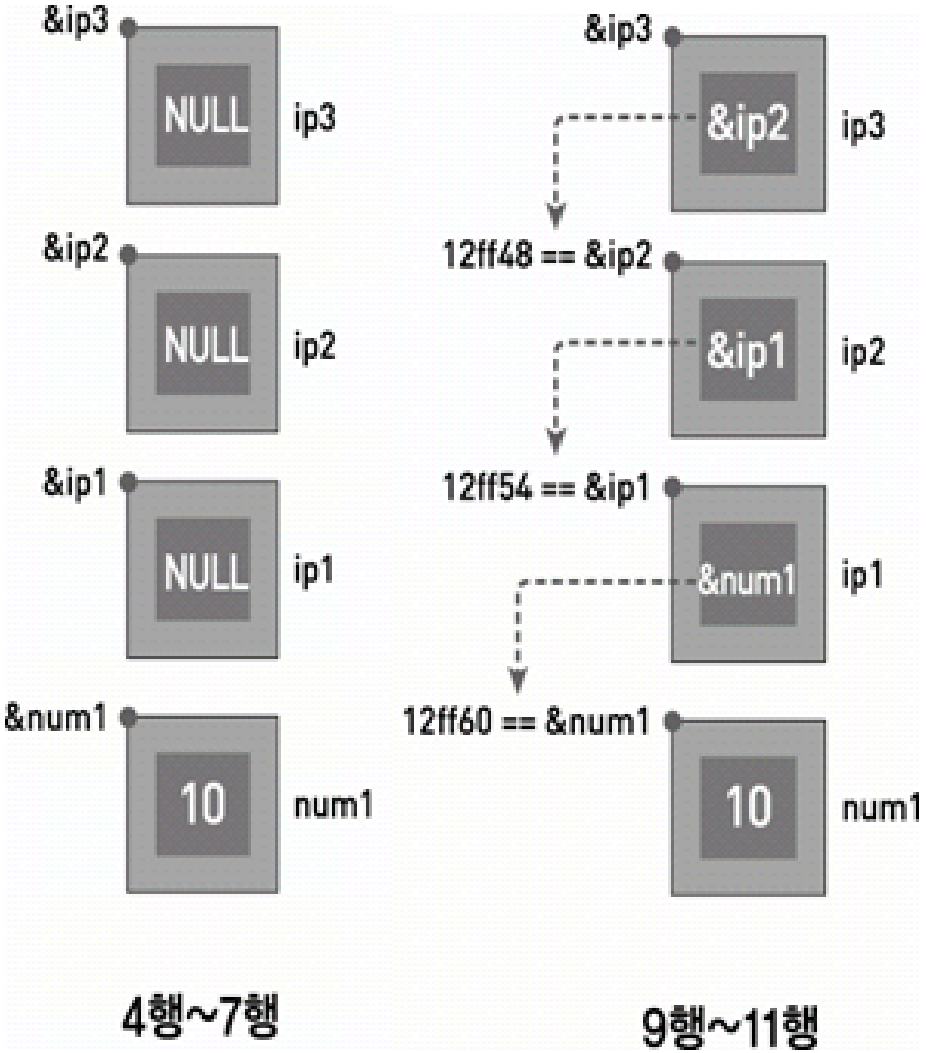


8행~9행

3.3 다차원 포인터 변수의 선언과 사용 (8/11)---[3-9.c 실습]

```
int num1=10;
int* ip1=NULL;
int** ip2=NULL;
int*** ip3=NULL;

ip1=&num1;
ip2=&ip1;
ip3=&ip2;
printf("%d %d %d %d ₩n", num1, *ip1, **ip2, ***ip3);
printf("%x %x %x %x ₩n", &num1, ip1, *ip2, **ip3);
printf("%x %x %x ₩n", &ip1, ip2, *ip3);
printf("%x %x ₩n", &ip2, ip3);

printf("%d %d ₩n", sizeof(int), sizeof(int*));
printf("%d %d₩n", sizeof(int**), sizeof(int***));

printf("%d %d ₩n", sizeof(num1), sizeof(ip1));
printf("%d %d₩n", sizeof(ip2), sizeof(ip3));
```

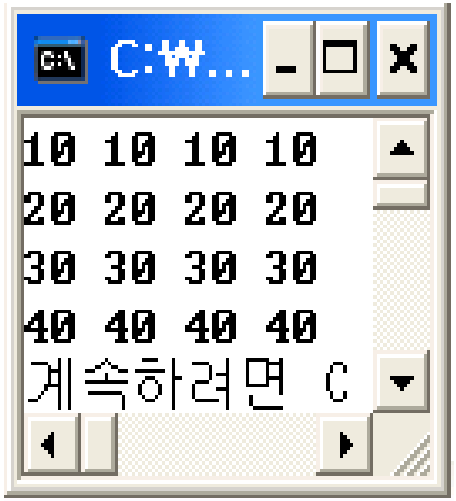
3.3 다차원 포인터 변수의 선언과 사용 (9/11)---[3-9.c 분석]



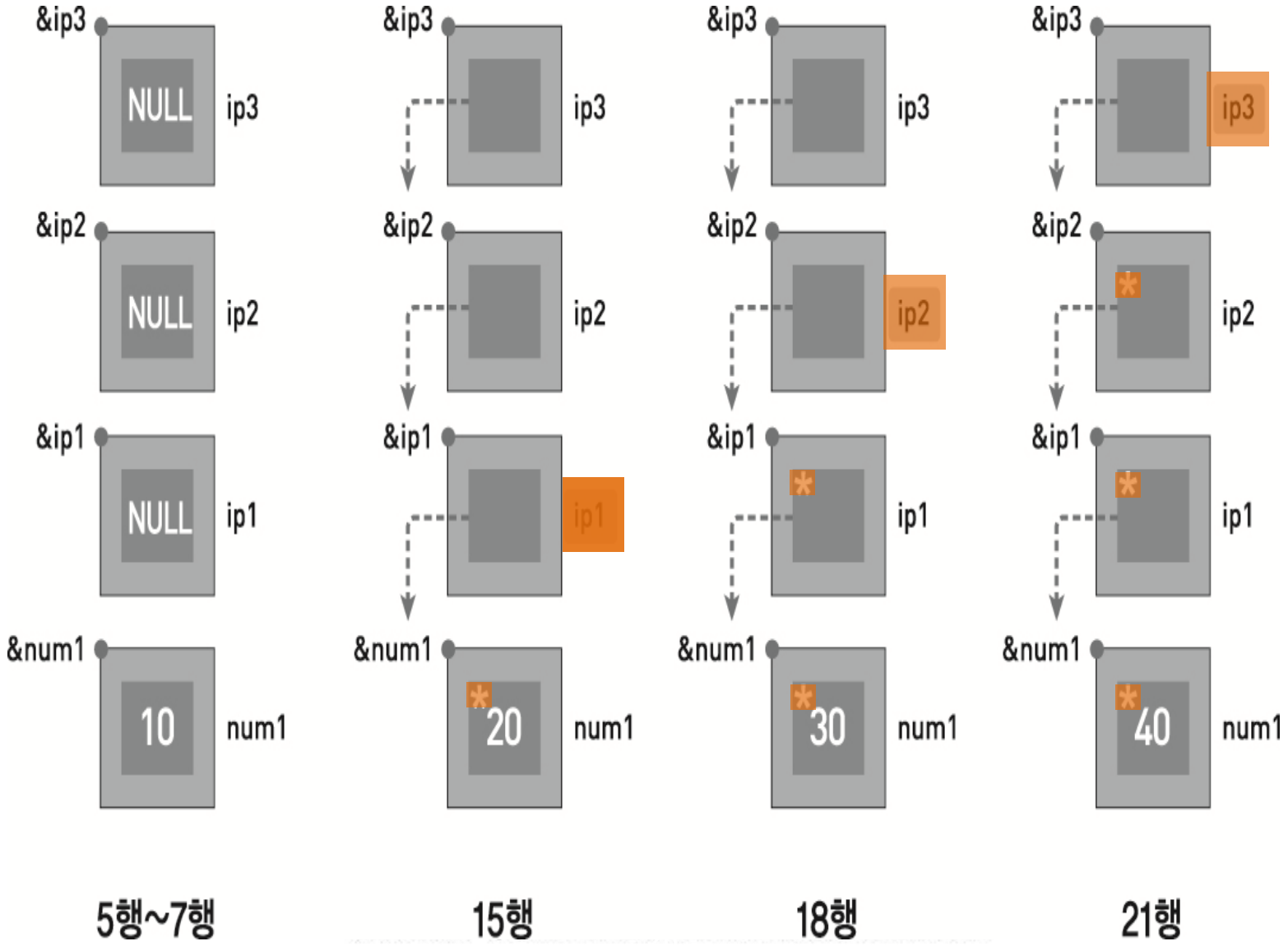
ip3==&ip2
*ip3==ip2==&ip1
**ip3==*ip2==ip1==&num1
***ip3==**ip2==*ip1==num1==10

3.3 다차원 포인터 변수의 선언과 사용 (10/11)---[3-10.c 실습]

```
int num1=10;
int* ip1=NULL;
int** ip2=NULL;
int*** ip3=NULL;

ip1=&num1;
ip2=&ip1;
ip3=&ip2;

printf("%d %d %d %d ₩n", num1, *ip1, **ip2, ***ip3);

*ip1=20;
printf("%d %d %d %d ₩n", num1, *ip1, **ip2, ***ip3);

**ip2=30;
printf("%d %d %d %d ₩n", num1, *ip1, **ip2, ***ip3);

***ip3=40;
printf("%d %d %d %d ₩n", num1, *ip1, **ip2, ***ip3);
```



3.3 다차원 포인터 변수의 선언과 사용 (11/11)---[3-10.c 분석]



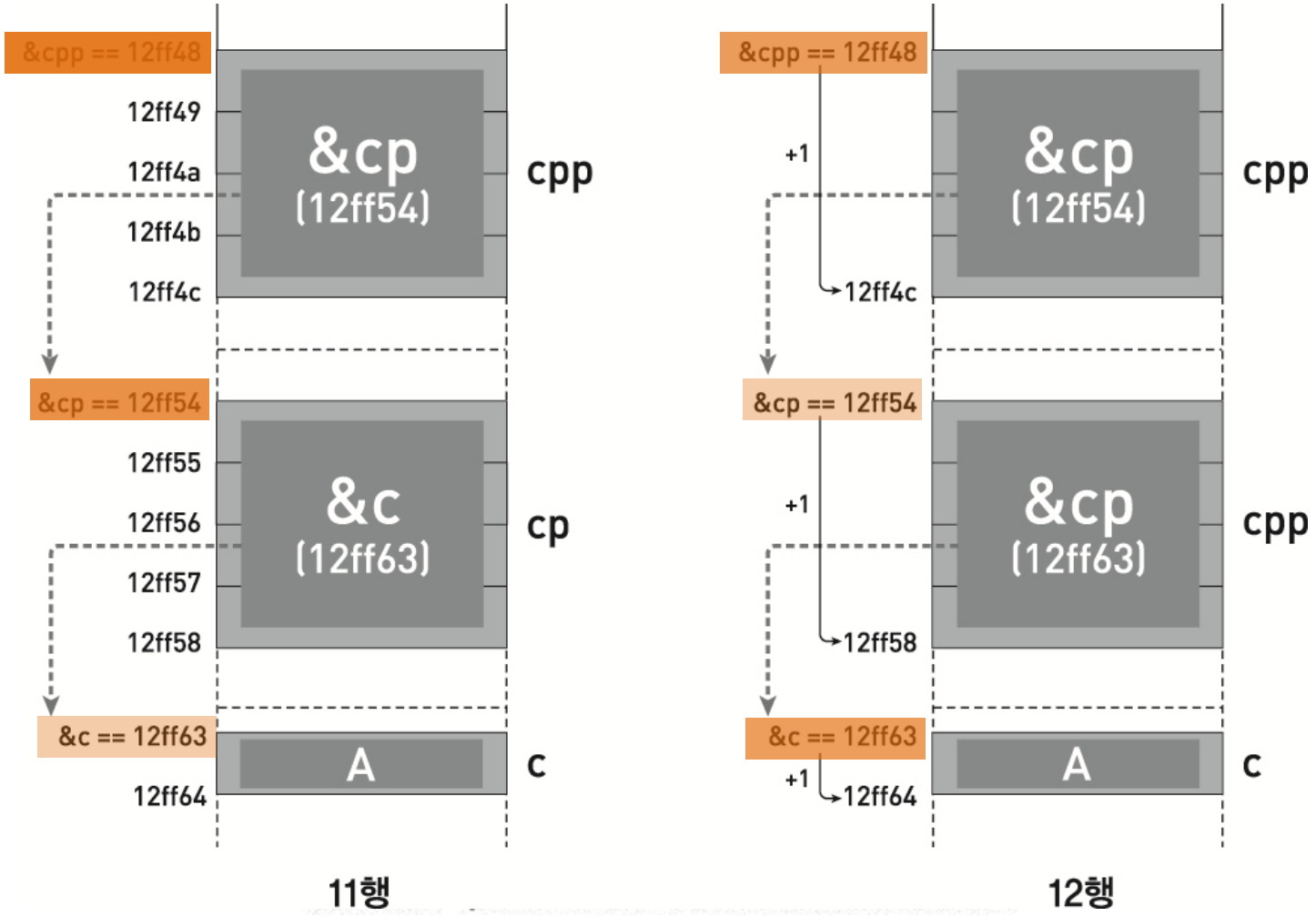
3.4 주소의 가감산

3.4 주소의 가감산 (1/8)---[3-11.c 실습]

```
#include <stdio.h>
int main( )
{
    char c='A';
    char* cp=NULL;
    char** cpp=NULL;

    cp=&c;
    cpp=&cp;

    printf("%x %x %x ₩n", &c, &cp, &cpp);
    printf("%x %x %x ₩n", &c+1, &cp+1, &cpp+1);

    printf("%c %x %x ₩n", c, cp, cpp);
    printf("%c %x %x ₩n", c+1, cp+1, cpp+1);
    return 0;
}
```

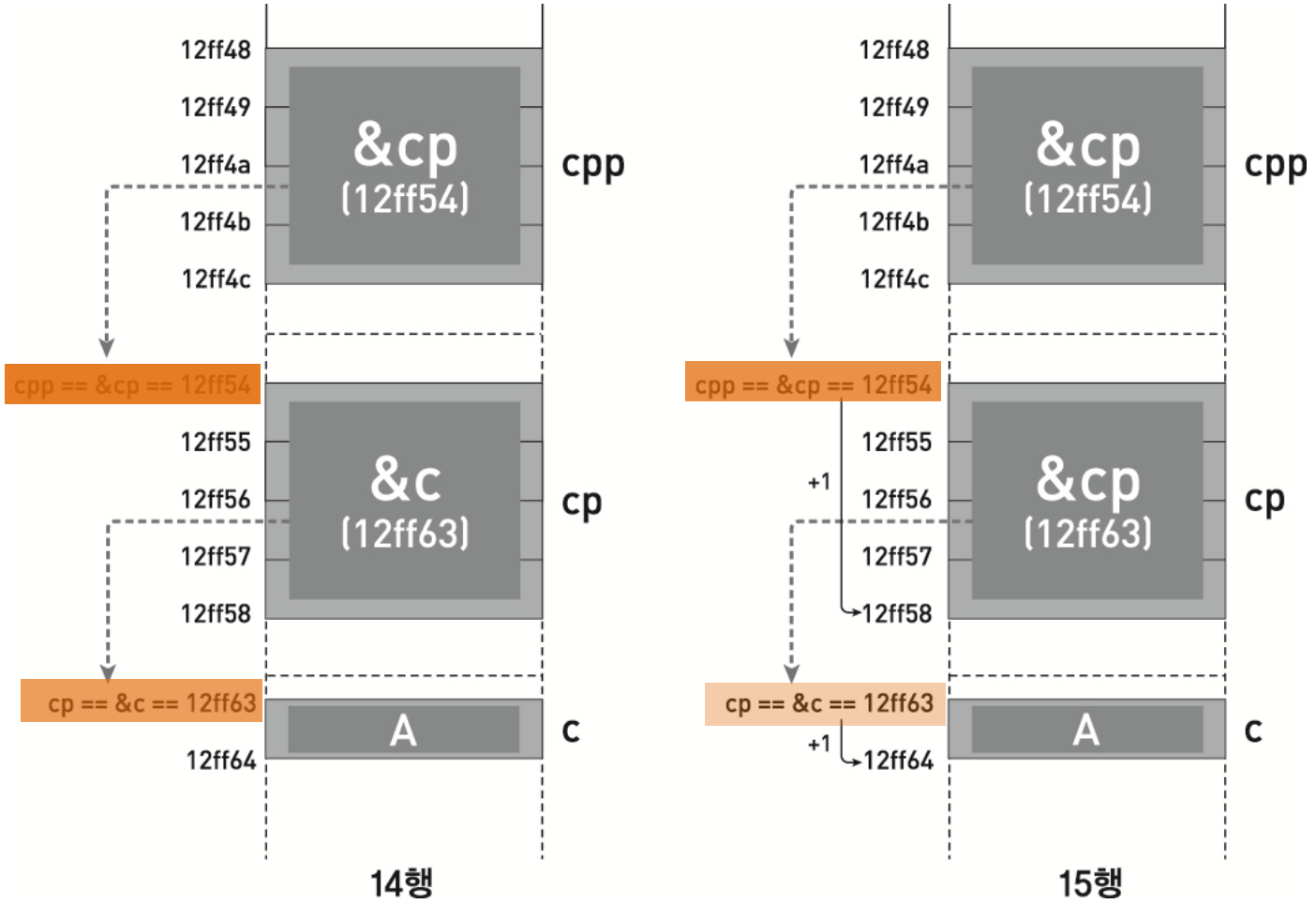
3.4 주소의 가감산 (2/8)---[3-11.c 분석]

```
printf("%x %x %x ₩n", &c, &cp, &cpp);
printf("%x %x %x ₩n", &c+1, &cp+1, &cpp+1);
```



3.4 주소의 가감산 (3/8)---[3-11.c 분석]

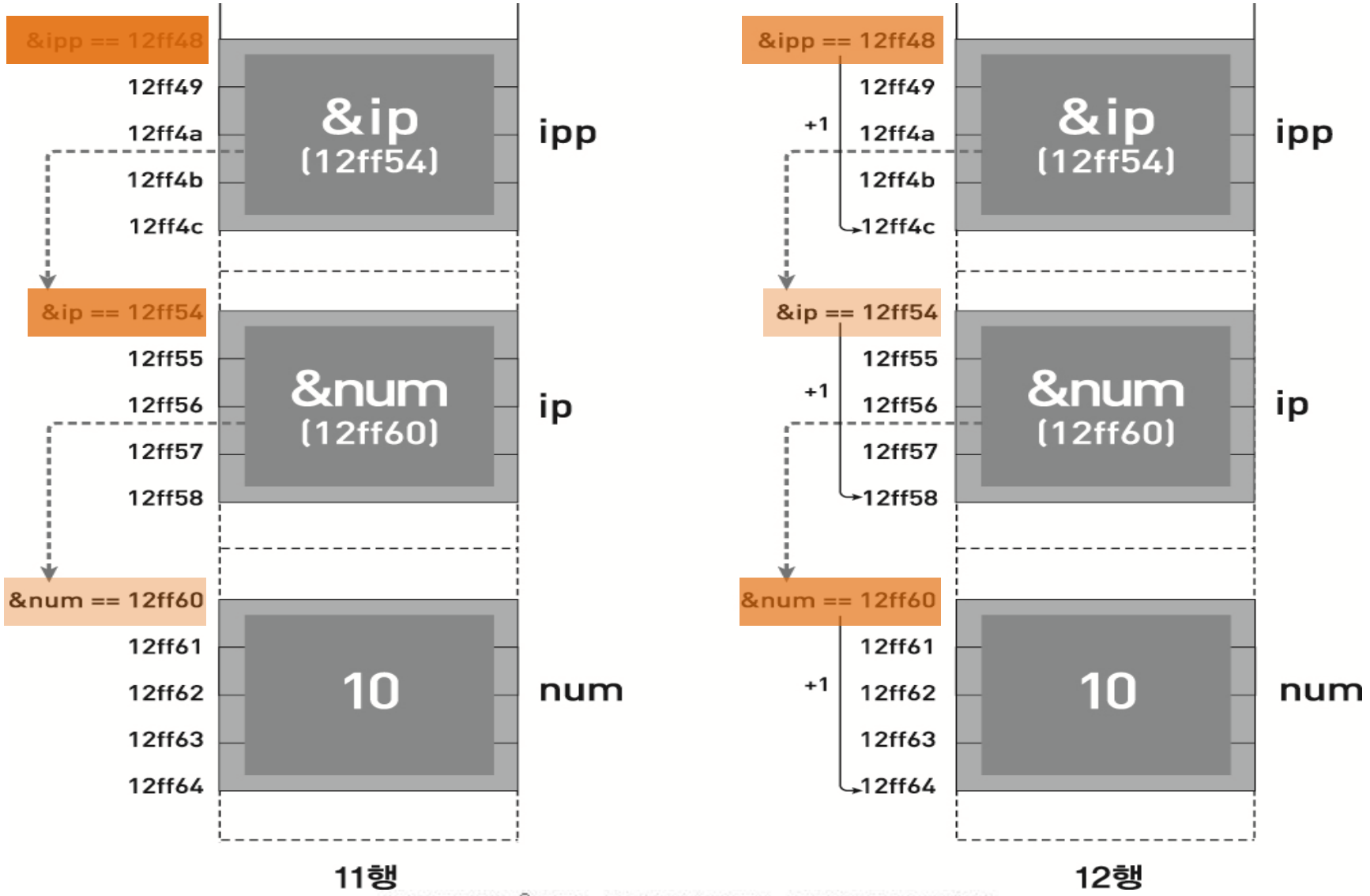
```
printf("%c %x %x ₩n", c, cp, cpp);
printf("%c %x %x ₩n", c+1, cp+1, cpp+1);
```



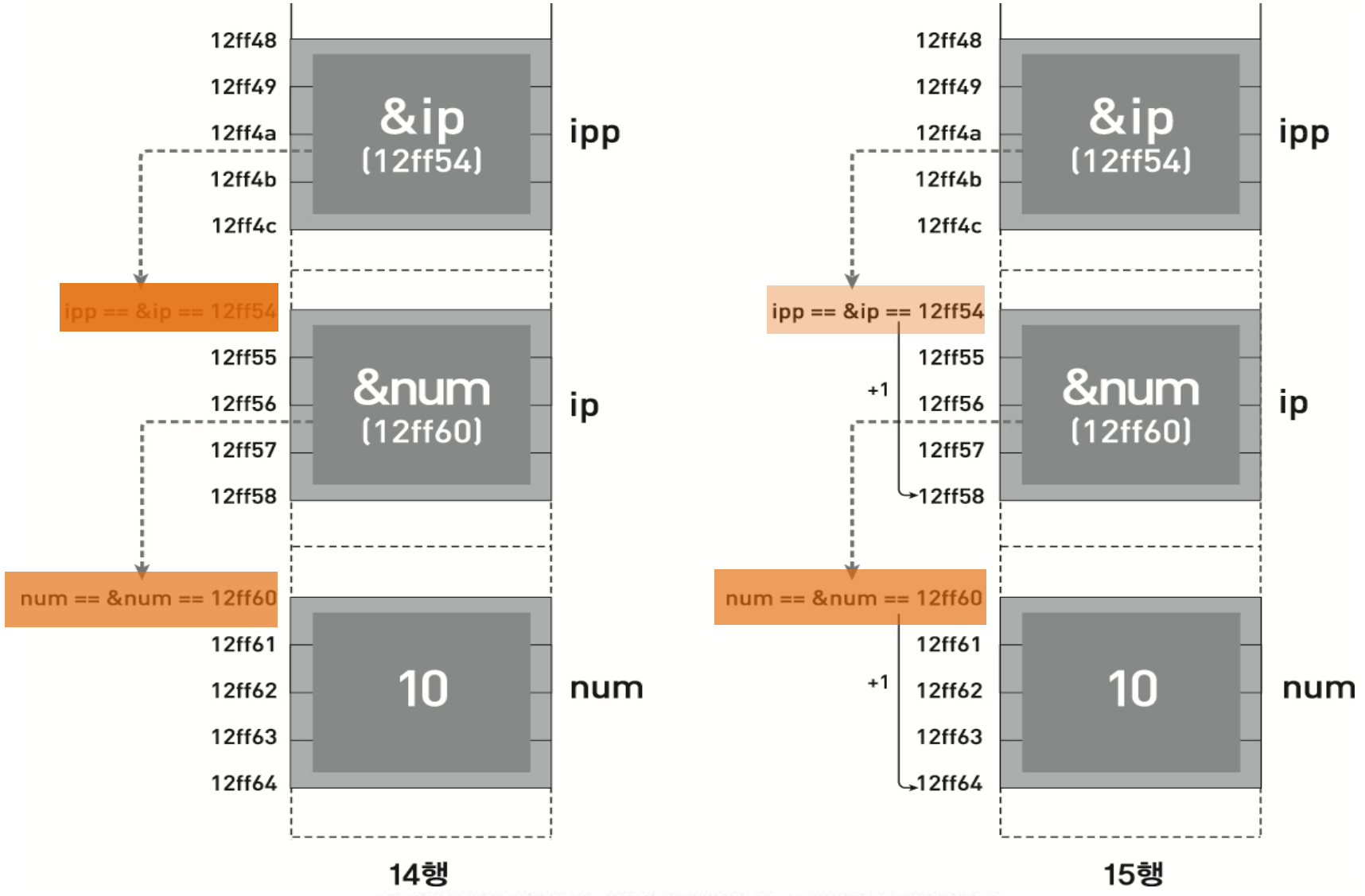
3.4 주소의 가감산 (4/8)---[3-12.c 실습]

```
#include <stdio.h>
int main( )
{
    int num=10;
    int* ip=NULL;
    int** ipp=NULL;

    ip=&num;
    ipp=&ip;

    printf("%x %x %x ₩n", &num, &ip, &ipp);
    printf("%x %x %x ₩n", &num+1, &ip+1, &ipp+1);

    printf("%d %x %x ₩n", num, ip, ipp);
    printf("%d %x %x ₩n", num+1, ip+1, ipp+1);
    return 0;
}
```

3.4 주소의 가감산 (5/8)---[3-12.c 분석]



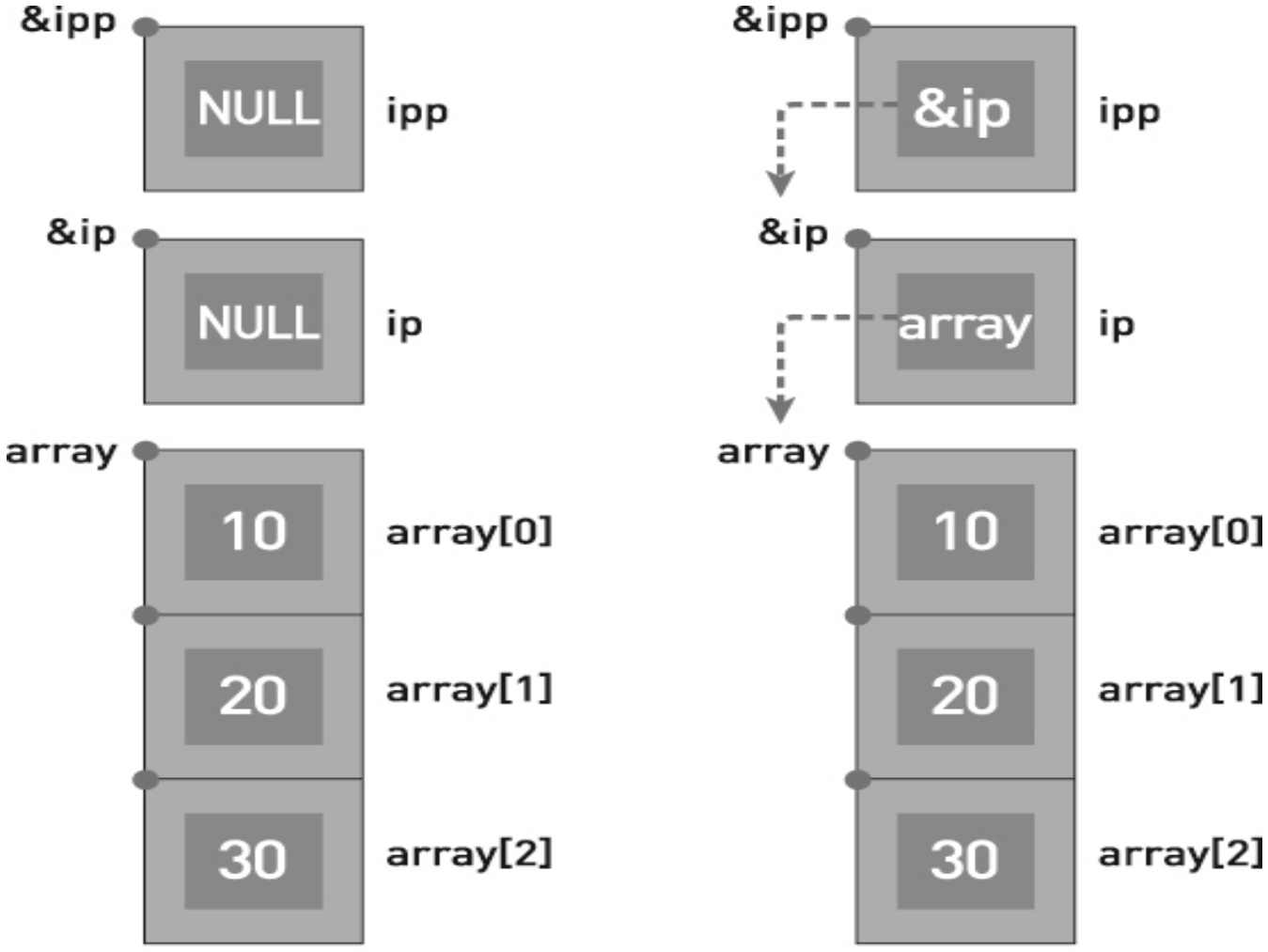
3.4 주소의 가감산 (6/8)---[3-12.c 분석]



3.4 주소의 가감산 (7/8)---[3-13.c 실습]

```
#include <stdio.h>
int main( )
{
    int array[3]={10,20,30};
    int* ip=NULL;
    int** ipp=NULL;

    ip=array;
    ipp=&ip;

    printf("%d %d %d ₩n", array[0], array[1], array[2]);
    printf("%d %d %d ₩n", *(ip+0), *(ip+1), *(ip+2));
    printf("%d %d %d ₩n", *(*ipp+0), *(*ipp+1), *(*ipp+2));

    return 0;
}
```

3.4 주소의 가감산 (8/8)---[3-13.c 분석]



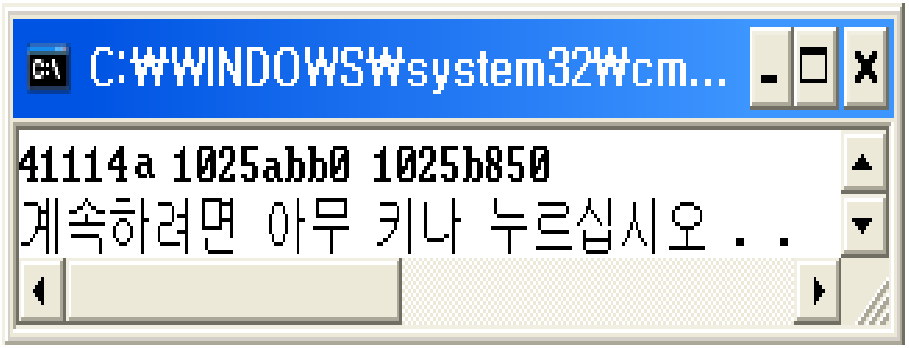
4행~6행 8행~9행

3.5 함수 포인터

3.5 함수 포인터 (1/5)---[3-14.c 실습]

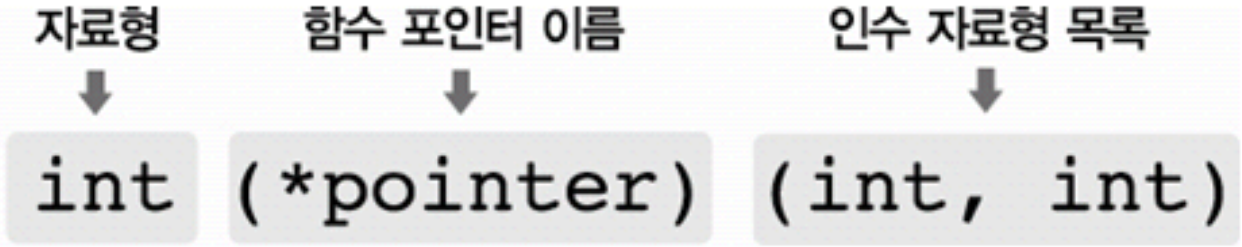
▶ 함수 이름은 '함수의 시작 주소'

```
#include <stdio.h>
int main(void)
{
    printf("%x %x %x ₩n", main, printf, scanf);
    return 0;
}
```



3.5 함수 포인터 (2/5)

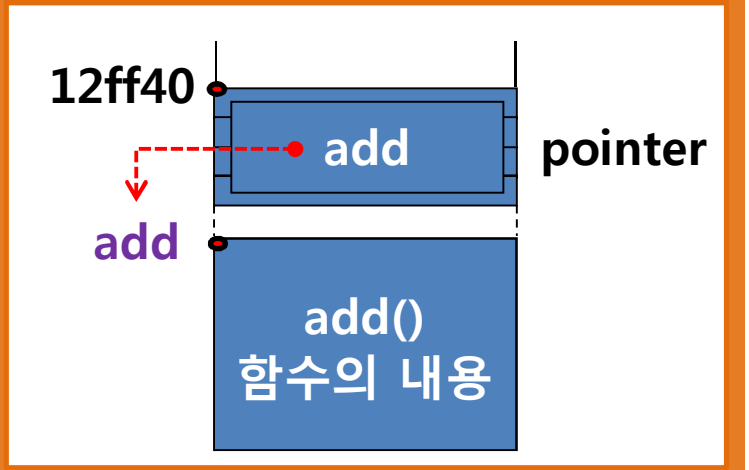
▶ **함수 포인터:** 함수의 시작 주소를 저장하는 변수



- ✓ **자료형:** 가리키는 대상이 되는 함수의 자료형을 설정
- ✓ **함수 포인터 이름:** 괄호와 *을 반드시 사용
- ✓ **인수 자료형 목록:** 가리키는 대상이 되는 함수의 인수들의 자료형 목록

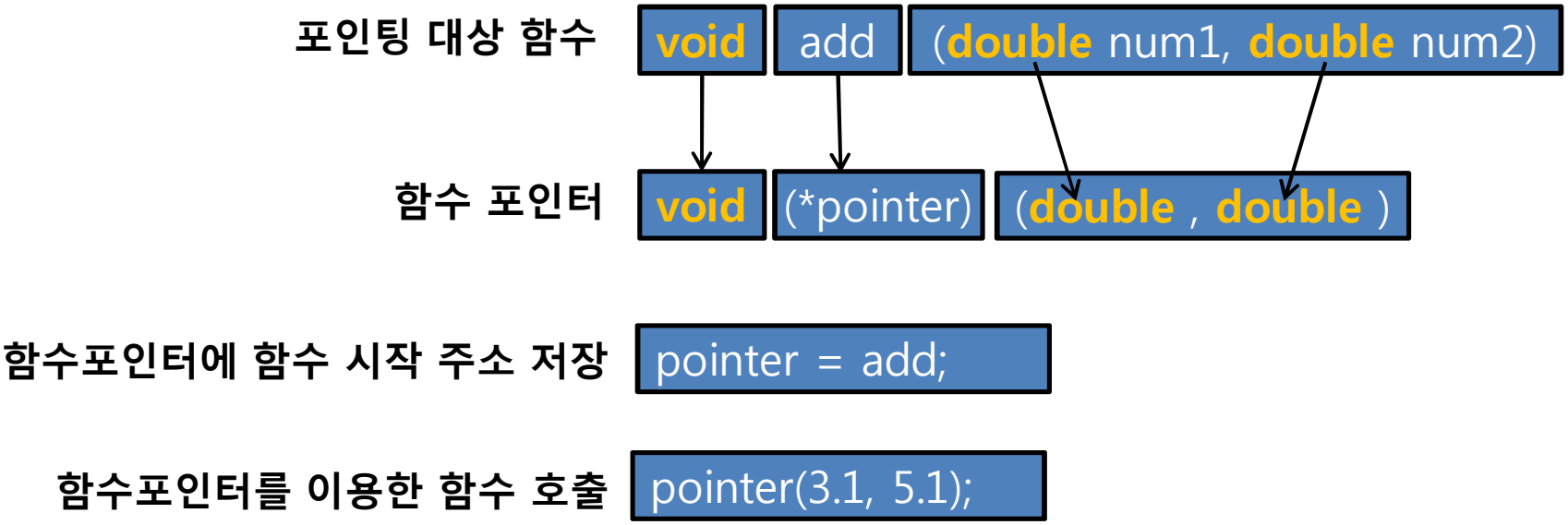
3.5 함수 포인터 (2/5)---[3-15.c 실습]

```
#include <stdio.h>
void add(double num1, double num2);
int main( )
{
   double x=3.1, y=5.1;
   void (*pointer) (double, double);  // 함수 포인터 선언

   printf("add 함수의 주소 : %x₩n", add);
   printf("함수 포인터의 주소 : %x ₩n", &pointer);

   pointer=add;
   pointer(x, y);   // 함수 포인터를 이용한 호출

   return 0;
}
```




```
void add(double num1, double num2)
{
   double result;
   result=num1+num2;
   printf("%lf + %lf = %lf입니다.₩n", num1, num2, result);
}
```

3.5 함수 포인터 (3/5)---[3-15.c 분석]

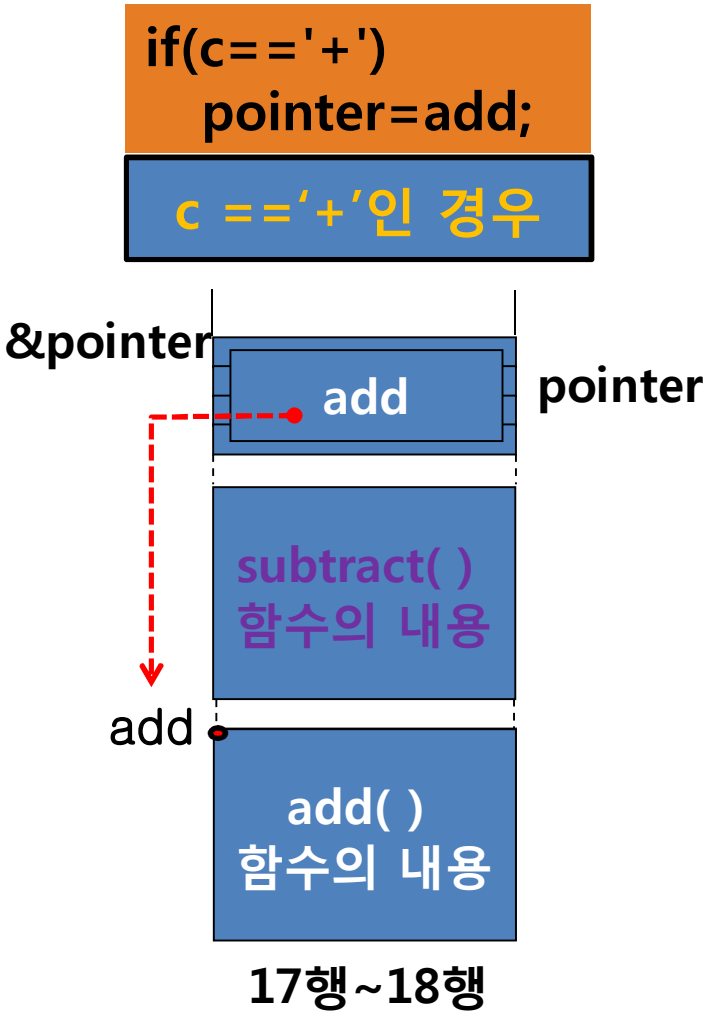


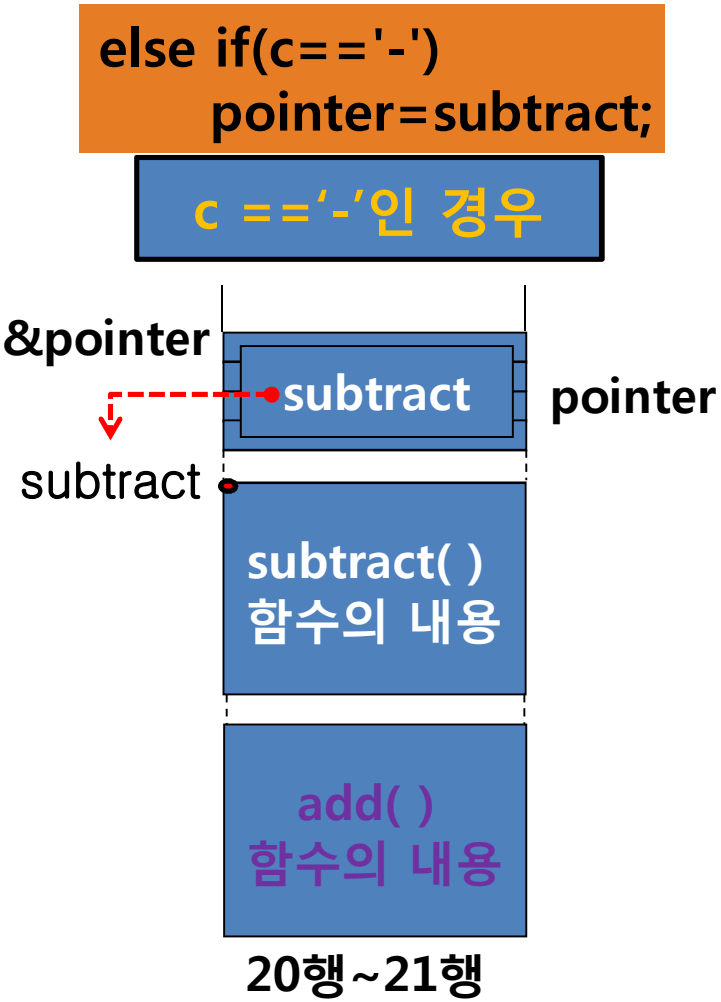
3.5 함수 포인터 (4/5)---[3-16.c 실습]

[3-16.c 핵심코드]

```
int x, z;
char   c;
void (*pointer) (int, int);

scanf("%d %c %d", &x, &c, &z);

   if(c=='+')
      pointer=add;

   else if(c=='-')
      pointer=subtract;

   pointer(x,z);
```

3.5 함수 포인터 (5/5)---[3-16.c 분석]





3.5 함수 포인터 (5/5)---[3-16.c 분석]

▶ 함수 포인터의 필요성

✓ '일반적인 함수 호출 보다 빠른 처리 속도를 기대한다.'

✓ 사용 분야

- 컴파일러, 인터프리터, 게임 프로그래밍과 같은 시스템 프로그래밍 분야

공부한 내용 떠올리기

- ▶ 포인터의 역할
- ▶ 포인터 변수의 선언과 사용 방법
- ▶ 포인터를 잘못 사용하는 경우
- ▶ 다차원 포인터 변수의 선언과 사용 방법
- ▶ 주소의 가감산
- ▶ 함수 포인터

무관심 (출처: 사랑과 지혜의 탈무드)