기초프로그래밍

제12장 포인터

Sangsoo Lim
CSAI
Dongguk University

차례

- 포인터란
- 포인터 변수의 선언과 사용
- 다차원 포인터 변수의 선언과 사용
- 주소의 가감산
- 함수 포인터

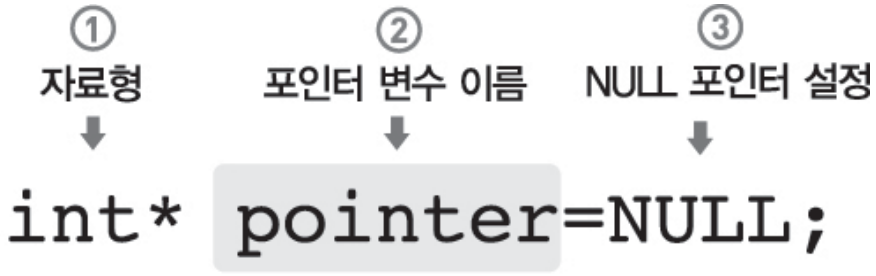
포인터란

- **포인터란?**
  - 주소를 저장하는 변수이다.
  - C언어의 장점 중에 하나가 바로 포인터(포인터 변수)이다.

- **포인터를 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
  - 동적 할당된 메모리 영역(힙영역)에 접근과 조작 용이

포인터 변수의 선언과 사용

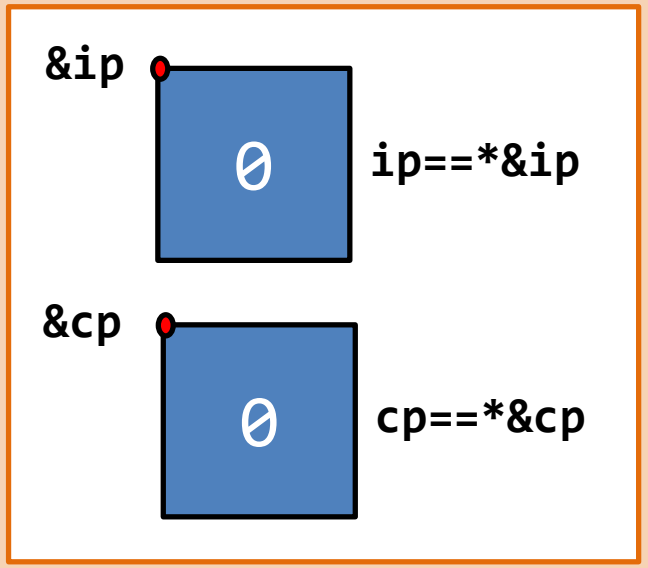
- **포인터 변수의 선언**
  - **자료형:** 포인터 변수의 자료형 지정, 자료형 다음에 * 연산자를 붙임
  - **포인터 변수 이름:** 주소를 저장할 변수의 이름 지정
  - **NULL 포인터 설정:** 포인터 변수 선언 시 NULL로 초기화




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

포인터 변수의 선언과 사용

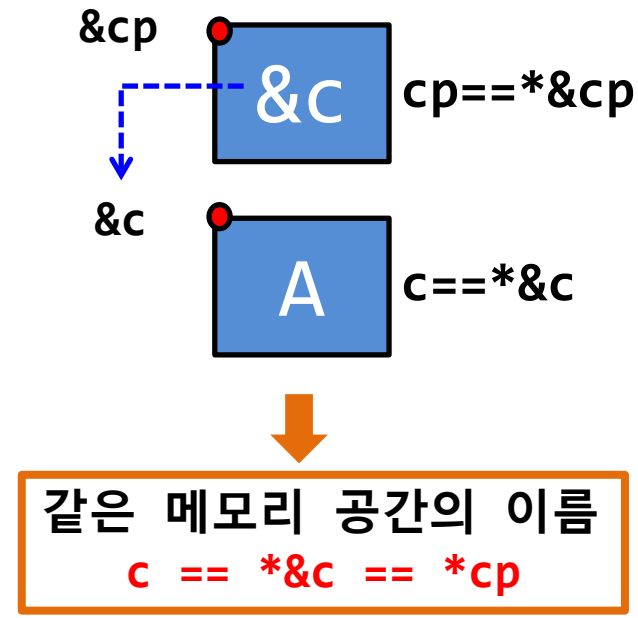
```c
/* 12-2.c */

#include <stdio.h>
int main(void)
{
    // 포인터 변수 선언
    char* cp = NULL;
    int* ip = NULL;
    printf("%p %p\n", (void*)&cp, (void*)cp);
    printf("%p %p\n", (void*)&ip, (void*)ip);
    printf("%lu %lu\n", sizeof(char*), sizeof(int*)); // 포인터 크기 출력
    printf("%lu %lu\n", sizeof(cp), sizeof(ip)); // 포인터 변수 크기 출력
    return 0;
}
```



모든 포인터 변수는 4바이트 이다.

포인터 변수의 선언과 사용

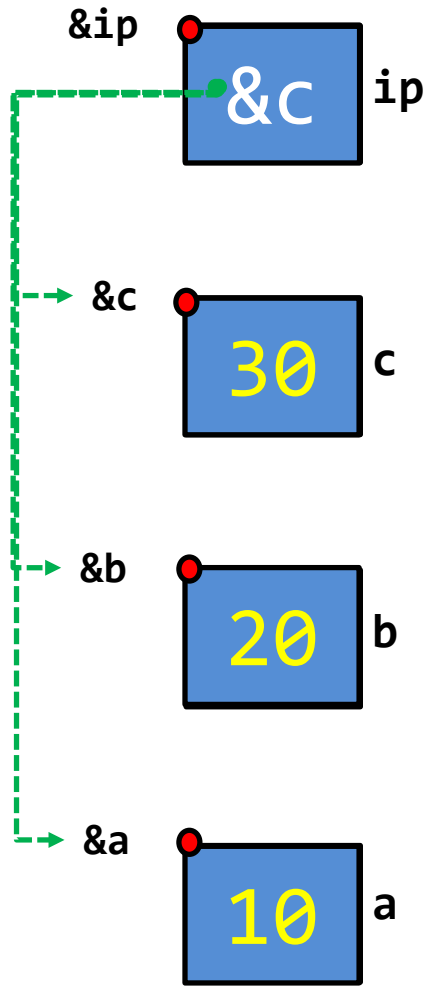
```c
#include <stdio.h>
int main(void)
{
    char c = 'A';
    char* cp = NULL;

    cp = &c;  // 주소 저장

    // 변수 c의 주소와 값 출력
    printf("%p %c %c \n", (void*)&c, c, *&c);

    // 포인터 변수 cp의 주소, cp가 가리키는 주소, *cp 값 출력
    printf("%p %p %c \n", (void*)&cp, (void*)cp, *cp);

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



포인터 변수의 선언과 사용

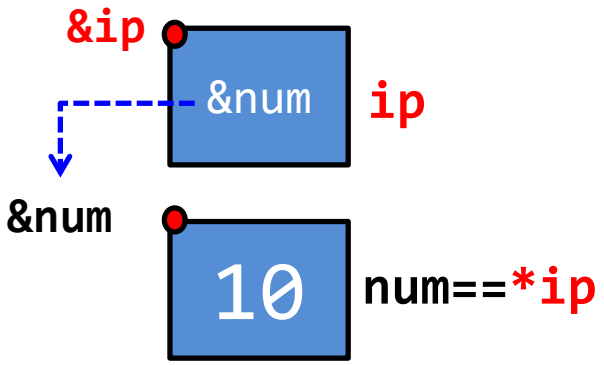
```c
#include <stdio.h>
int main(void)
{
    int a = 0, b = 0, c = 0;
    int* ip = NULL; // 포인터 변수 선언

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

    ip = &b; // b의 주소 저장
    *ip = 20; // b에 20을 저장
    printf("%d %d %d %d\n", a, b, c, *ip);

    ip = &c; // c의 주소 저장
    *ip = 30; // c에 30을 저장
    printf("%d %d %d %d\n", a, b, c, *ip);

    return 0;
}
```

```
10 0 0 10
10 20 0 20
10 20 30 30
```



포인터 변수의 선언과 사용

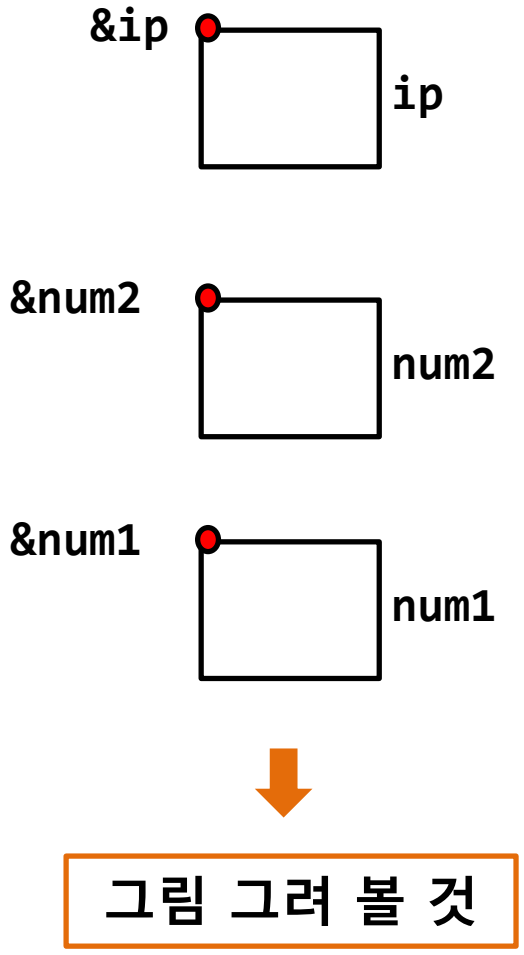
```c
#include <stdio.h>
int main(void)
{
    int num = 10;
    int* ip = NULL;  // 포인터 변수 선언

    ip = &num;       // 주소 저장

    printf("%p %p %d \n", (void*)&*&ip,
(void*)*&ip, **&ip);
    printf("%p %p %d \n", (void*)&ip,
(void*)ip, *ip);
    return 0;
}
```




```
ecdb90c0 ecdb90bc 10
ecdb90c0 ecdb90bc 10
```

포인터 변수의 선언과 사용

```c
/* 12-6.c */
#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;
}
```



포인터 변수의 선언과 사용

- **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;
}
```

포인터 변수의 선언과 사용

- **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

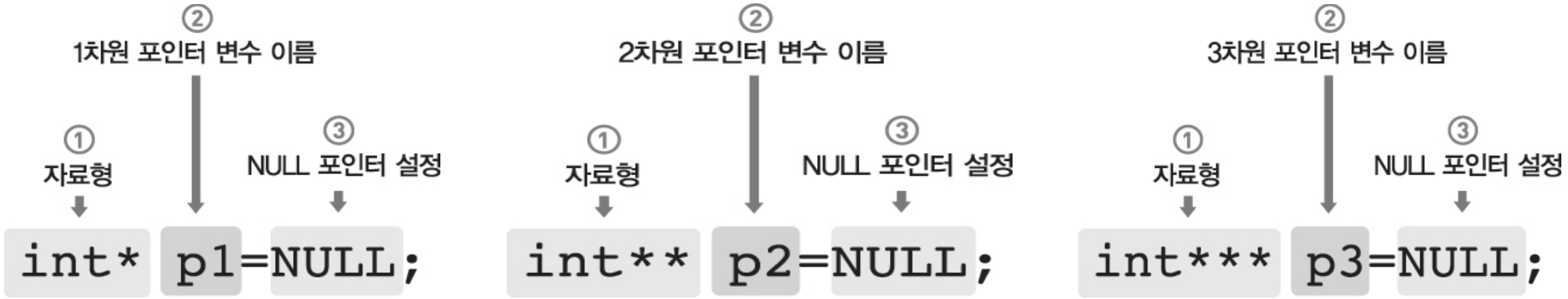
같은 표현
==

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

포인터 변수의 선언과 초기화를
동시에 수행

다차원 포인터 변수의 선언과 사용

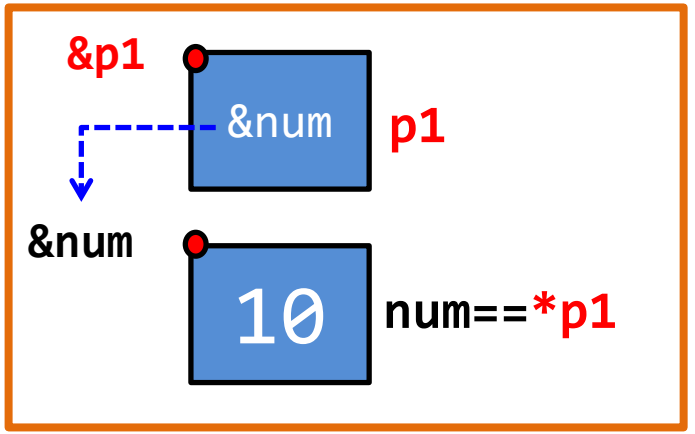
- **다차원 포인터 변수란?**
  - 2차원 이상의 포인터 변수



다차원 포인터 변수의 선언과 사용

- **1차원 포인터 변수**
  - **일반 변수의 주소를 저장**

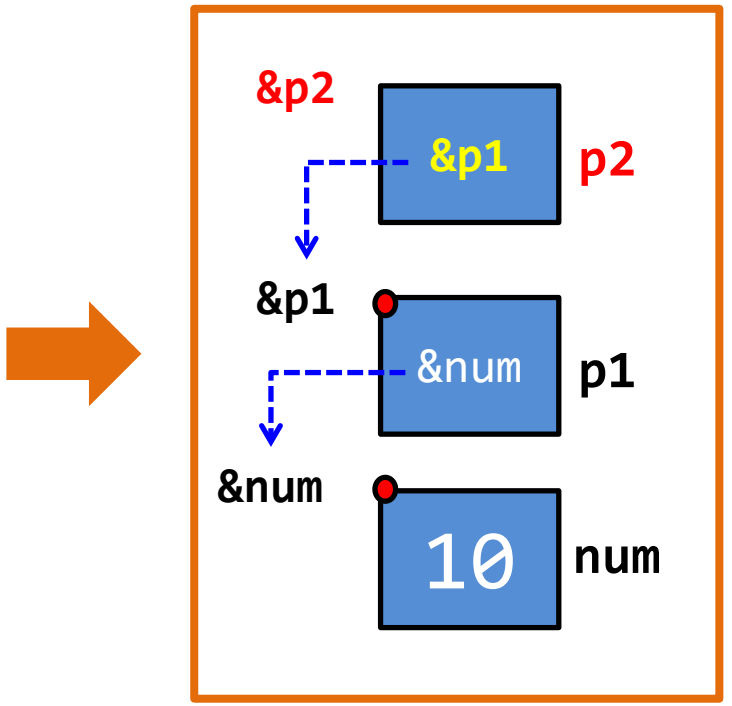
```
#include <stdio.h>
int main(void)
{
    int num=10;
    int* p1=NULL;
    p1=&num;
    return 0;
}
```



다차원 포인터 변수의 선언과 사용

- **2차원 포인터 변수**
  - **1차원 포인터 변수의 주소를 저장**

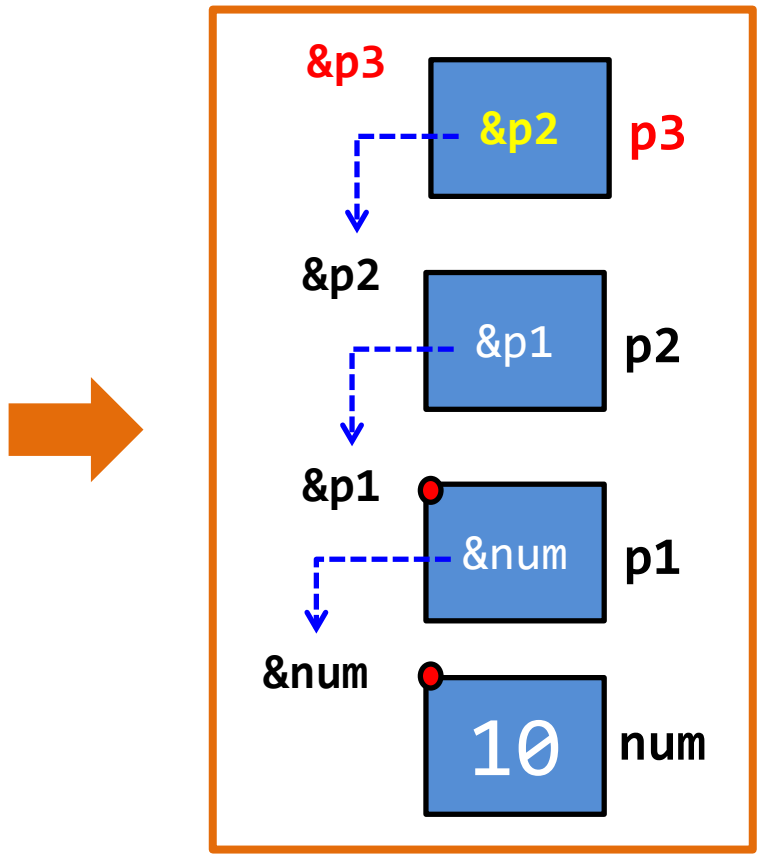
```
#include <stdio.h>
int main(void)
{
    int num=10;
    int*  p1=NULL;
    int** p2=NULL;

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



다차원 포인터 변수의 선언과 사용

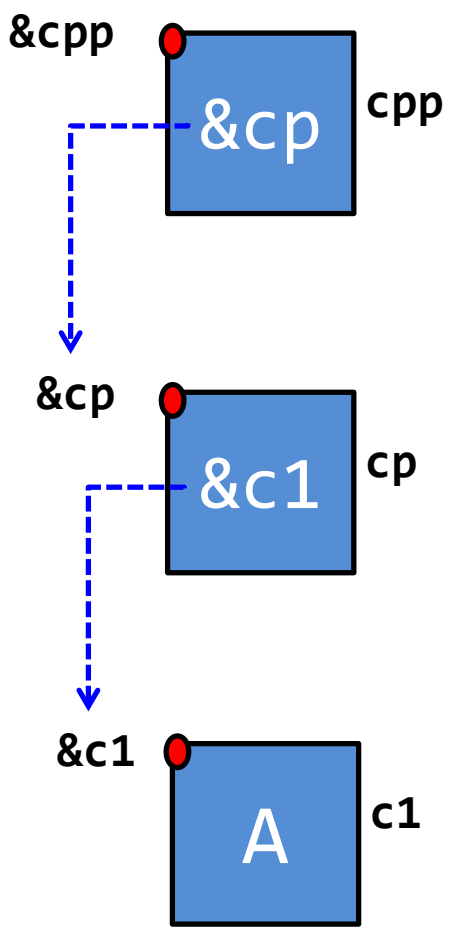
- **3차원 포인터 변수**
  - **2차원 포인터 변수의 주소를 저장**

```
#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;
}
```



다차원 포인터 변수의 선언과 사용

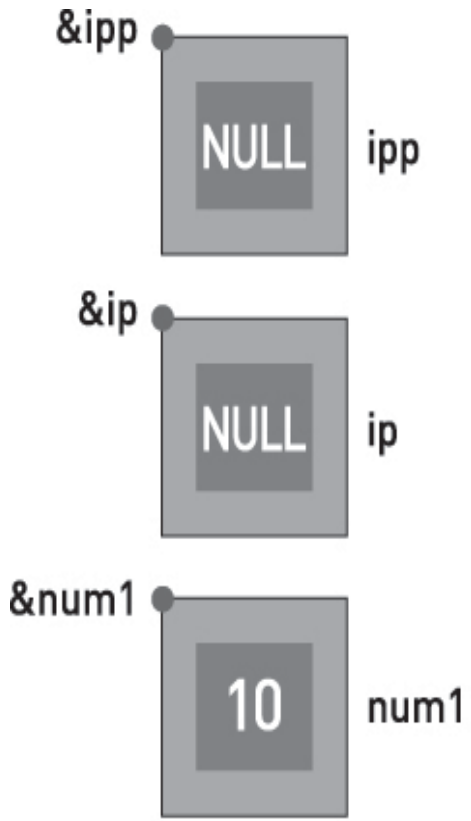
```c
#include <stdio.h>
int main(void)
{
    char c1 = 'A';
    char* cp = NULL;
    char** cpp = NULL;

    cp = &c1;
    cpp = &cp;

    printf("%c %p %p \n", c1, cp, cpp);
    printf("%p %p %p \n", &c1, &cp, &cpp);
    printf("%c %c %c \n", c1, *cp, **cpp);

    return 0;
}
```



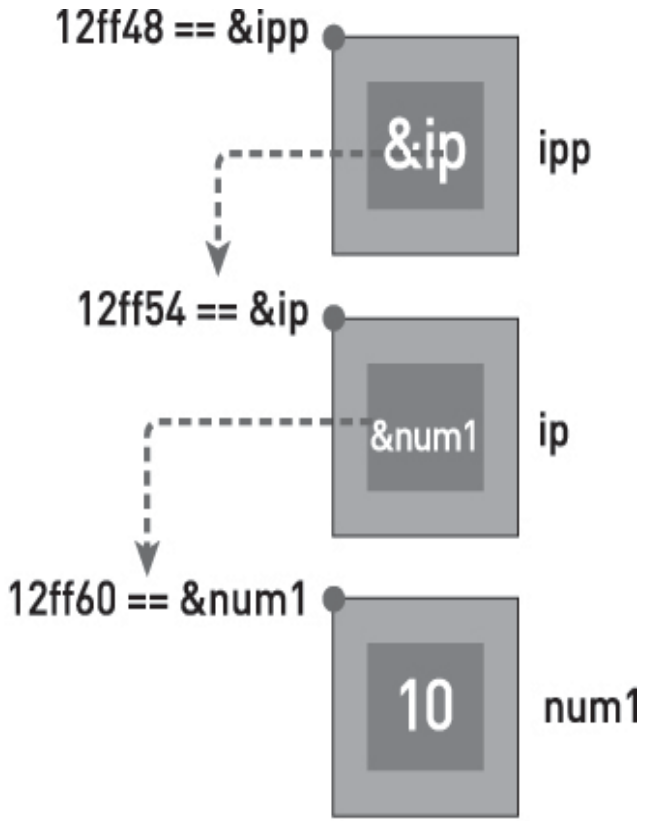
다차원 포인터 변수의 선언과 사용

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

    ip = &num1;
    ipp = &ip;

    printf("%d %p %p \n", num1, ip, ipp);
    printf("%p %p %p \n", &num1, &ip, &ipp);
    printf("%d %p %p \n", *&num1, *&ip, *&ipp);

    printf("%d %d %d \n", num1, *ip, **ipp);
    printf("%p %p %p \n", &num1, ip, *ipp);

    return 0;
}
```

다차원 포인터 변수의 선언과 사용



4행~6행

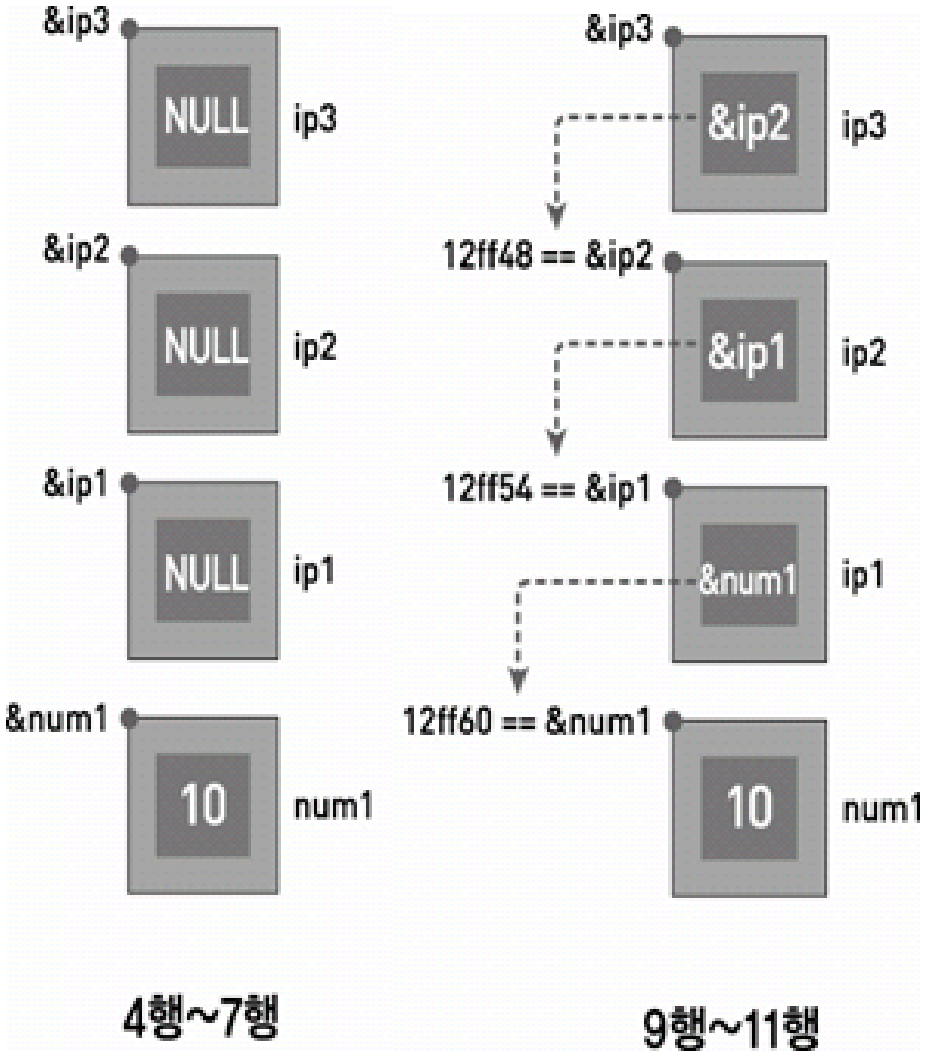


8행~9행

다차원 포인터 변수의 선언과 사용

```c
#include <stdio.h>

int main(void)
{
   int num1 = 10;
   int* ip1 = NULL;
   int** ip2 = NULL;
   int*** ip3 = NULL;

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

   // Print values of num1 using different levels of pointers
   printf("%d %d %d %d \n", num1, *ip1, **ip2, ***ip3);

   // Print addresses of num1, ip1, *ip2, **ip3
   printf("%p %p %p %p \n", &num1, ip1, *ip2, **ip3);

   // Print addresses of ip1, ip2, *ip3
   printf("%p %p %p \n", &ip1, ip2, *ip3);

   // Print addresses of ip2 and ip3
   printf("%p %p \n", &ip2, ip3);

   // Print sizes of int and different pointer types
   printf("%zu %zu \n", sizeof(int), sizeof(int*));
   printf("%zu %zu\n", sizeof(int**), sizeof(int***));

   // Print sizes of num1, ip1, ip2, ip3
   printf("%zu %zu \n", sizeof(num1), sizeof(ip1));
   printf("%zu %zu\n", sizeof(ip2), sizeof(ip3));

   return 0;
}
```

다차원 포인터 변수의 선언과 사용



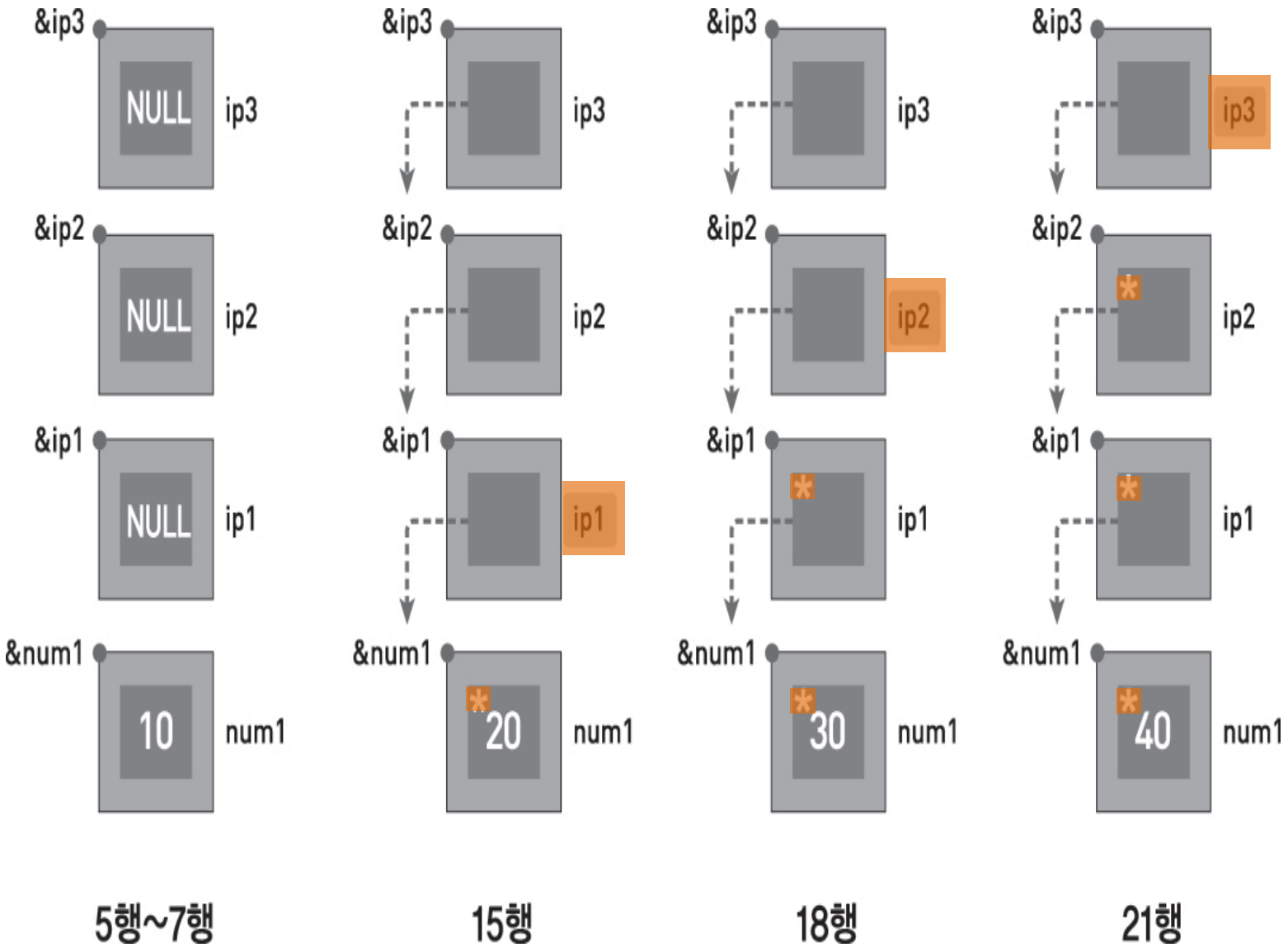

```
   ip3==&ip2
  *ip3==ip2==&ip1
 **ip3==*ip2==ip1==&num1
***ip3==**ip2==*ip1==num1==10
```

다차원 포인터 변수의 선언과 사용

```c
/* 12-10.c */
   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);
```

```
10 10 10 10
20 20 20 20
30 30 30 30
40 40 40 40
```

다차원 포인터 변수의 선언과 사용



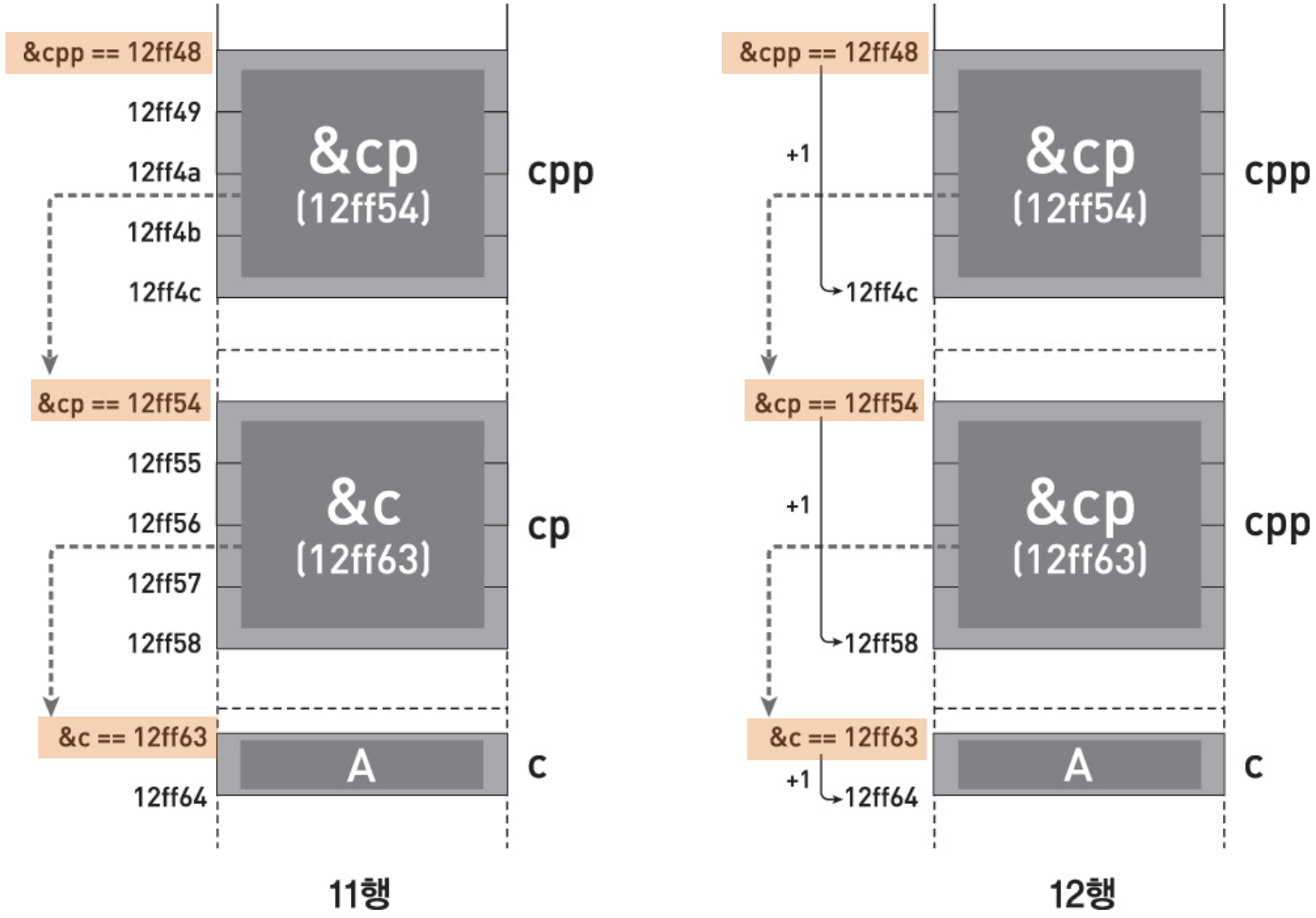
주소의 가감산

```
/* 12-11.c */
#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;
}
```

```
/* 12-11.c */
#include <stdio.h>

int main() {
   char c = 'A';
   char* cp = NULL;
   char** cpp = NULL;

   cp = &c;
   cpp = &cp;

   printf("%p %p %p \n", (void*)&c, (void*)&cp, (void*)&cpp);
   printf("%p %p %p \n", (void*)(&c + 1), (void*)(&cp + 1),
(void*)(&cpp + 1));

   printf("%c %p %p \n", c, (void*)cp, (void*)cpp);
   printf("%c %p %p \n", c + 1, (void*)(cp + 1), (void*)(cpp + 1));

   return 0;
}
```

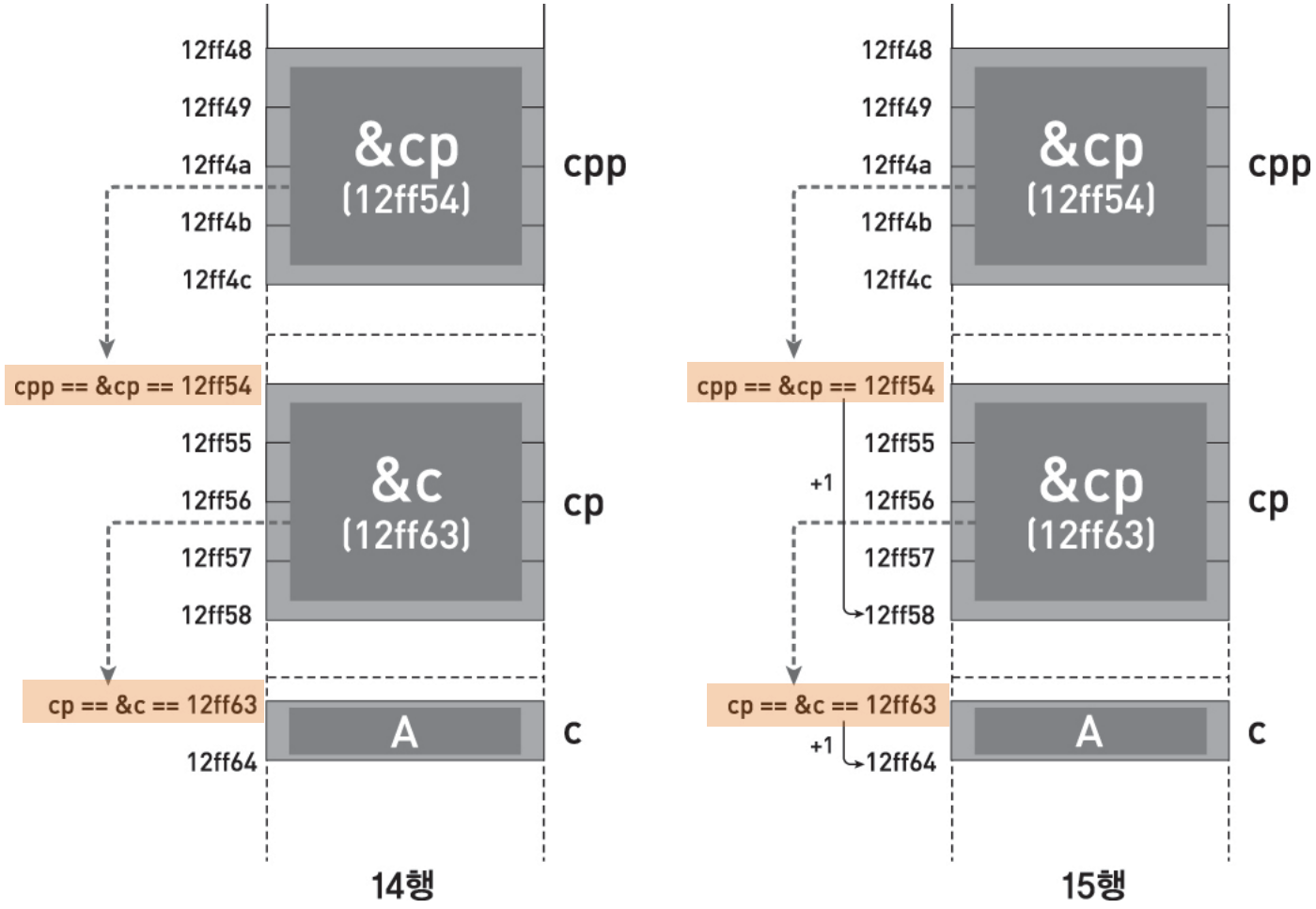
주소의 가감산

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



주소의 가감산

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

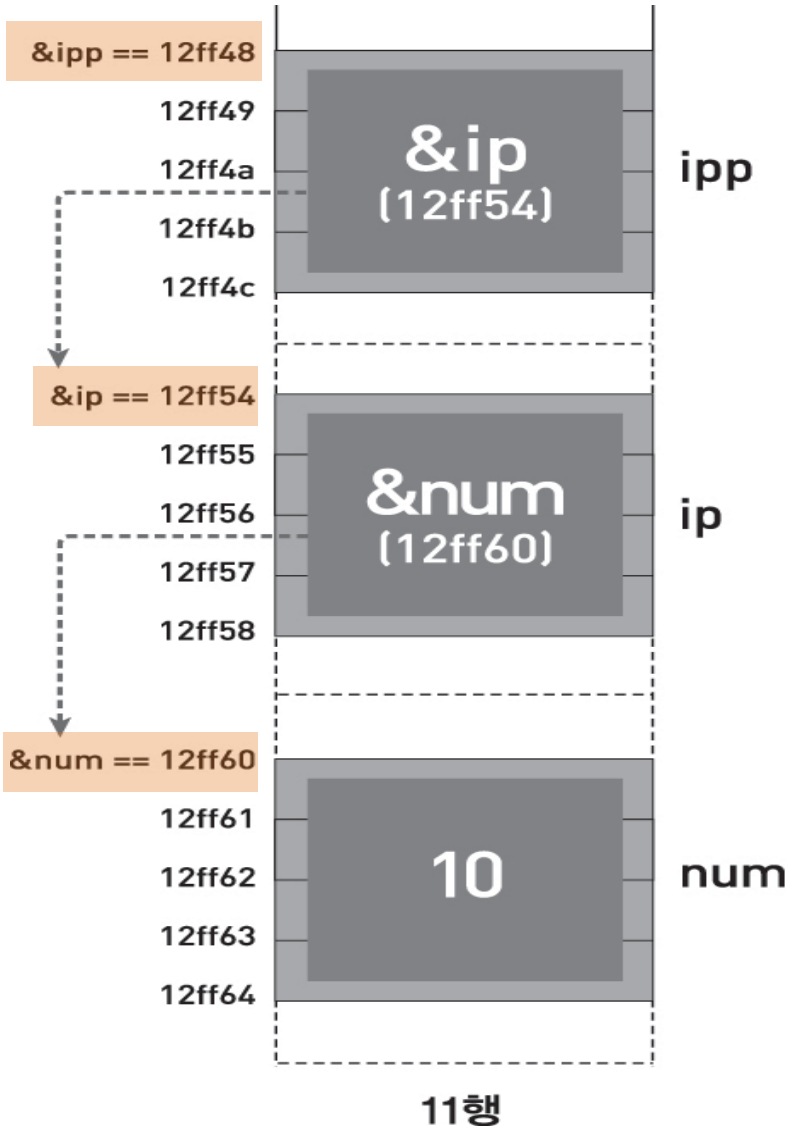


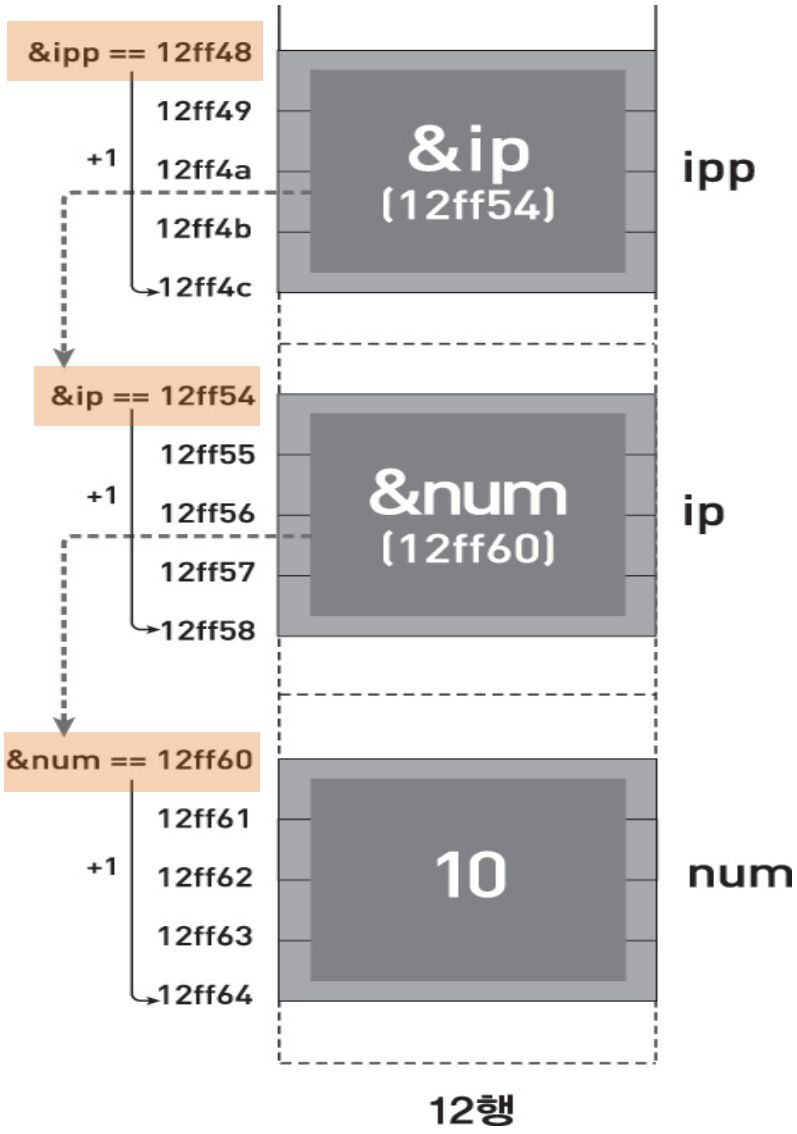
주소의 가감산

```
/* 12-12.c */
#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;
}
```

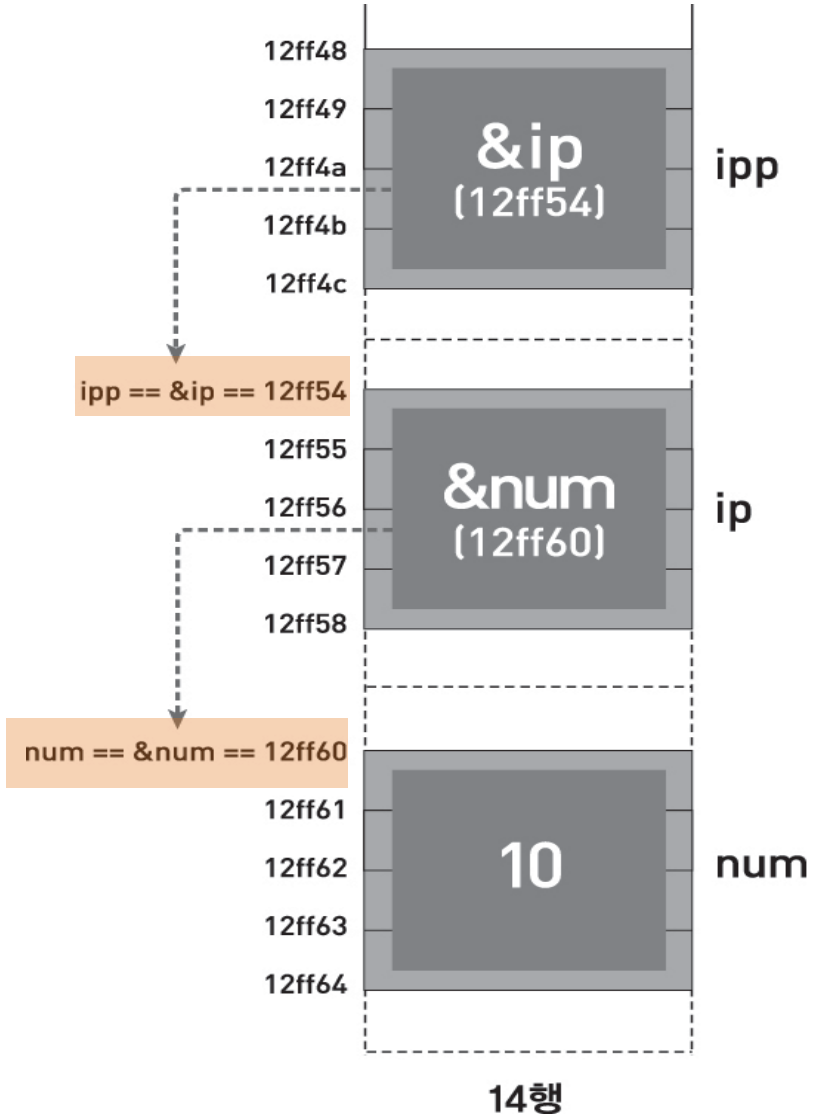
```
/* 12-12.c */
#include <stdio.h>

int main() {
   int num = 10;
   int* ip = NULL;
   int** ipp = NULL;

   ip = &num;
   ipp = &ip;

   printf("%p %p %p \n", (void*)&num, (void*)&ip, (void*)&ipp);
   printf("%p %p %p \n", (void*)(&num + 1), (void*)(&ip + 1),
(void*)(&ipp + 1));

   printf("%d %p %p \n", num, (void*)ip, (void*)ipp);
   printf("%d %p %p \n", num + 1, (void*)(ip + 1), (void*)(ipp + 1));

   return 0;
}
```

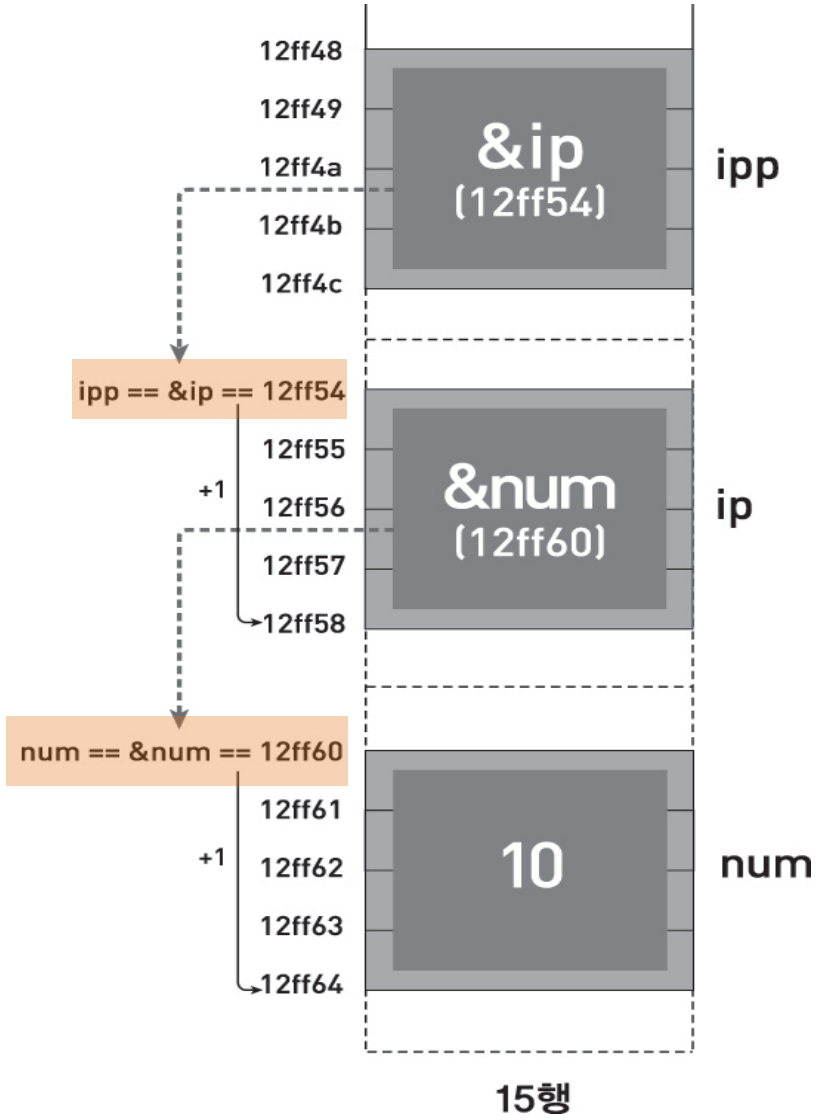
주소의 가감산


&ipp == 12ff48
12ff49
12ff4a
12ff4b
12ff4c
&ip
(12ff54)
ipp
&ip == 12ff54
12ff55
12ff56
12ff57
12ff58
&num
(12ff60)
ip
&num == 12ff60
12ff61
12ff62
12ff63
12ff64
10
num
11행


&ipp == 12ff48
12ff49
+1
12ff4a
12ff4b
12ff4c
&ip
(12ff54)
ipp
&ip == 12ff54
12ff55
+1
12ff56
12ff57
12ff58
&num
(12ff60)
ip
&num == 12ff60
12ff61
+1
12ff62
12ff63
12ff64
10
num
12행

주소의 가감산

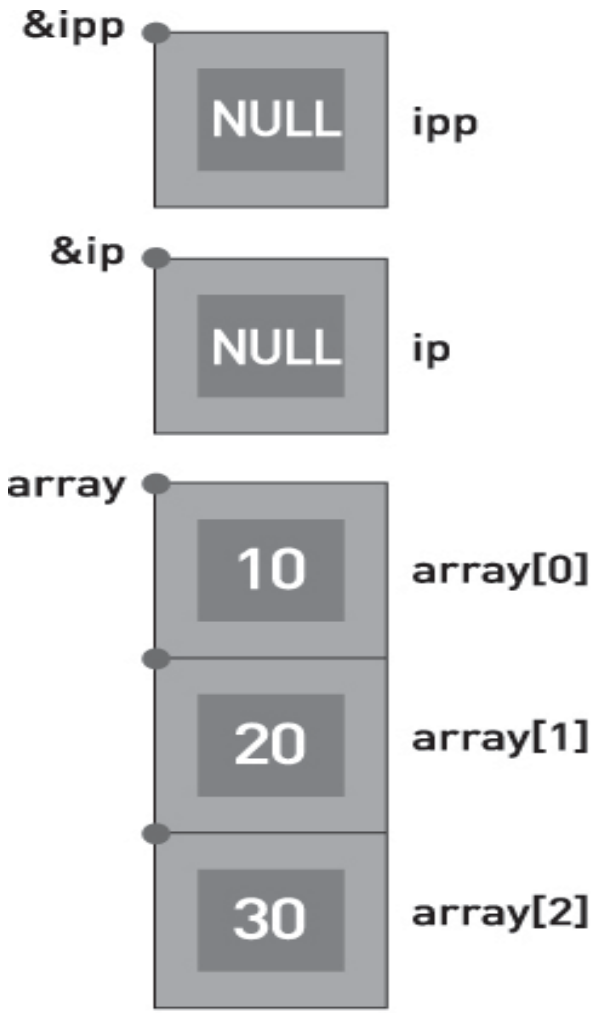

12ff48
12ff49
12ff4a
12ff4b
12ff4c
&ip
(12ff54)
ipp
ipp == &ip == 12ff54
12ff55
12ff56
12ff57
12ff58
&num
(12ff60)
ip
num == &num == 12ff60
12ff61
12ff62
12ff63
12ff64
10
num
14행


12ff48
12ff49
12ff4a
12ff4b
12ff4c
&ip
(12ff54)
ipp
ipp == &ip == 12ff54
+1
12ff55
12ff56
12ff57
12ff58
&num
(12ff60)
ip
num == &num == 12ff60
+1
12ff61
12ff62
12ff63
12ff64
10
num
15행

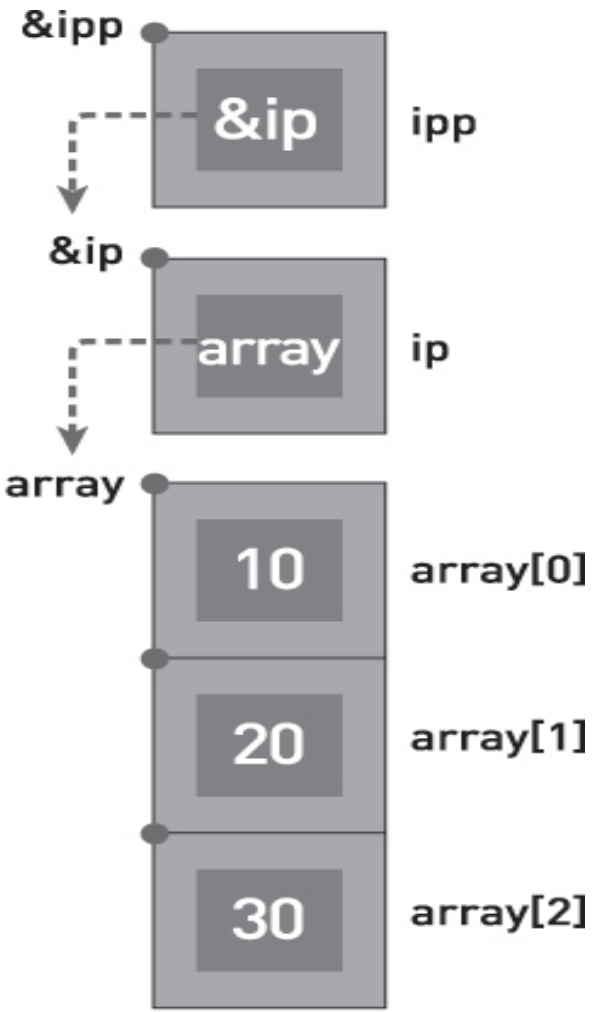
주소의 가감산

```c
/* 12-13.c */
#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;
}
```

주소의 가감산



4행~6행



8행~9행

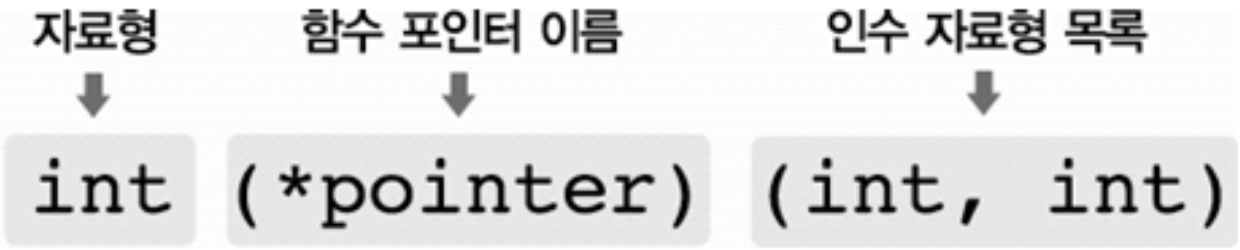
함수 포인터

- **함수 이름은 '함수의 시작 주소'**

```c
/* 12-14.c */
#include <stdio.h>

int main(void) {
   printf("%p %p %p \n", (void*)main, (void*)printf, (void*)scanf);
   return 0;
}
```

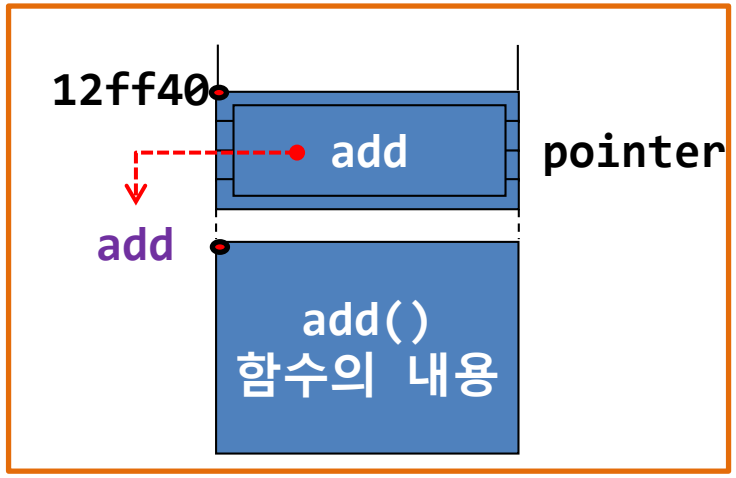
함수 포인터

- **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



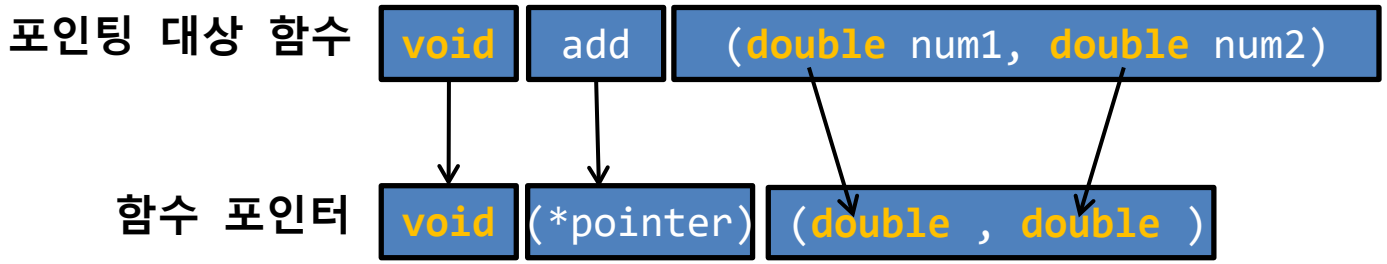
함수 포인터

```c
/* 12-15.c */
#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 : %p\n", (void*)add);
   printf("함수 포인터의 주소 : %p \n", (void*)&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);
}
```




```
add 함수의 주소 : 0x5ddadbab0215
함수 포인터의 주소 : 0x7ffee2960420
3.100000 + 5.100000 = 8.200000입니다.
```

함수 포인터



함수포인터에 함수 시작 주소 저장

```
pointer = add;
```

함수포인터를 이용한 함수 호출

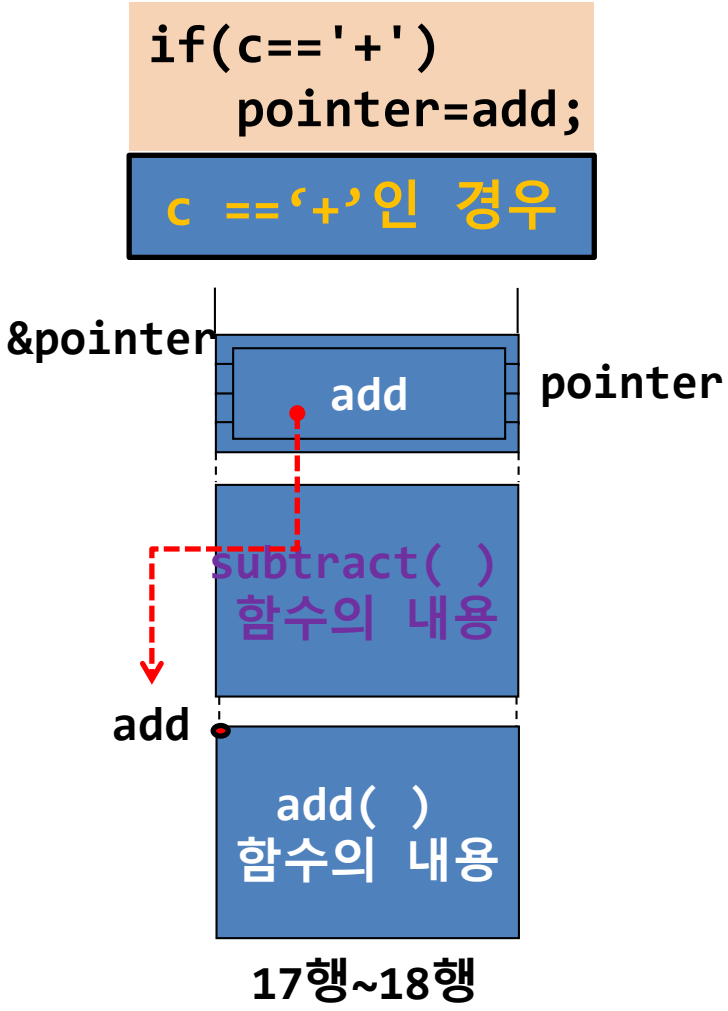
```
pointer(3.1, 5.1);
```

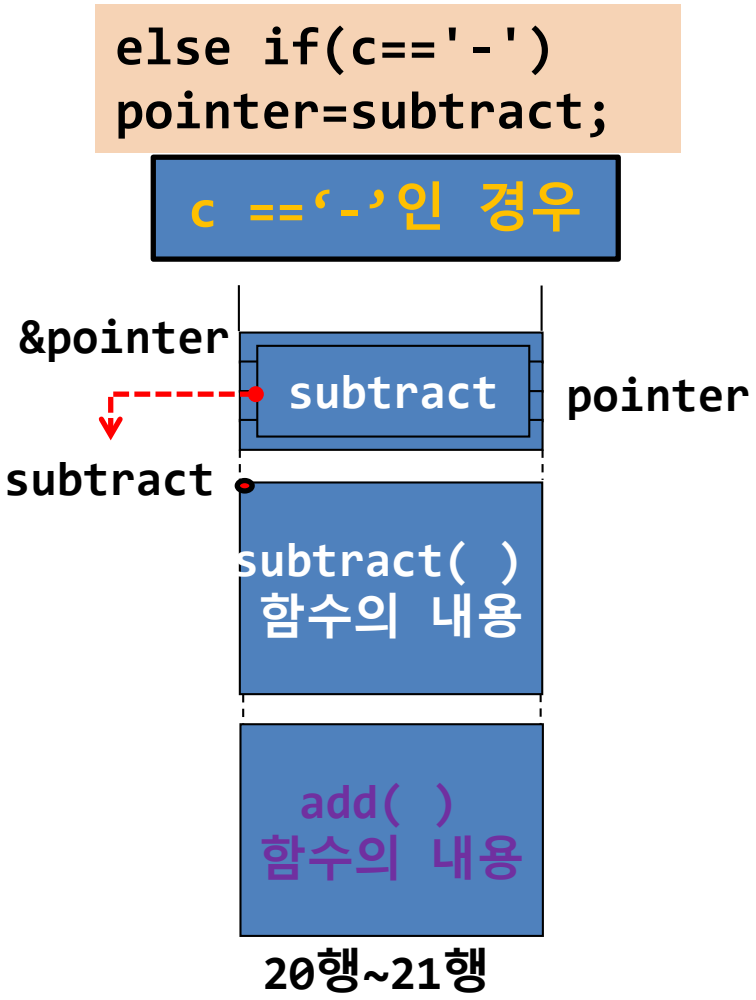
함수 포인터

```c
/* 12-16.c */
#include <stdio.h>

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

void add(int a, int b) {
    printf("%d + %d = %d\n", a, b, a + b);
}

void subtract(int a, int b) {
    printf("%d - %d = %d\n", a, b, a - b);
}
```

```c
int main() {
    printf("Enter expression (e.g., 5 + 3):
");
    scanf("%d %c %d", &x, &c, &z);

    if (c == '+') {
        pointer = add;
    } else if (c == '-') {
        pointer = subtract;
    } else {
        printf("Invalid operator!\n");
        return 1; // Exit the program with
an error code
    }

    pointer(x, z); // Call the selected
function

    return 0;
}
```

함수 포인터


if(c=='+')
pointer=add;
c ==‘+’인 경우
&pointer
add
pointer
subtract( )
함수의 내용
add
add( )
함수의 내용
17행~18행


else if(c=='-')
pointer=subtract;
c ==‘-’인 경우
&pointer
subtract
pointer
subtract
subtract( )
함수의 내용
add( )
함수의 내용
20행~21행

함수 포인터

- **함수 포인터의 필요성**
  - 일반적인 함수 호출 보다 빠른 처리 속도를 기대한다.
  - 사용 분야
    - 컴파일러, 인터프리터, 게임 프로그래밍과 같은 시스템 프로그래밍 분야

Summary

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