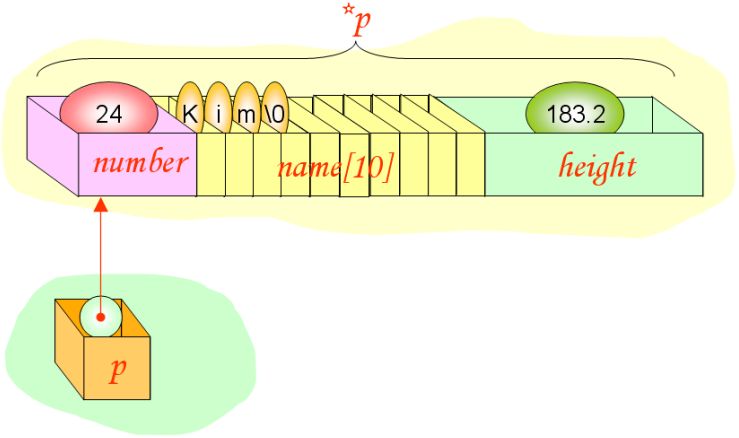
구조체 Part 2

구조체와 포인터

- 구조체를 가리키는 포인터

```
struct student s = { 20070001, "홍길동", 180.2 };
struct student *p;

p = &s;

printf("학번=%d 이름=%s 키=%f \n", s.number, s.name, s.height);
printf("학번=%d 이름=%s 키=%f \n", (*p).number, (*p).name, (*p).height);
```

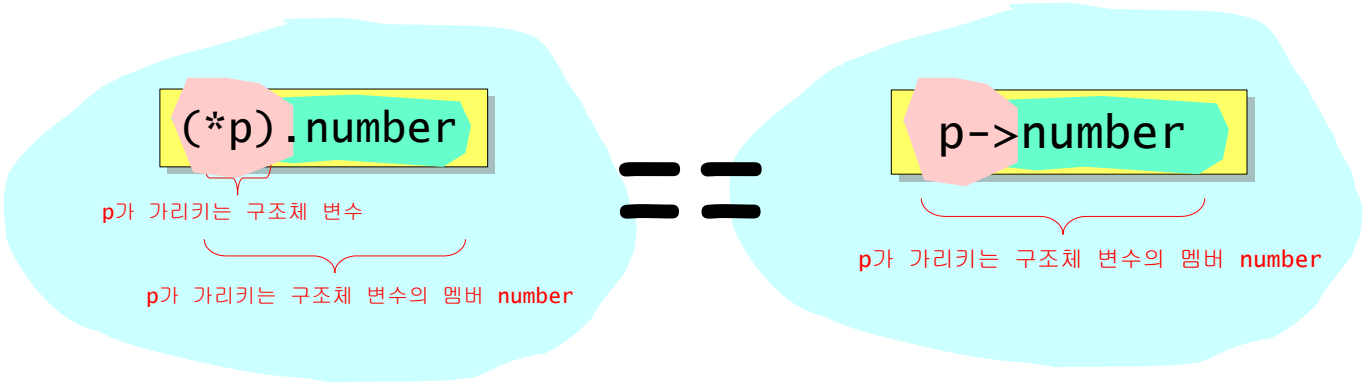
반드시 괄호가 있어야 한다



→ 연산자

- "→" 연산자는 구조체 포인터를 사용하여 구조체 멤버를 참조할 때 사용

```
struct student *p;

struct student s = { 20070001, "홍길동", 180.2 };
struct student *p;

p = &s;
printf("학번=%d 이름=%s 키=%f \n", s.number, s.name, s.height);
printf("학번=%d 이름=%s 키=%f \n", (*p).number,(*p).name,(*p).height);
printf("학번=%d 이름=%s 키=%f \n", p->number, p->name, p->height);
```



예제

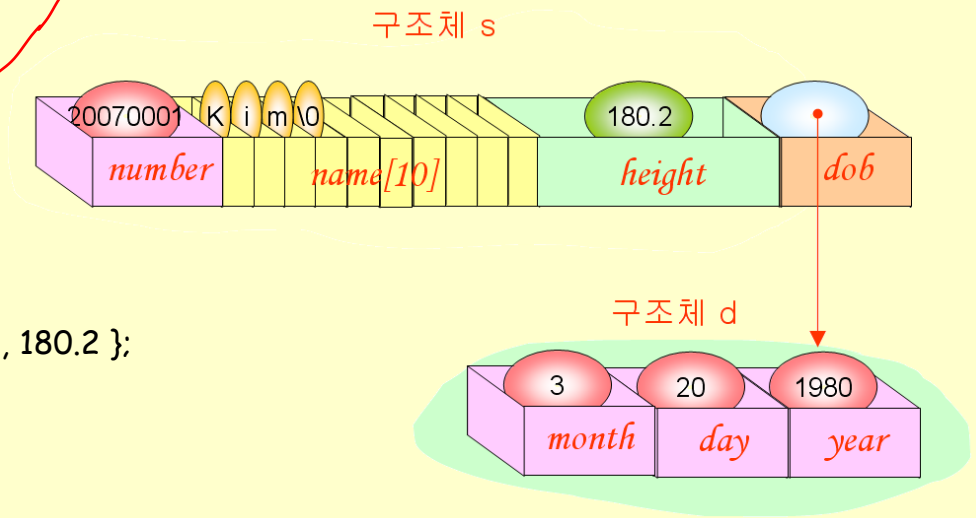
```c
// 포인터를 통한 구조체 참조
#include <stdio.h>

struct student {
	int number;
	char name[20];
	double height;
};

int main(void)
{
	struct student s = { 20070001, "홍길동", 180.2 };
	struct student *p;

	p = &s;

	printf("학번=%d 이름=%s 키=%f \n", s.number, s.name, s.height);
	printf("학번=%d 이름=%s 키=%f \n", (*p).number,(*p).name,(*p).height);
	printf("학번=%d 이름=%s 키=%f \n", p->number, p->name, p->height);

	return 0;
}
```

```
학번=20070001 이름=홍길동 키=180.200000
학번=20070001 이름=홍길동 키=180.200000
학번=20070001 이름=홍길동 키=180.200000
```

구조체포인터를 멤버로 가지는 구조체

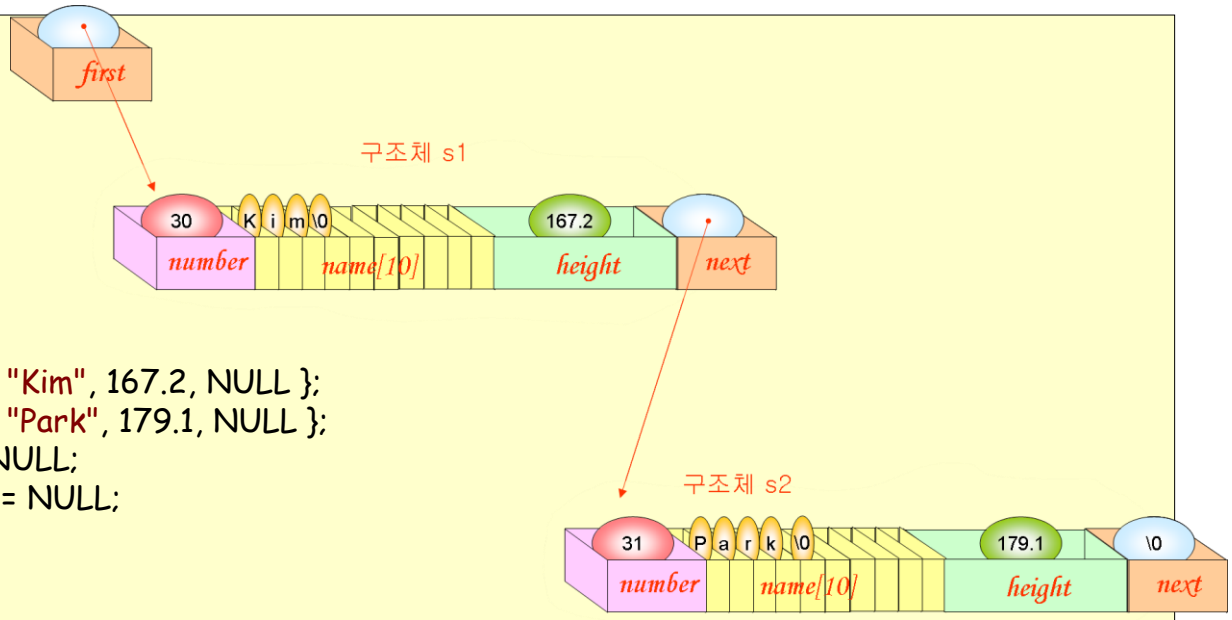
```
struct date {
        int month;
        int day;
        int year;
};
struct student {
        int number;
        char name[20];
        double height;
        struct date *dob;
};
int main(void)
{
        struct date d = { 3, 20, 1980 };
        struct student s = { 20070001, "Kim", 180.2 };

        s.dob = &d;

        printf("학번: %d\n", s.number);
        printf("이름: %s\n", s.name);
        printf("신장: %f\n", s.height);
        printf("생년월일: %d년 %d월 %d일\n", s.dob->year, s.dob->month, s.dob->day);
        return 0;
}
```

학번: 20070001
이름: Kim
신장: 180.200000
생년월일: 1980년 3월 20일



자기 참조 구조체(self-referential structure) : 연결리스트(linked List)

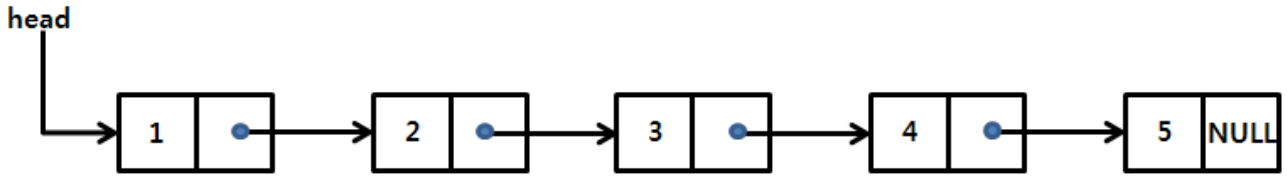
```c
struct student {
	int number;
	char name[10];
	double height;
	struct student *next;
};

int main(void)
{
	struct student s1 = { 30, "Kim", 167.2, NULL };
	struct student s2 = { 31, "Park", 179.1, NULL };
	struct student *first = NULL;
	struct student *current = NULL;

	first = &s1;
	s1.next = &s2;
	s2.next = NULL;

	current = first;
	while( current != NULL )
	{
		printf("학생의 번호=%d 이름=%s, 키=%f\n", current->number,
				current->name, current->height);
		current = current->next;
	}
}
```




```
학생의 번호=30 이름=Kim, 키=167.200000
학생의 번호=31 이름=Park, 키=179.100000
```

Linked List

- 아래 그림에 보여준 바와 같은 연결리스트를 만들고, 1부터 순서대로 값을 출력하는 프로그램을 작성하시오.

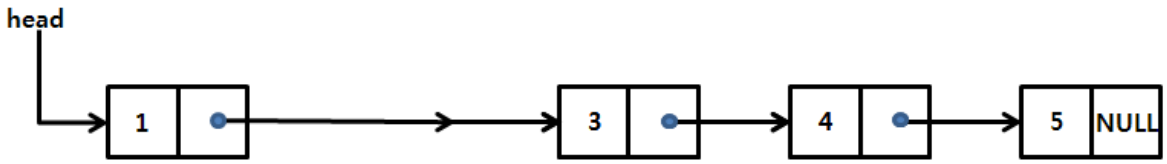



```
struct LinkedList {
        int data ;
        struct linked_list *next ;
};
struct LinkedList node[5] ;
struct LinkedList *head;
struct LinkedList *p;
```

```
struct LinkedList *head;
struct LinkedList *p;

head = &node[0];       // head = node;
p = head;
p->data = 1;

// p는 node[0]를 가르키고 있다
for(i=1; i<5; i++) {
   p->next = &node[i] ;
   p = p->next ;
   p->data = i + 1 ;
}

p->next = NULL;
p = head;

whilr(p->next) {
   printf("node Value = %d₩n", p->data);
}
printf("node Value = %d₩n", p->data);
```

Node의 삭제

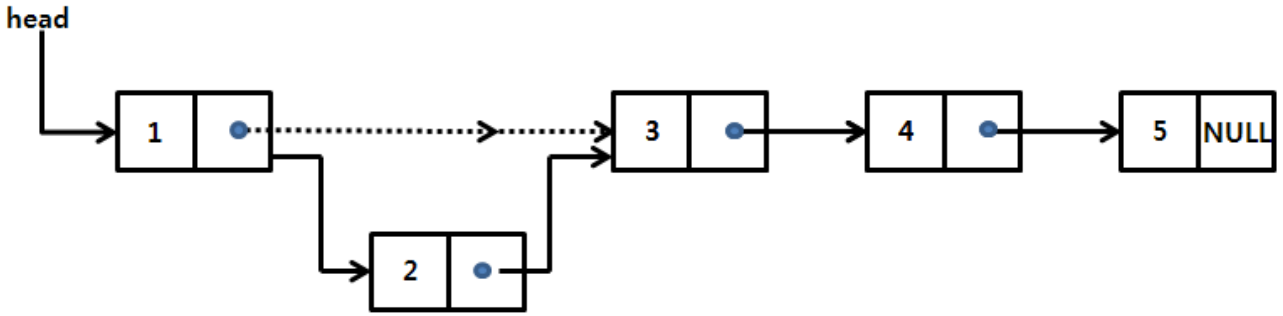
- 상기 주어진 연결리스트에서 아래 그림처럼 node 1을 삭제하고자 하는데, 어떻게 하면 되는지 프로그램을 작성하시오.



Node의 삽입

[ANS] head->next = head->next->next ;

앞의 연결리스트에 다시 NODE 1을 아래 그림처럼 삽입하고자 한다. 어떻게 하면 되는지 프로그램을 작성하시오



[ANS]

```
p = &node[1] ;
p->data = 2;
p->next = head->next;
head->next = p ;
```

구조체와 함수

- *구조체*를 함수의 인수로 전달하는 경우
  - 구조체의 복사본이 함수로 전달되게 된다.
  - 만약 구조체의 크기가 크면 그만큼 시간과 메모리가 소요된다.

```
int equal(struct student s1, struct student s2)
{
      if( strcmp(s1.name, s2.name) == 0 )
            return 1;
      else
            return 0;
}
```

- *구조체의 포인터*를 함수의 인수로 전달하는 경우
  - 시간과 공간을 절약할 수 있다.

```
int equal(struct student const *p1, struct student const *p2)
{
      if( strcmp(p1->name, p2->name) == 0 )
            return 1;
      else
            return 0;
}
```

구조체를 반환하는 경우

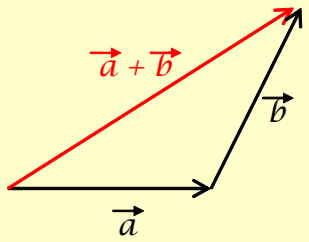
- 복사본이 반환된다.

```
struct student make_student(void)
{
      struct student s;

      printf("나이:");
      scanf("%d", &s.age);
      printf("이름:");
      scanf("%s", s.name);
      printf("키:");
      scanf("%f", &s.height);

      return s;
}
```

구조체 s의 복사본이 반환된다.

예제

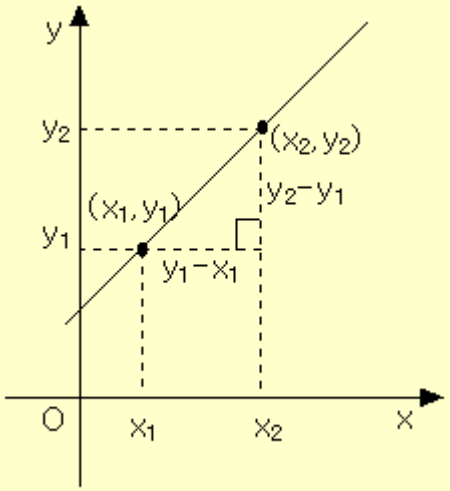
```
#include <stdio.h>

struct vector {
	float x;
	float y;
};
struct vector get_vector_sum(struct vector a, struct vector b);

int main(void)
{
	struct vector a = { 2.0, 3.0 };
	struct vector b = { 5.0, 6.0 };
	struct vector sum;

	sum = get_vector_sum(a, b);
	printf("벡터의 합은 (%f, %f)입니다.\n", sum.x, sum.y);

	return 0;
}

struct vector get_vector_sum(struct vector a, struct vector b)
{
	struct vector result;

	result.x = a.x + b.x;
	result.y = a.y + b.y;

	return result;
}
```




```
벡터의 합은 (7.000000, 9.000000)입니다.
```

예제

```
#include <stdio.h>
struct point {
        int x;
        int y;
};
// 기울기와 y절편을 계산
int get_line_parameter(struct point p1, struct point p2, float *slope, float *yintercept)
{
        if( p1.x == p2.x )
                return (-1);
        else
        {
          *slope = (float)(p2.y - p1.y)/(float)(p2.x - p1.x);
          *yintercept = p1.y - (*slope) * p1.x;
          return (0);
        }
}
int main(void)
{
        struct point pt1 = {3, 3}, pt2 = {6, 6};
        float s,y;

        if( get_line_parameter(pt1, pt2, &s, &y) == -1 )
                printf("오류: 두점의 x좌표값이 동일합니다.\n");
        else
                printf("기울기는 %f, y절편은 %f\n", s, y);
        return 0;
}
```



기울기는 1.000000, y절편은 0.000000