

09

구조체

키보드에 있는 특수문자 이름

부호	영어이름	영어발음	한글이름
`	Single quotation	싱글쿼테이션	홀따옴표
~	TILDE, WIGGLE	틸더, 위글	지령이, 물결, 동안
!	EXCLAMATION POINT	엑스클러메이션 포인트	느낌표
@	AT SIGN	앳 사인	골뱅이
#	CROSSHATCH	크로스햇지	우물
\$	DOLLAR SIGN	달러 사인	달러 표시
%	PERCENT SIGN	퍼센트 사인	백분율, 퍼센트
^	CIRCUMFLEX	서컴플렉스	삿갓, 산표, 누승
&	AMPERSAND	앰퍼샌드	파배기, 만두표
*	ASTERISK	애스터리스크	눈표, 별표, 곱하기
(EFT PARENTHESIS	레프트 퍼렌써시스	왼쪽 소괄호
)	RIGHT PARENTHESIS	라이트 퍼렌써시스	오른쪽 소괄호
()	PARENTHESSES	퍼렌써시스	소괄호
-	HYPHEN	하이픈	빼기, 붙임표
_	UNDERSCORE	언더스커어	밑줄
=	EQUAL SIGN	이퀄 싸인	등호, 같음
+	PLUS SIGN	플러스 싸인	더하기
\	BACKSLASH	백슬래쉬	원 표시
	VERTICAL BAR	버티컬 바	수직선
[LEFT BRACKET	레프트 브레킷	왼쪽 대괄호
]	RIGHT BRACKET	라이트 브레킷	오른쪽 대괄호
[]	BRACKETS	브레킷	대괄호
{	LEFT BRACE	레프트 브레이스	왼쪽 중괄호
}	RIGHT BRACE	라이트 브레이스	오른쪽 중괄호
{ }	BRACES	브레이스	중괄호
;	SEMICOLON	세미콘론	머무름표, 쌍반점
:	COLON	콜론	그침표, 쌍점
"	QUOTATION MARK	쿼테이션 마크	따옴표, 가지런표
'	APOSTROPHE	어퍼스트로피	홀 따옴표
,	COMMA	커머	쉼표, 숨표
.	PERIOD	피어리어드	마침표, 소숫점
<	LESS THAN	레스 댄	왼쪽 부등호
>	GREATER THAN	모어 댄	오른쪽 부등호
<>	ANGLE BRACKETS	앵글 브레킷	부등호
/	SLASH	슬래쉬	사선, 빗금, 나누기
?	QUESTION MARK	퀘스천 마크	물음표
공백	SPACE	스페이스	공백, 여백

& 앰퍼샌드 (AMPERSAND) a.k.a 앤드 표시

*** 애스터리스크 (ASTERISK)** a.k.a 별 표시

\ (₩) 백슬래쉬 (BACKSLASH) a.k.a 원 표시, 돈 표시 => ₩

; **세미콜론 (SEMICOLON)** a.k.a 세미콜론

: **콜론 (COLON)** a.k.a 땡땡

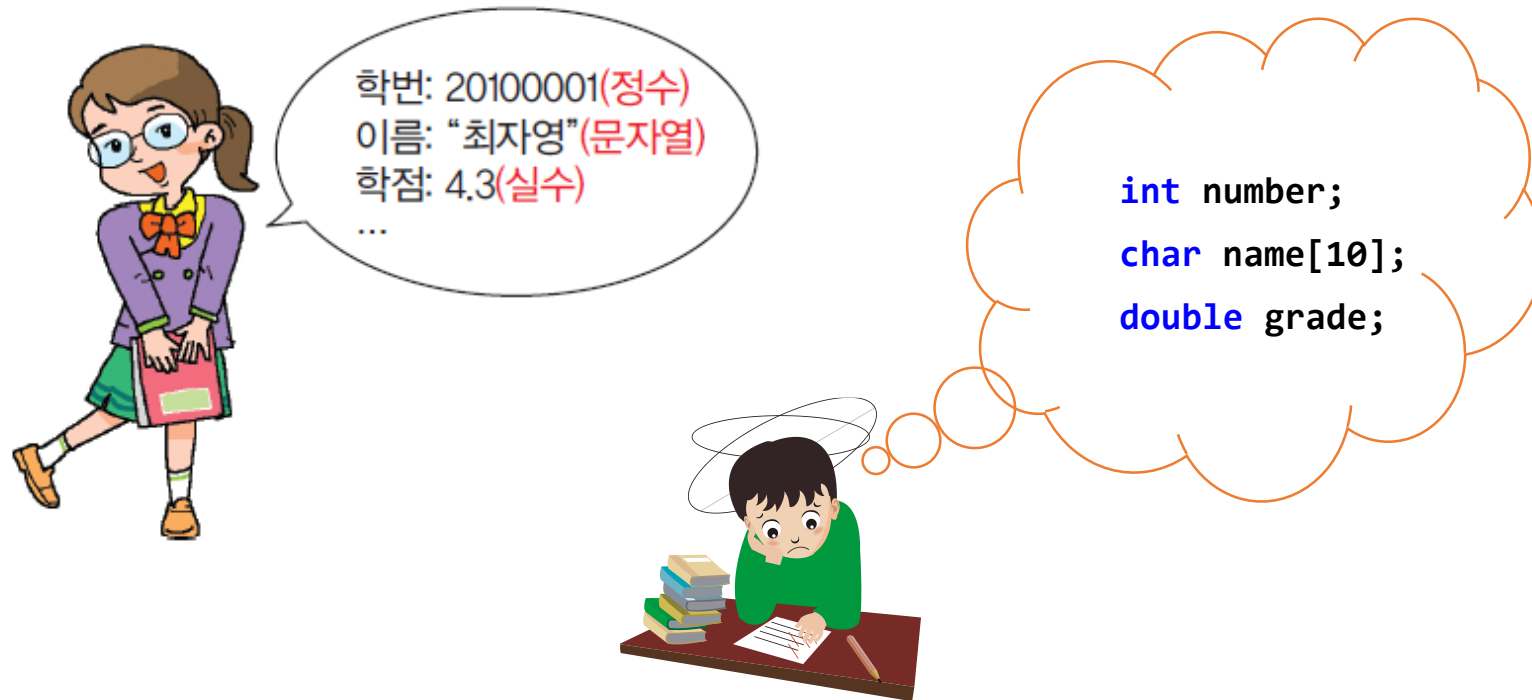
, **커머 (COMMA)** a.k.a 콤마, 쉼표

	필드 ↓	필드 ↓	필드 ↓
레코드 →	number	name	grade
	20190001	김말숙	4.0
	20190002	최태영	4.0
	20190003	이영숙	4.0

2. 구조체 – 구조체의 필요성

■ 학생에 대한 데이터를 하나로 모으려면?

- 개별 변수로 나타낸 데이터를 묶을수 있을까?

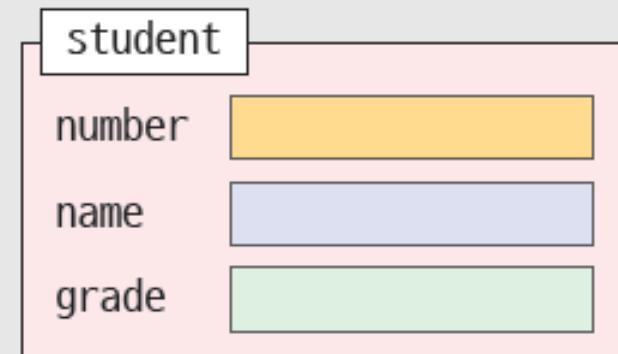


2. 구조체 – 구조체의 필요성

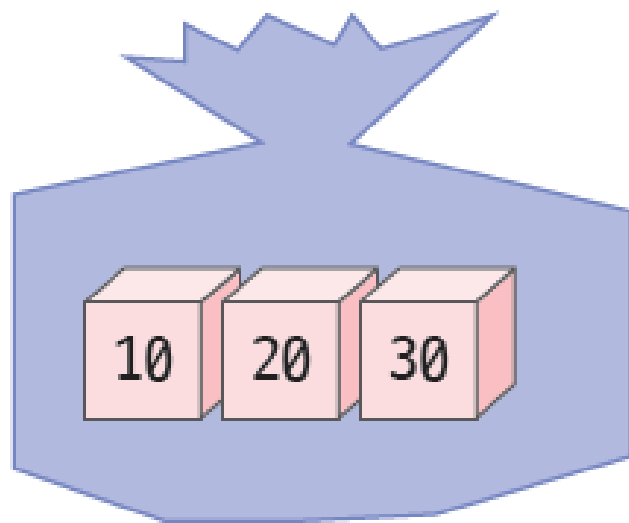
5 |

- 구조체를 사용하면 변수들을 하나로 묶을 수 있다

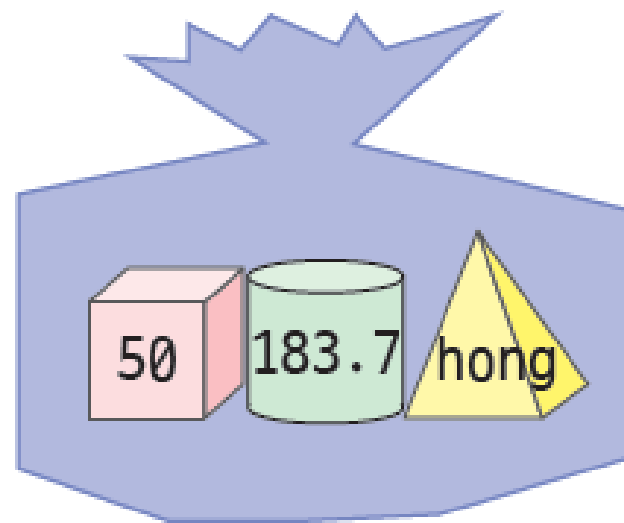
```
struct student {  
    int number;    // 학번  
    char name[10]; // 이름  
    double grade;  // 학점  
};
```



■ 구조체와 배열의 차이점



배열

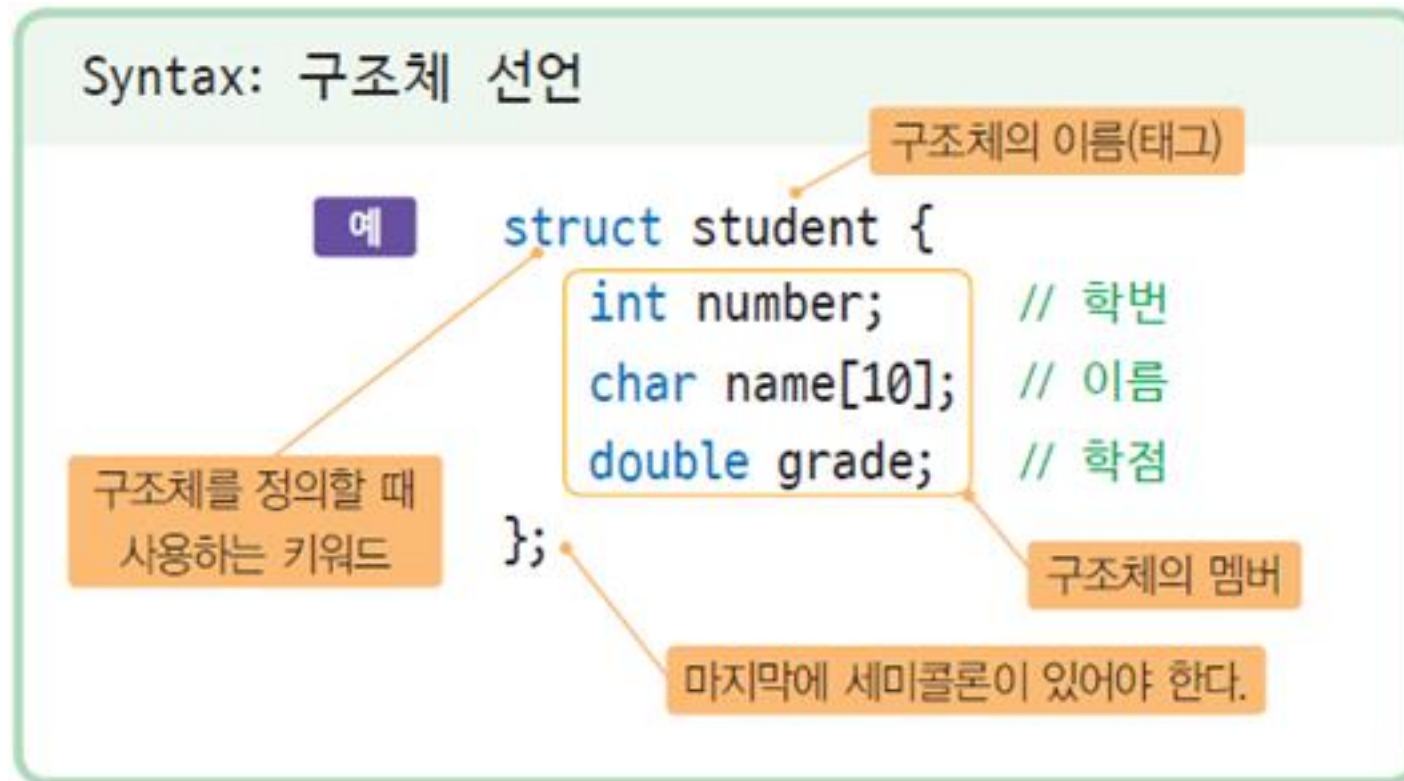


구조체

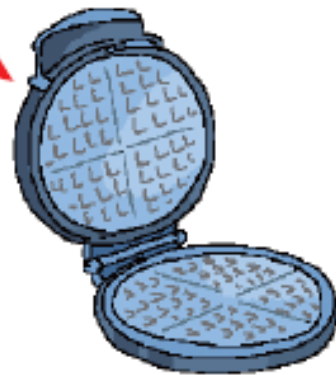
2. 구조체 – 구조체 선언

71

- 구조체를 사용하면 연관된 변수들을 묶을 수 있다



구조체를 정의하는 것은 와플이나 붕어빵을 만드는 틀을 정의하는 것과 같다.



구조체

와플이나 붕어빵을 실제로 만들기 위해서는 구조체 변수를 선언하여야 한다.



구조체 변수


```
// x값과 y값으로 이루어지는 화면의 좌표
struct point {
    int x;           // x 좌표
    int y;           // y 좌표
};
```

```
// 복소수
struct complex {
    double real;      // 실수부
    double imag;      // 허수부
};
```

```
// 날짜
struct date {
    int month;
    int day;
    int year;
};
```

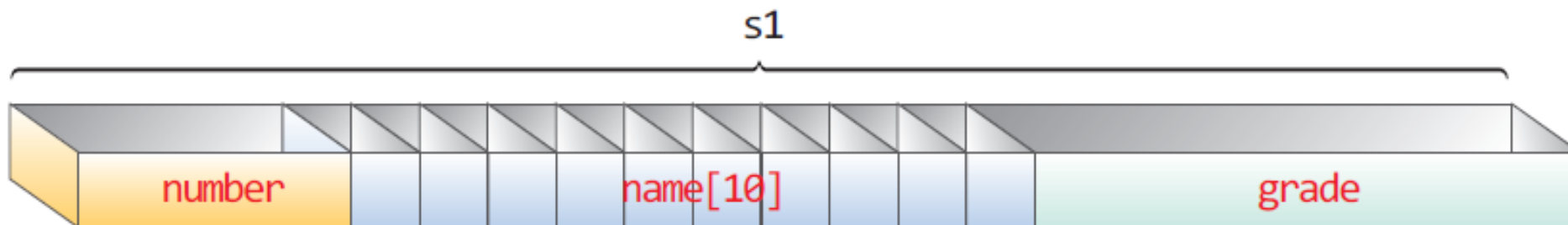
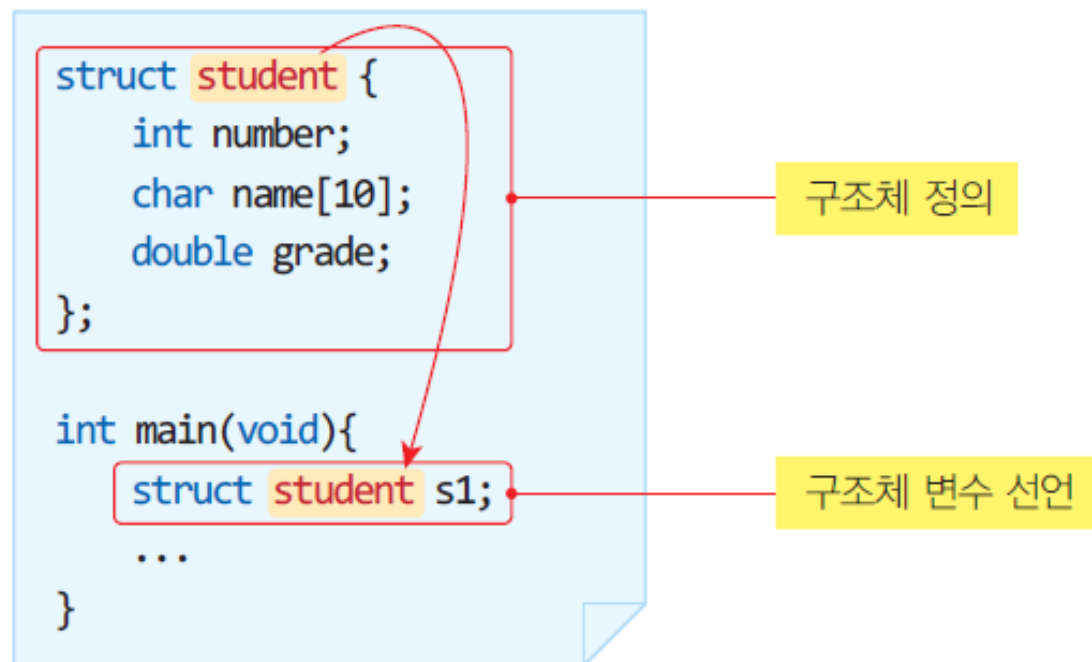
```
// 사각형
struct rect {
    int x;
    int y;
    int width;
    int grade;
};
```

```
// 직원
struct employee {
    char name[20];    // 이름
    int age;           // 나이
    int gender;        // 성별
    int salary;        // 월급
};
```

2. 구조체 – 구조체 변수 선언

10 |

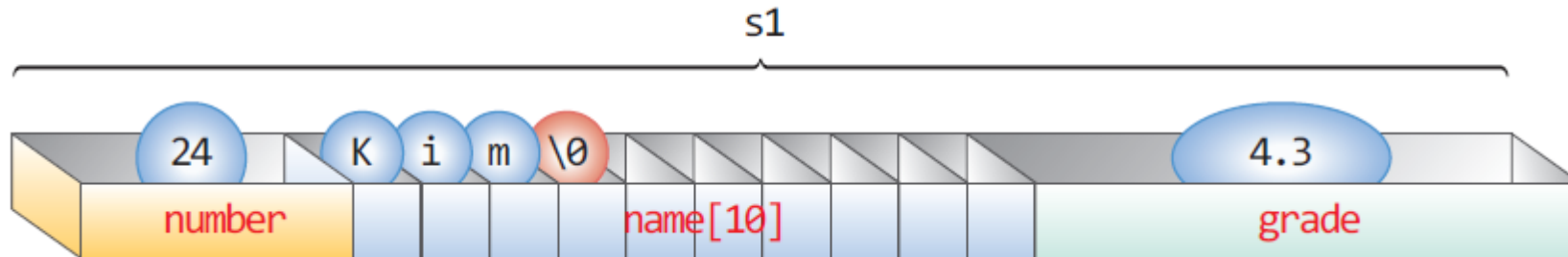
- 구조체 정의와 구조체 변수 선언은 다르다.



2. 구조체 – 구조체의 초기화

- 중괄호를 이용하여 초기값을 나열한다.
- 구조체형의 마지막은 반드시 '};'로 끝나야 함.

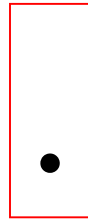
```
struct student {  
    int number;  
    char name[10];  
    double grade;  
};  
struct student s1 = { 24, "Kim", 4.3 };
```



Syntax: 구조체 멤버 접근

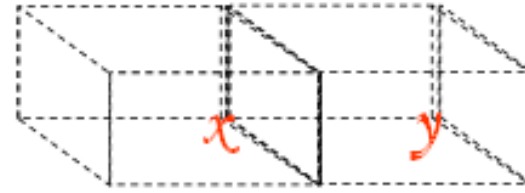
예 `s1.grade = 3.8;`

구조체 변수 구조체 멤버

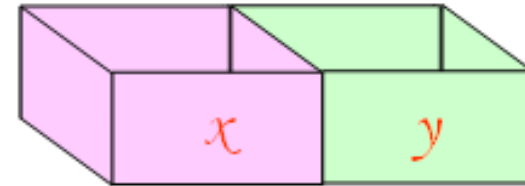


.기호는
구조체에서
멤버를 참조할
때 사용하는
연산자입니다.

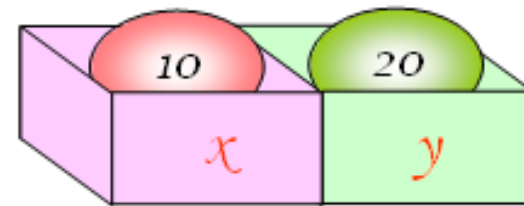
```
struct point {  
    int x;  
    int y;  
};
```



```
struct point p1;
```



```
p1.x = 10;  
p1.y = 20;
```



```
01 #include <stdio.h>
02 #include <string.h>
03
04 struct student {
05     int number;
06     char name[10];
07     double grade;
08 };
09
10 int main(void)
11 {
12     struct student s;
13
14     s.number = 20170001;
15     strcpy_s(s.name, sizeof(s.name), "홍길동");
16     s.grade = 4.3;
17
18     printf("학번: %d\n", s.number);
19     printf("이름: %s\n", s.name);
20     printf(" 학점: %f\n", s.grade);
21     return 0;
22 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C program: "학번: 20170001", "이름: 홍길동", and "학점: 4.300000". Below the output, there is a prompt message in Korean: "계속하려면 아무 키나 누르십시오 . . .".

```
01 #include <stdio.h>
02 // 2차원 공간의 점을 구조체로 나타낸다.
03 struct point {
04     int x;
05     int y;
06 };
07
08 int main(void)
09 {
10     struct point p = { 1, 2 };
11     struct point q = { .y = 2, .x = 1 };
12     struct point r = p;
13     r = (struct point) { 1, 2 };
14
15     printf("p=(%d, %d) \n", p.x, p.y);
16     printf("q=(%d, %d) \n", q.x, q.y);
17     printf("r=(%d, %d) \n", r.x, r.y);
18     return 0;
19 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of the C program: "p=(1, 2)", "q=(1, 2)", and "r=(1, 2)". Below the output, there is a prompt "계속하려면 아무 키나 누르십시오 . . ." (Press any key to continue).

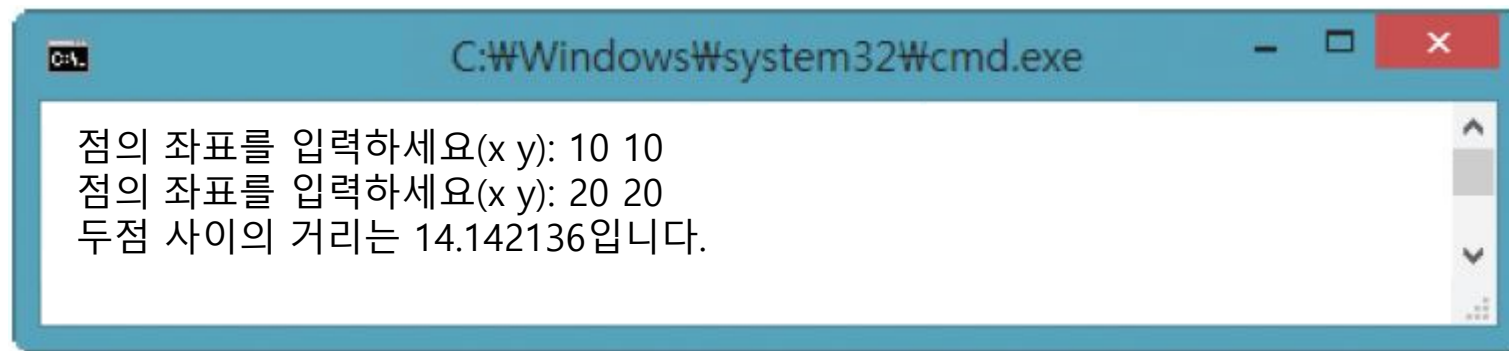
```
C:\Windows\system32\cmd.exe
p=(1, 2)
q=(1, 2)
r=(1, 2)
계속하려면 아무 키나 누르십시오 . . .
```

```
01 #include <stdio.h>
02
03 struct Book {                // 책의 정보를 저장할 구조체 선언
04     char title[50];          // 책 제목
05     char author[35];         // 저자
06     int pages;               // 페이지 수
07     int price;               // 가격
08 };
09
10 int main()
11 {
12     struct Book book1;        // 구조체 변수 선언
13     printf(" 도서 제목(string) : ");
14     gets(book1.title);        // 책 제목 입력, 구조체의 멤버변수 액세스 방법
15     printf(" 저자명(string) : ");
16     gets(book1.author);       // 저자 입력
17     printf(" 전체 페이지수(int) : ");
18     scanf_s("%d", &book1.pages); // 책 페이지 수 입력
19     printf(" 가격(int) : ");
20     scanf_s("%d", &book1.price); // 책값 입력
21
22     // 입력 받은 데이터 출력, 구조체의 멤버변수 액세스하여 출력.
23     printf("%s, %s, %dp, %d원\n", book1.title, book1.author, book1.pages, book1.price);
24 }
```


예제 설명

앞에서 정의한 point 구조체를 사용하여 2개의 점을 나타내는 변수를 선언한 후에 사용자로부터 점의 좌표를 입력받는다. 구조체 멤버의 값을 키보드로 입력할 수 있다. 이때 멤버 앞에 &을 붙여야 한다. 최종적으로 이들 사이의 거리를 계산하여 보자

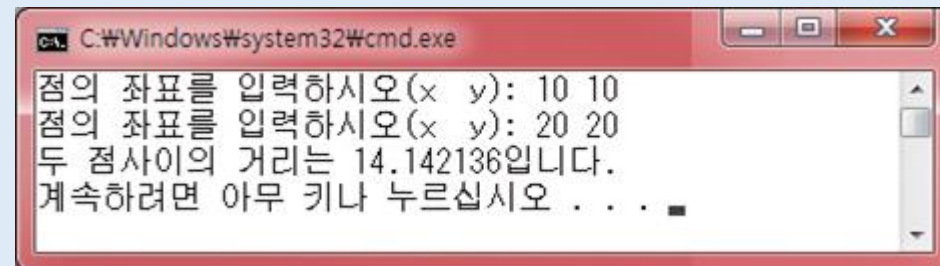
실행 결과



```
C:\Windows\system32\cmd.exe

점의 좌표를 입력하세요(x y): 10 10
점의 좌표를 입력하세요(x y): 20 20
두점 사이의 거리는 14.142136입니다.
```

```
01 #include <stdio.h>
02 #include <math.h>
03
04 struct point {
05     int x;
06     int y;
07 };
08
09 int main(void)
10 {
11     struct point p1, p2;
12     double xdiff, ydiff;
13     double dist;
14     printf("점의 좌표를 입력하시오(x y): ");
15     scanf_s("%d %d", &p1.x, &p1.y);
16
17     printf("점의 좌표를 입력하시오(x y): ");
18     scanf_s("%d %d", &p2.x, &p2.y);
19
20     xdiff = p1.x - p2.x;
21     ydiff = p1.y - p2.y;
22
23     dist = sqrt(xdiff * xdiff + ydiff * ydiff);
24     printf("두 점사이의 거리는 %f입니다.\n", dist);
25     return 0;
26 }
```



```
C:\Windows\system32\cmd.exe
점의 좌표를 입력하시오(x y): 10 10
점의 좌표를 입력하시오(x y): 20 20
두 점사이의 거리는 14.142136입니다.
계속하려면 아무 키나 누르십시오 . . .
```

■ 구조체 변수의 초기값 대입

```
char name[10] = "Kim";  
int kor = 90;  
int eng = 80;  
float avg;
```



```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
};  
struct student s = {"Kim", 90, 80};
```

- 구조체 대입은 가능하고 권장된다.

```
struct point {  
    int x;  
    int y;  
};  
struct point p1 = {10, 20};  
struct point p2 = {30, 40};  
  
p2 = p1;
```

구조체 변수끼리의 대입 연산은 가능하다.

- 구조체 비교는 불가능하다.

```
struct point {  
    int x;  
    int y;  
};  
struct point p1 = {10, 20};  
struct point p2 = {30, 40};  
  
if( p1 == p2 )           // 컴파일 오류  
{  
    printf("p1와 p2이 같습니다.")  
}
```

- 같은 구조체 변수끼리 대입은 가능하지만 비교는 불가능하다.

```
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point p1 = {10, 20};
    struct point p2 = {30, 40};

    p2 = p1;                                // 대입 가능

    if( p1 == p2 )                          // 비교 -> 컴파일 오류!!
        printf("p1와 p2이 같습니다.");

    if( (p1.x == p2.x) && (p1.y == p2.y) ) // 올바른 비교
        printf("p1와 p2이 같습니다.");
}
```

예제 설명 사각형을 구조체로 표현하고 사각형의 면적을 계산해보자.

실행 결과

A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The window contains three lines of text: "왼쪽 상단의 좌표를 입력하시오: 1 1", "오른쪽 상단의 좌표를 입력하시오: 6 6", and "면적은 25이고 둘레는 20입니다."

[예제] 점들간의 거리 계산

```

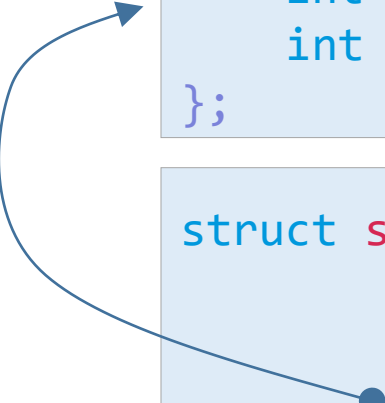
01 #include <stdio.h>
02
03 struct point {
04     int x;
05     int y;
06 };
07
08 struct rect {
09     struct point p1;
10     struct point p2;
11 };
12
13 int main(void)
14 {
15     struct rect r;
16     int w, h, area, peri;
17
18     printf("왼쪽 상단의 좌표를 입력하시오: ");
19     scanf_s("%d %d", &r.p1.x, &r.p1.y);
20     printf("오른쪽 상단의 좌표를 입력하시오: ");
21     scanf_s("%d %d", &r.p2.x, &r.p2.y);
22
23     w = r.p2.x - r.p1.x;
24     h = r.p2.y - r.p1.y;
25     area = w * h;
26     peri = 2 * w + 2 * h;
27     printf("면적은 %d이고 둘레는 %d입니다.\n", area, peri);
28     return 0;
29 }

```



왼쪽 상단의 좌표를 입력하시오: 1 1
 오른쪽 상단의 좌표를 입력하시오: 6 6
 면적은 25이고 둘레는 20입니다.


```
struct date {                      // 구조체 선언
    int year;
    int month;
    int day;
};
```



```
struct student {                   // 구조체 선언
    int number;
    char name[10];
    struct date dob;               // 구조체 안에 구조체 포함
    double grade;
};
struct student s1;                 // 구조체 변수 선언
```

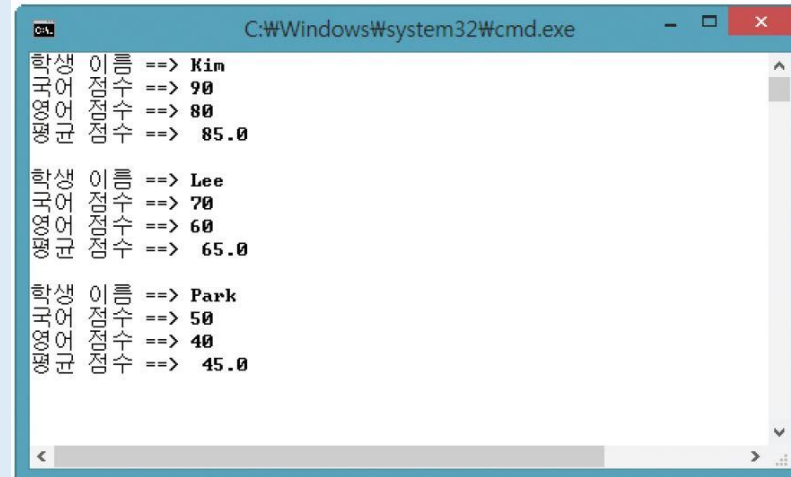
```
s1.dob.year = 1983;                // 멤버 참조
s1.dob.month = 03;
s1.dob.day = 29;
```

2. 구조체 – 구조체 배열을 사용하지 않은 예제

26 |

```
01 #include <stdio.h>
02 #include <string.h>
03
04 void main( ) {
05     char name[3][10];      ---학생 이름 배열을 선언한다.
06     int i, kor[3], eng[3];  ---국어, 영어점수 배열을 선언한다.
07     float avg[3];          ---평균 점수 배열을 선언한다.
08
09     strcpy_s(name[0], sizeof(name[0]), "Kim");  ---첫 번째 학생의 정보를 대입한다(학생 이름, 국어 점수,
10     kor[0] = 90;                                     영어 점수, 평균 점수).
11     eng[0] = 80;
12     avg[0] = (kor[0] + eng[0]) / 2.0f;
13
14     strcpy_s(name[1], sizeof(name[1]), "Lee");  ---두 번째 학생의 정보를 대입한다(학생 이름, 국어 점수,
15     kor[1] = 70;                                     영어 점수, 평균 점수).
16     eng[1] = 60;
17     avg[1] = (kor[1] + eng[1]) / 2.0f;
18
19     strcpy_s(name[2], sizeof(name[2]), "Park");  ---세 번째 학생의 정보를 대입한다(학생 이름, 국어 점수,
20     kor[2] = 50;                                     영어 점수, 평균 점수).
21     eng[2] = 40;
22     avg[2] = (kor[2] + eng[2]) / 2.0f;
23
24     for(i=0; i<3; i++)  ---세 번 반복하고 배열의 내용을 출력한다.
25     {
26         printf("학생 이름 == > %s\n", name[i]);
27         printf("국어 점수 == > %d\n", kor[i]);
28         printf("영어 점수 == > %d\n", eng[i]);
29         printf("평균 점수 == > %5.1f\n", avg[i]);
30         printf("\n");
31     }
32 }
```

실행 결과 ▼

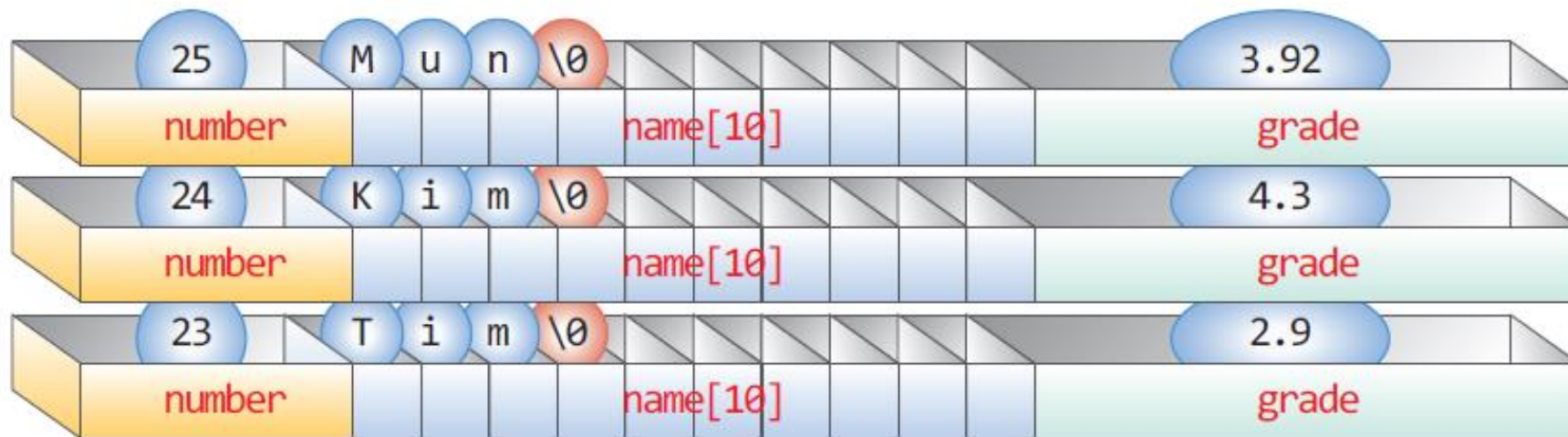


```
C:\Windows\system32\cmd.exe
학생 이름 ==> Kim
국어 점수 ==> 90
영어 점수 ==> 80
평균 점수 ==> 85.0
학생 이름 ==> Lee
국어 점수 ==> 70
영어 점수 ==> 60
평균 점수 ==> 65.0
학생 이름 ==> Park
국어 점수 ==> 50
영어 점수 ==> 40
평균 점수 ==> 45.0
```

■ 구조체를 여러 개 모은것

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

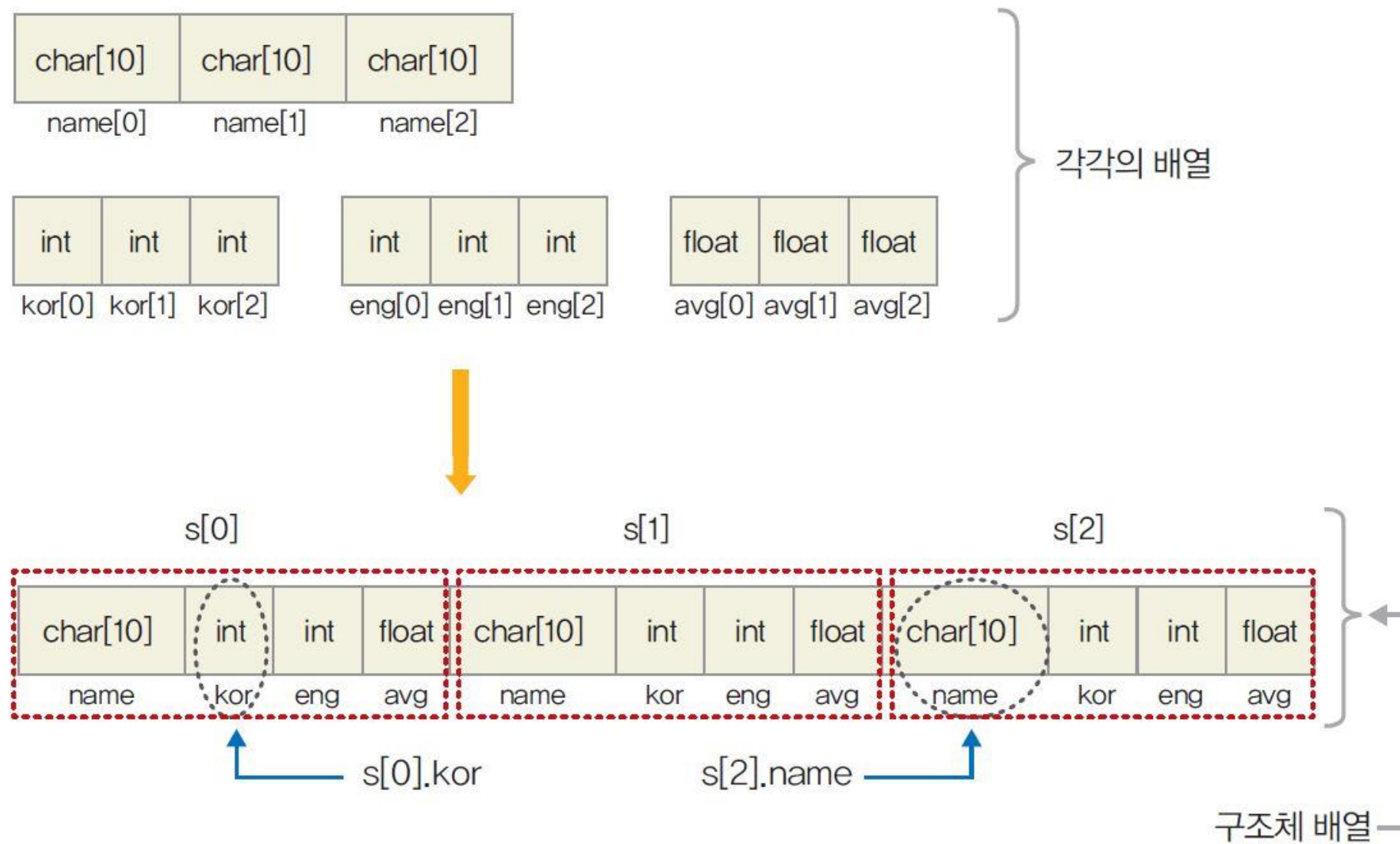
```
struct student list[100];    // 구조체 배열 선언
```



```
01 struct student {
02     int number;
03     char name[20];
04     double height;
05 };
06
07 int main(void)
08 {
09     struct student list[100];    // 구조체의 배열 선언
10
11     list[2].number = 27;
12     strcpy_s(list[2].name, sizeof(list[2].name), "홍길동");
13     list[2].height = 178.0;
14 }
```

```
struct student list[3] = {
    { 1, "Park", 172.8 },
    { 2, "Kim", 179.2 },
    { 3, "Lee", 180.3 }
};
```

2. 구조체 – 구조체 배열



2. 구조체 – 구조체 배열을 사용하여 변경 예제

```
01 #include <stdio.h>
02 #include <string.h>
03
04 int main( )
05 {
06     struct student {
07         char name[10];
08         int kor;
09         int eng;
10         float avg;
11     };
12
13     struct student s[3];
14
15     int i;
16
17     strcpy_s(s[0].name, sizeof(s[0].name), "Kim");
18     s[0].kor = 90;
19     s[0].eng = 80;
20     s[0].avg = (s[0].kor + s[0].eng) / 2.0f;
```

----구조체형을 선언한다.

----구조체 배열 s[3]을 선언한다.

----첫 번째 학생의 정보를 대입한다
(학생 이름, 국어 점수, 영어 점수, 평균 점수).

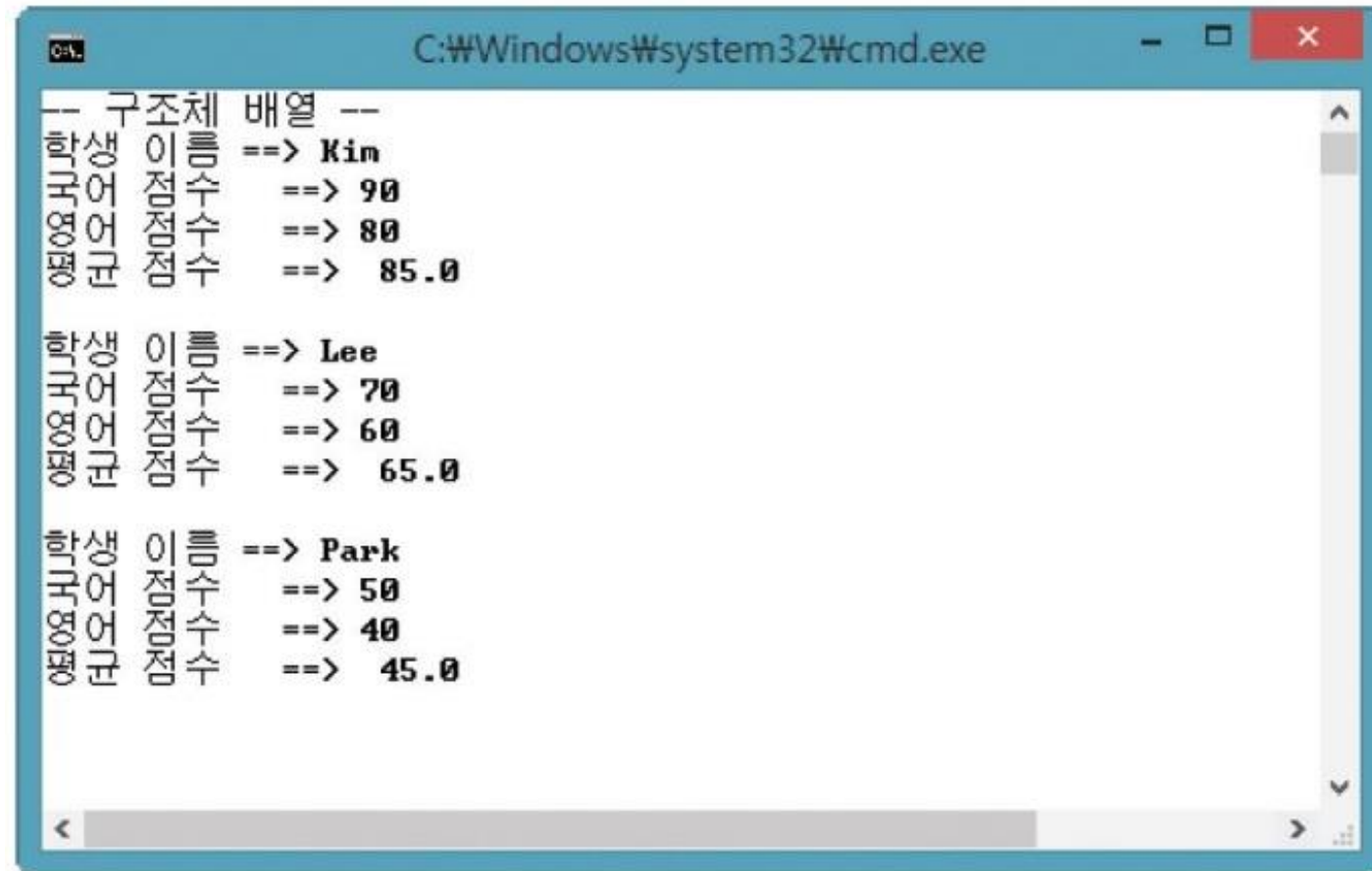
```
21
22     strcpy_s( s[1].name, sizeof(s[1].name), "Lee");
23     s[1].kor = 70;
24     s[1].eng = 60;
25     s[1].avg = (s[1].kor + s[1].eng) / 2.0f;
26
27     strcpy_s(s[2].name, sizeof(s[2].name), "Park");
28     s[2].kor = 50;
29     s[2].eng = 40;
30     s[2].avg = (s[2].kor + s[2].eng) / 2.0f;
31
32     printf("-- 구조체 배열 -- \n");
33     for(i=0; i<3; i++)
34     {
35         printf("학생 이름 == > %s\n", s[i].name);
36         printf("국어 점수 == > %d\n", s[i].kor);
37         printf("영어 점수 == > %d\n", s[i].eng);
38         printf("평균 점수 == > %5.1f\n", s[i].avg);
39         printf("\n");
40     }
41 }
```

----두 번째 학생의 정보를 대입한다.

----세 번째 학생의 정보를 대입한다.

----세 번 반복하고 구조체 배열의 내용을 출력한다.

실행 결과 ▼



```
C:\Windows\system32\cmd.exe

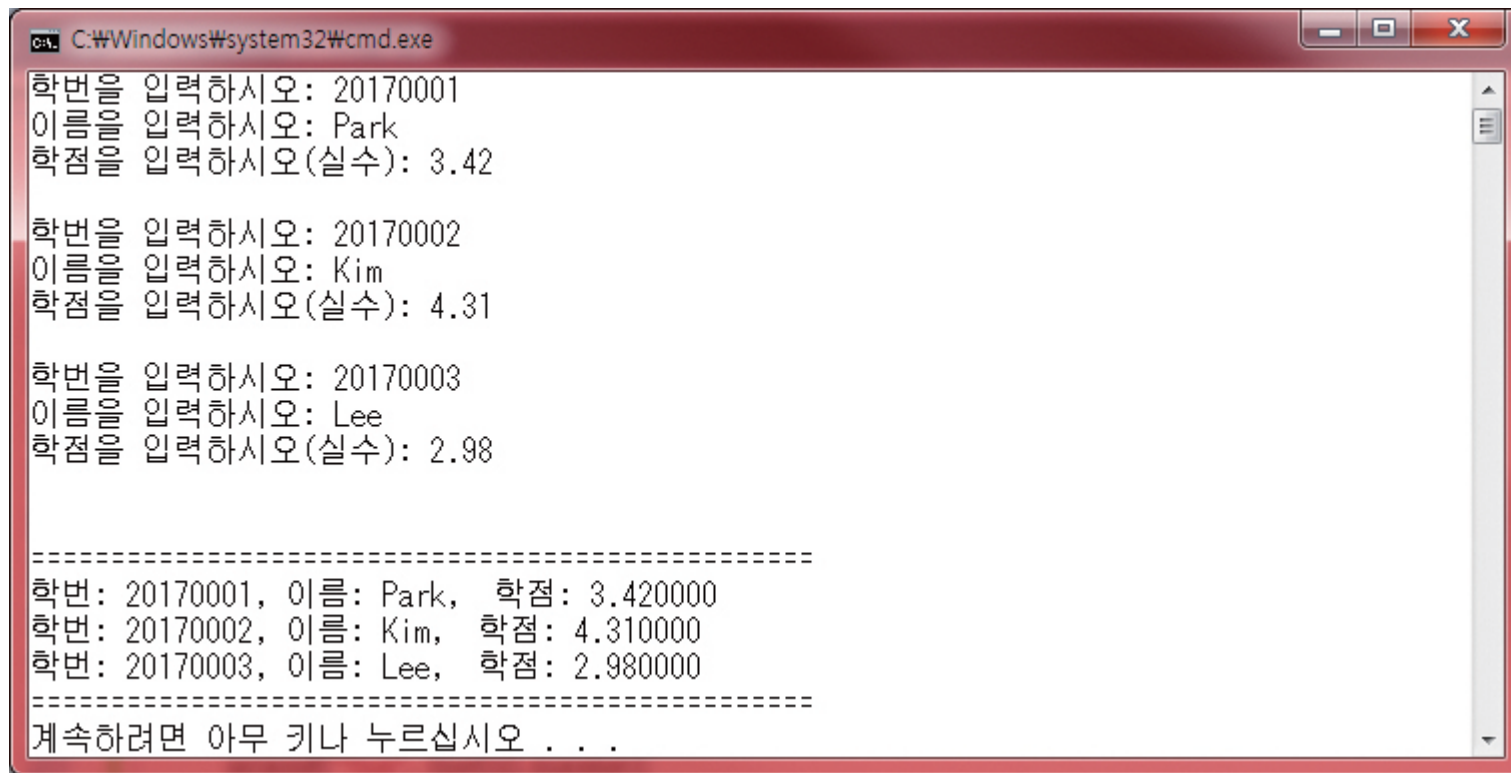
-- 구조체 배열 --
학생 이름 ==> Kim
국어 점수 ==> 90
영어 점수 ==> 80
평균 점수 ==> 85.0

학생 이름 ==> Lee
국어 점수 ==> 70
영어 점수 ==> 60
평균 점수 ==> 65.0

학생 이름 ==> Park
국어 점수 ==> 50
영어 점수 ==> 40
평균 점수 ==> 45.0
```

```
01 #include <stdio.h>
02
03 struct score { // 구조체 정의. struct 구조체명
04     char name[30];
05     int kor;
06     int eng;
07     int math;
08     int total;
09     float avr;
10 };
11
12 int main()
13 {
14     struct score Class[3] = { {"강아지", 95, 87, 97, },
15                               { "박기동", 99, 89, 80, },
16                               { "홍길동", 88, 77, 66, } }; // 구조체 배열 선언 및 초기화
17     int i;
18     for (i=0; i<3; i++) {
19         Class[i].total = Class[i].kor + Class[i].eng + Class[i].math;
20         Class[i].avr = (float)Class[i].total/3.;
21
22         printf("이름 : %s, 국어 : %d, 영어 : %d, 수학: %d, ", Class[i].name, Class[i].kor, Class[i].eng, Class[i].math);
23         printf("총점 : %d, 평균 : %5.1f\n\n", Class[i].total, Class[i].avr);
24     }
25 }
```


- 학생들의 데이터를 반복 구조를 사용하여 입력 받는다. 데이터들은 구조체의 배열에 저장된다.



```
C:\Windows\system32\cmd.exe

학번을 입력하시오: 20170001
이름을 입력하시오: Park
학점을 입력하시오(실수): 3.42

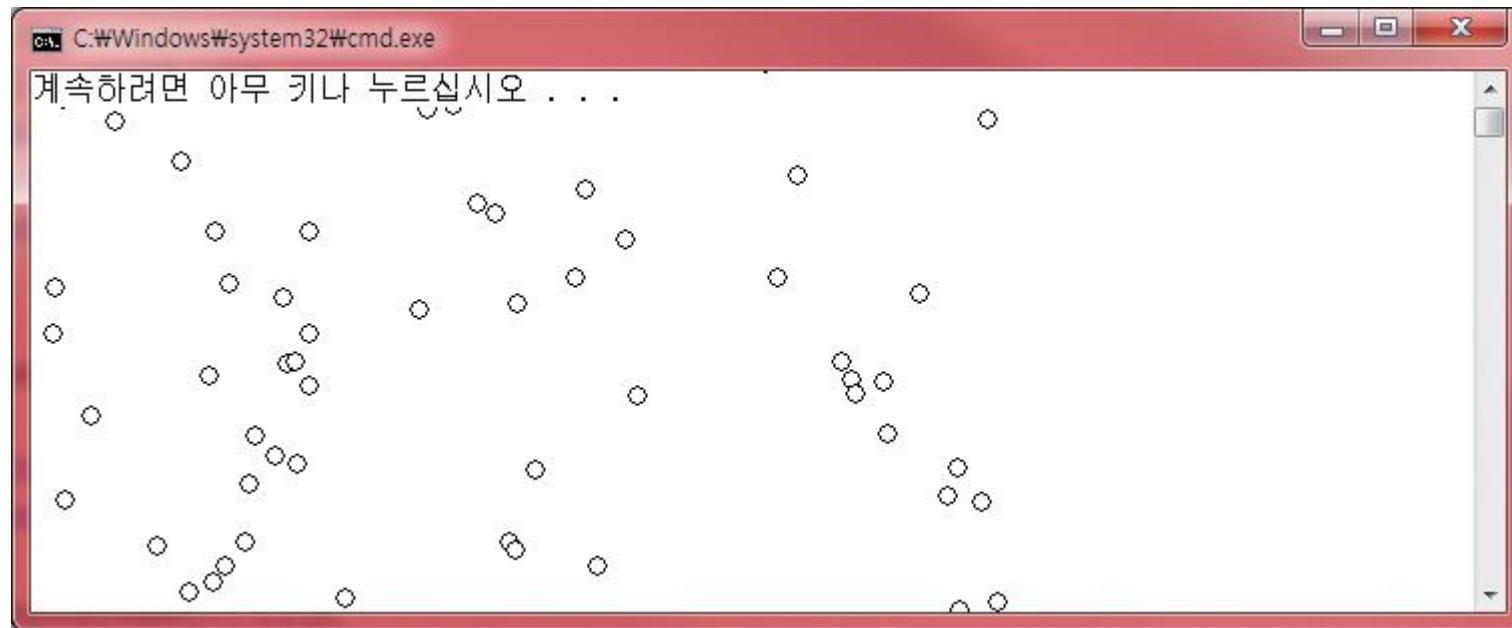
학번을 입력하시오: 20170002
이름을 입력하시오: Kim
학점을 입력하시오(실수): 4.31

학번을 입력하시오: 20170003
이름을 입력하시오: Lee
학점을 입력하시오(실수): 2.98

=====
학번: 20170001, 이름: Park, 학점: 3.420000
학번: 20170002, 이름: Kim, 학점: 4.310000
학번: 20170003, 이름: Lee, 학점: 2.980000
=====
계속하려면 아무 키나 누르십시오 . . .
```

```
01 #include <stdio.h>
02 #define SIZE 3
03
04 struct student {
05     int number;
06     char name[20];
07     double grade;
08 };
09
10 int main(void)
11 {
12     struct student list[SIZE];
13     int i;
14
15     for (i = 0; i < SIZE; i++){
16         printf("학번을 입력하시오: ");
17         scanf_s("%d", &list[i].number);
18         printf("이름을 입력하시오: ");
19         scanf_s("%s", list[i].name, sizeof(list[i].name));
20         printf("학점을 입력하시오(실수): ");
21         scanf_s("%lf", &list[i].grade);
22         printf("\n");
23     }
24     printf("\n===== \n");
25     for (i = 0; i < SIZE; i++)
26         printf("학번: %d, 이름: %s, 학점: %f\n", list[i].number, list[i].name, list[i].grade);
27     printf("===== \n");
28     return 0;
29 }
```

- 2차원 공간의 점을 구조체로 표현하고 크기가 100인 구조체 배열을 선언한다. 여기에 난수를 저장한 후에 하나씩 꺼내서 화면에 점을 그려보자.



```
01 #include <windows.h>
02 #include <stdlib.h>
03 #include <time.h>
04 #define SIZE 100
05 struct point {
06     int x;
07     int y;
08 };
09
10 int main(void)
11 {
12     struct point p[SIZE];
13     int i;
14     srand((unsigned)time(NULL));
15     for (int i = 0; i < SIZE; i++) {
16         p[i].x = rand() % 500;
17         p[i].y = rand() % 500;
18     }
19     HDC hdc = GetWindowDC(GetForegroundWindow());
20     for (int i = 0; i < SIZE; i++)
21         Ellipse(hdc, p[i].x, p[i].y, p[i].x + 10, p[i].y + 10);
22     return 0;
23 }
```

■ 구조체를 함수의 인수로 전달하는 경우

- 구조체의 복사본이 함수로 전달되게 된다.
- 만약 구조체의 크기가 크면 그만큼 시간과 메모리가 소요된다.

구조체의 경우, 복사된다.

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

```
int main(void)
{
    struct student a = { 1, "hong", 3.8 };
    struct student b = { 2, "kim", 4.0 };
    if( equal(a, b) == 1 ){
        printf("같은 학생 \n");
    }
    else {
        printf("다른 학생 \n");
    }
}
```

■ 구조체의 포인터를 함수의 인수로 전달하는 경우

- 시간과 공간을 절약할 수 있다.
- 원본 훼손의 가능성이 있다.

구조체 포인터를 보낸다.

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

포인터를 통하여 구조체에 접근한다.

```
int main(void)
{
    struct student a = { 1, "hong", 3.8 };
    struct student b = { 2, "kim", 4.0 };
    if( equal(&a, &b) == 1 ){
        printf("같은 학생 \n");
    }
    else {
        printf("다른 학생 \n");
    }
}
```

```
01 #include <stdio.h>
02
03 struct SHuman
04 {
05     char name[12];
06     int age;
07     double height;
08 };
09
10 void outHuman(struct SHuman some)
11 {
12     printf("%s : %d세, 키 %.2f\n", some.name, some.age, some.height);
13 }
14
15 int main()
16 {
17     struct SHuman hong = { "홍길동", 19, 179.8 };
18     outHuman(hong);
19 }
```

홍길동 : 19세, 키 179.80

- 복사본이 반환된다.
- 구조체가 인수나 반환값으로 사용될 때는 "값에 의한 호출" 원칙이 적용된다.

```
struct student create()
{
    struct student s;
    s.number = 3;
    strcpy(s.name, "park");
    s.grade = 4.0;
    return s;
}
```

구조체 s가 구조체 a로 복사된다.

```
int main(void)
{
    struct student a;
    a = create();
    return 0;
}
```



```
#include <stdio.h>

struct SHuman
{
    char name[12];
    int age;
    double height;
};

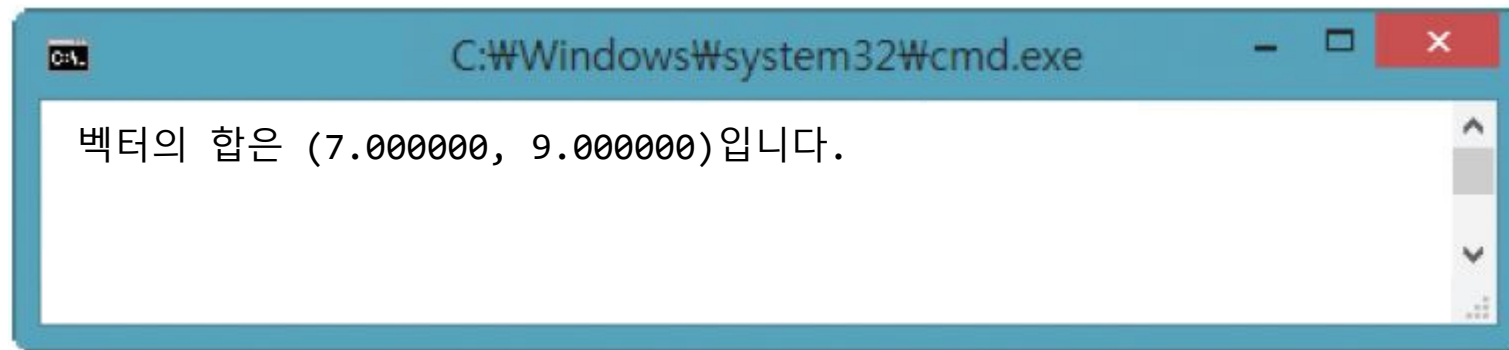
struct SHuman findHuman()
{
    struct SHuman some = { "홍길동", 19, 179.8 };
    return some;
}

int main()
{
    struct SHuman hong;
    hong = findHuman();
    printf("%s : %d세, 키 %.2f\n", hong.name, hong.age, hong.height);
}
```

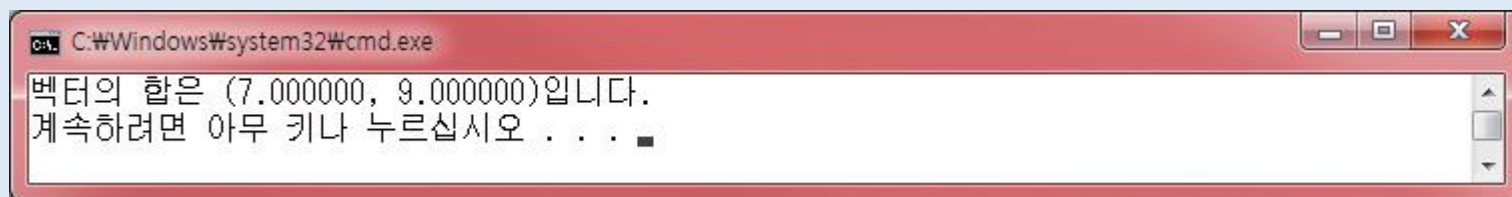
홍길동 : 19세, 키 179.80

예제 설명 두 벡터의 합을 구하는 함수 `get_vector_sum()`를 제작하여 보자

실행 결과



```
01 #include <stdio.h>
02
03 struct vector {
04     float x;
05     float y;
06 };
07 struct vector get_vector_sum(struct vector a, struct vector b);
08
09 int main(void)
10 {
11     struct vector a = { 2.0, 3.0 };
12     struct vector b = { 5.0, 6.0 };
13     struct vector sum;
14
15     sum = get_vector_sum(a, b);
16     printf("벡터의 합은 (%f, %f)입니다.\n", sum.x, sum.y);
17     return 0;
18 }
19 struct vector get_vector_sum(struct vector a, struct vector b)
20 {
21     struct vector result;
22     result.x = a.x + b.x;
23     result.y = a.y + b.y;
24     return result;
25 }
```



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The command prompt shows the output of the program: "벡터의 합은 (7.000000, 9.000000)입니다." followed by a prompt "계속하려면 아무 키나 누르십시오 . . .".

2. 구조체 – 구조체 변수의 다른 방법 선언

- 구조체 변수의 다른 선언 방법

❶ 구조체형과 변수를 별도로 선언

```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
};
```

```
struct student s;
```

❷ 구조체형과 변수를 동시에 선언

```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
} s;
```

❸ typedef를 이용하여 선언

```
typedef struct _student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
} student;  
student s;
```

2. 구조체 – 구조체와 포인터

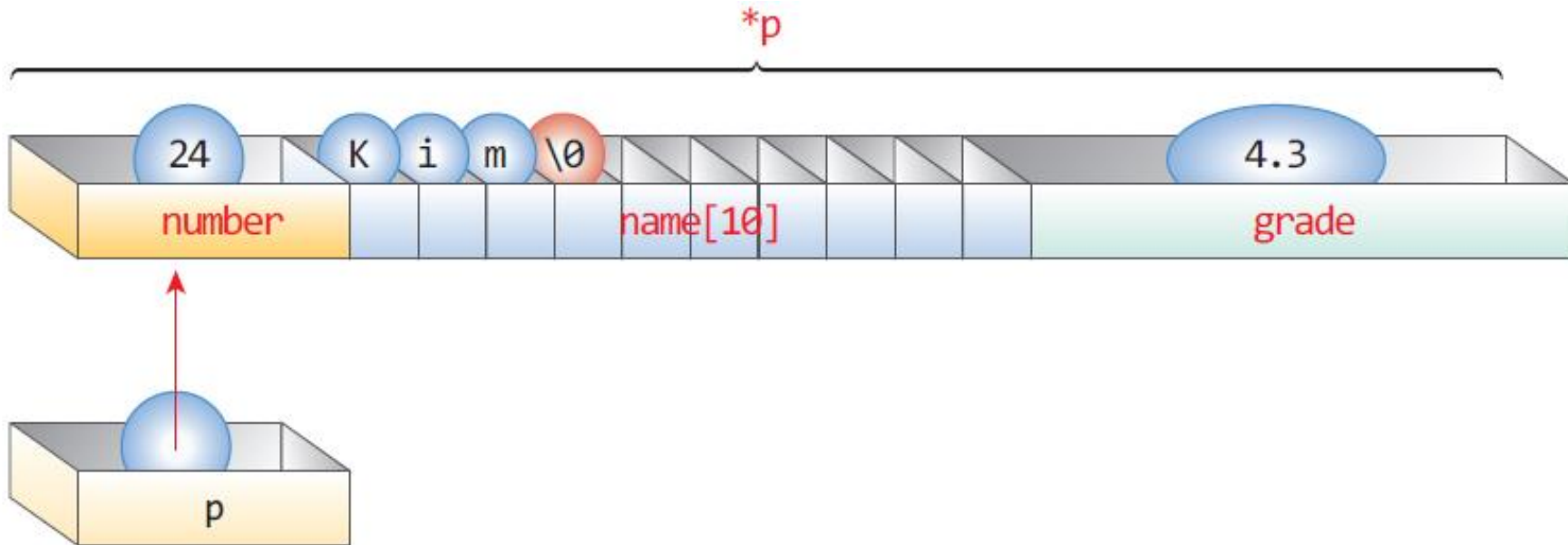
- 구조체 포인터 : 구조체 포인터 변수도 구조체의 주소값을 가진다.



```
struct student s = { 20070001, "홍길동", 4.3 };  
struct student *p;
```

```
p = &s;
```

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



2. 구조체 – 구조체와 포인터 : -> 연산자

- -> 연산자는 구조체 포인터로 구조체 멤버를 참조할 때 사용

```
struct student s = { 24, "Kim", 4.3 };  
struct student *p;
```

```
p = &s;
```

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

p가 가리키는 구조체 변수

`(*p).number`

p가 가리키는 구조체 변수의 멤버 number

`p->number`

p가 가리키는 구조체 변수의 멤버 number

2. 구조체 – 구조체와 포인터

- 일반 포인터 변수와 구조체 포인터 변수를 사용하는 경우

❶ 일반 포인터 변수 사용

```
int a;  
int* p;  
p = &a;  
*p = 100;
```

❷ 구조체 포인터 변수 사용

```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
};  
struct student s;  
struct student *p;  
p = &s;  
p->kor = 100;
```

```
01 #include <stdio.h>
02
03 int main( )
04 {
05     struct student {
06         char name[10];
07         int kor;
08         int eng;
09         float avg;
10     };
11
12     struct student s;
13     struct student *p;
14
15     p = &s;
16
17     printf("이름 입력 : ");
18     scanf_s("%s", p->name, sizeof(p->name));
19
20     printf("국어 점수 : ");
21     scanf_s("%d", &p->kor); //&s.kor
22
23     printf("영어 점수 : ");
24     scanf_s("%d", &p->eng); //&s.eng
25
26     p->avg = (p->kor + p->eng) / 2.0f;
27
28     printf("\n--- 구조체 포인터 활용 ---\n");
29     printf("학생 이름 == > %s\n", p->name);
30     printf("국어 점수 == > %d\n", p->kor);
31     printf("영어 점수 == > %d\n", p->eng);
32     printf("평균 점수 == > %5.1f\n", p->avg);
33 }
```

---student 구조체형을 선언한 후 멤버 변수를 선언한다
(학생 이름, 국어 점수, 영어 점수, 수학 점수, 평균 점수).

---구조체 변수 s를 선언한다.

---구조체 포인터 변수 p를 선언한다.

---포인터 p에 s의 주소를 대입한다.

---이름을 입력한다.

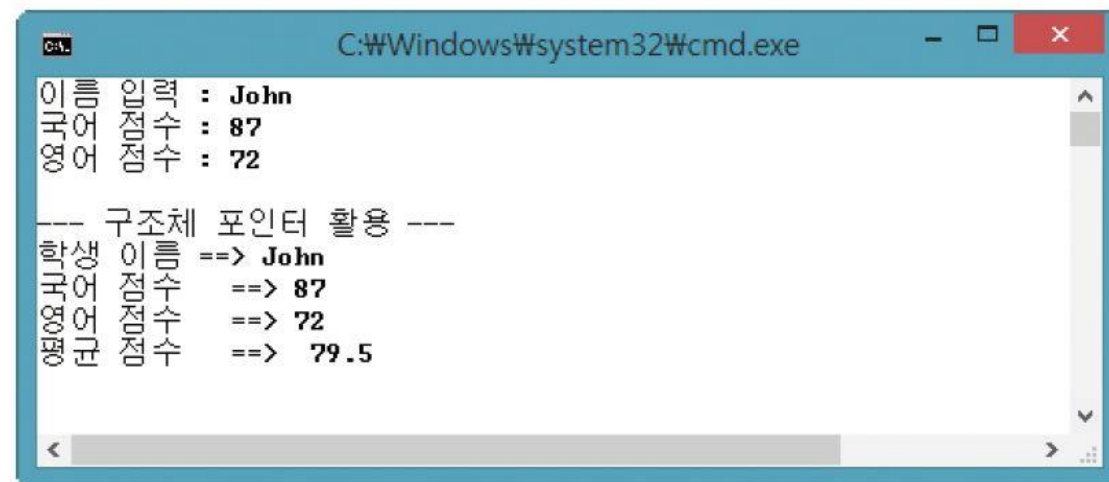
---국어 점수를 입력한다.

---영어 점수를 입력한다.

---평균 점수를 계산한다.

---학생 이름을 출력한다.
국어 점수를 출력한다.
영어 점수를 출력한다.
평균 점수를 출력한다.

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
이름 입력 : John
국어 점수 : 87
영어 점수 : 72

--- 구조체 포인터 활용 ---
학생 이름 ==> John
국어 점수 ==> 87
영어 점수 ==> 72
평균 점수 ==> 79.5
```

```
#include <stdio.h>

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

int main(void)
{
    struct student s = { 20070001, "홍길동", 4.3 };
    struct student *p;

    p = &s;
    printf("학번=%d 이름=%s 학점=%f \n", s.number, s.name, s.grade);
    printf("학번=%d 이름=%s 학점=%f \n", (*p).number, (*p).name, (*p).grade);
    printf("학번=%d 이름=%s 학점=%f \n", p->number, p->name, p->grade);
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
학번=20070001 이름=홍길동 학점=4.300000
학번=20070001 이름=홍길동 학점=4.300000
학번=20070001 이름=홍길동 학점=4.300000
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>
```

```
struct SHuman
```

```
{
```

```
    char name[12];
```

```
    int age;
```

```
    double height;
```

```
};
```

```
int main()
```

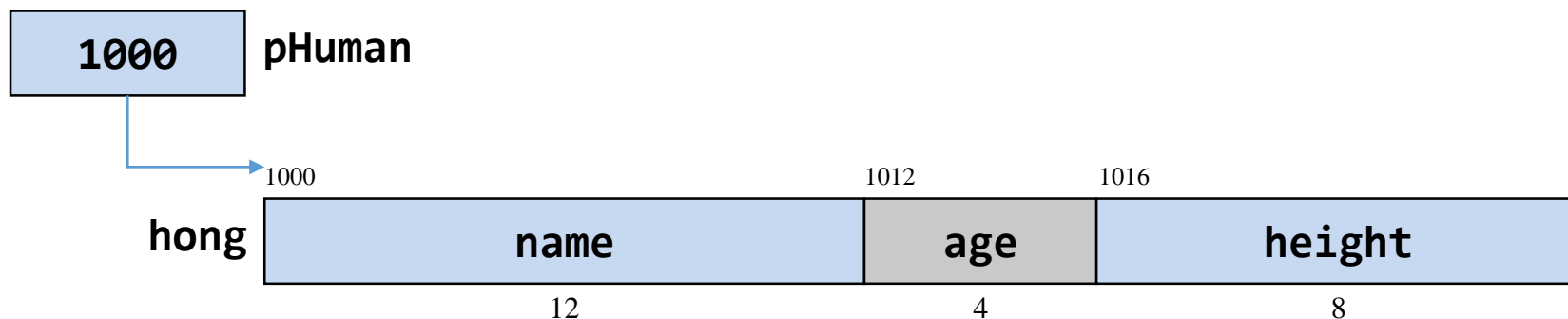
```
{
```

```
    struct SHuman hong = { "홍길동", 19, 179.8 };
```

```
    struct SHuman *pHuman = &hong;
```

```
    printf("나이 = %d\n", (*pHuman).age);
```

```
}
```



나이 = 19

```
01  #include <stdio.h>
02
03  struct SHuman
04  {
05      char name[12];
06      int age;
07      double height;
08  };
09
10  void outHuman(struct SHuman *pSome)
11  {
12      printf("%s : %d세, 키 %.2f\n", pSome->name, pSome->age, pSome->height);
13      pSome->age = 40;
14  }
15
16  int main()
17  {
18      struct SHuman hong = { "홍길동", 19, 179.8 };
19      outHuman(&hong);
20      printf("나이 = %d\n", hong.age);
21  }
```

홍길동 : 19세, 키 179.80
나이 = 40

```
#include <stdio.h>
#include <malloc.h>

struct SHuman
{
    char name[12];
    int age;
    double height;
};

int main()
{
    struct SHuman *pJuso[5]; // 구조체 포인터 배열

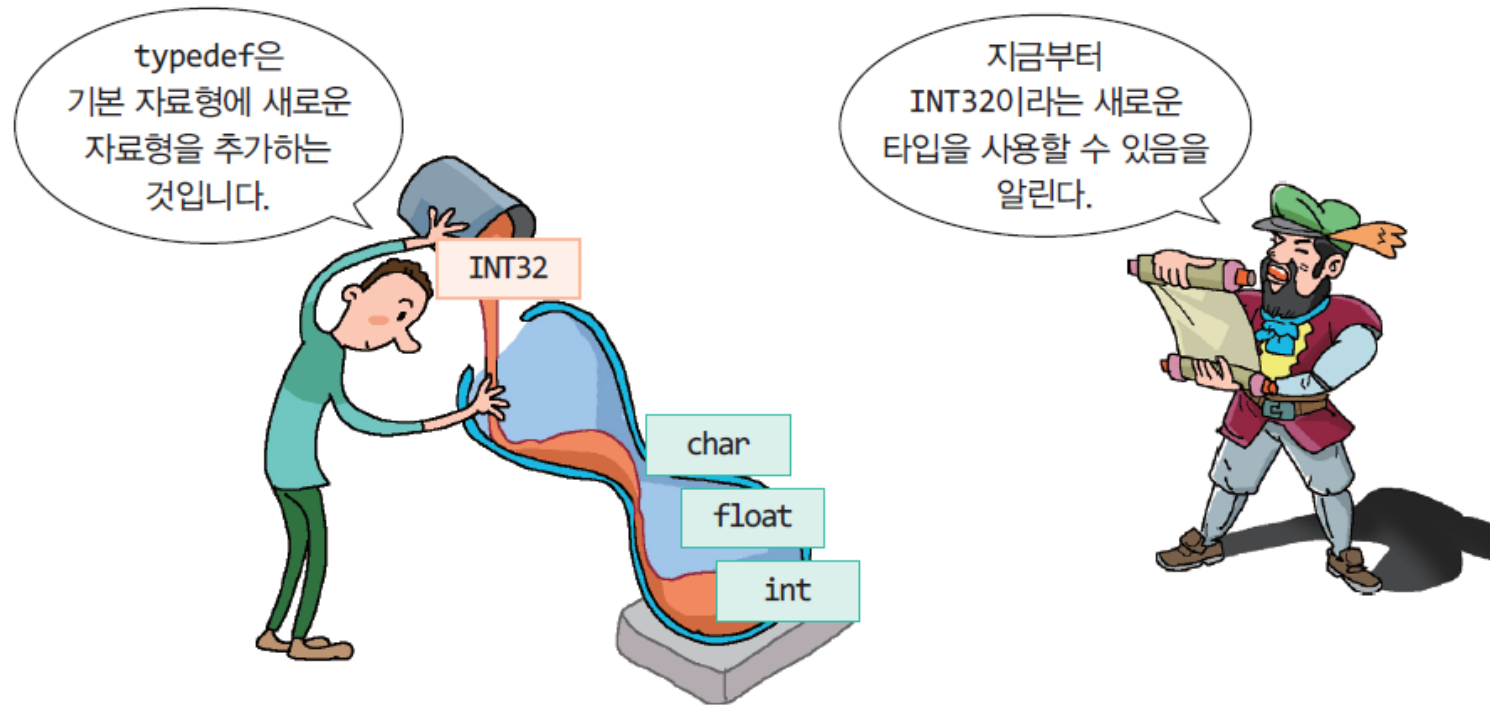
    // 구조체 할당
    for (int i = 0; i < 5; i++) {
        pJuso[i] = (struct SHuman *)malloc(sizeof(struct SHuman));
    }

    // 값 저장
    pJuso[3]->age = 45;
    printf("%d\n", pJuso[3]->age);

    // 해제
    for (int i = 0; i < 5; i++) {
        free(pJuso[i]);
    }
}
```

나이 = 45

- 새로운 자료형(type)을 정의(define)하는 것



Syntax 11.2 typedef

형식 `typedef old_type new_type;`

예 `typedef unsigned char BYTE;`

설명 `old_type`은 기존의 자료형이다. `new_type`은 새롭게 정의하려고 하는 자료형의 이름이다. 위의 문장은 새로운 자료형 `new_type`을 정의하는 것으로 그 내용은 `old_type`과 같다는 의미가 된다.

기존의 자료형

새로운 자료형


```
typedef int  INT32;  
typedef unsigned int  UINT32;  
  
INT32 i;    // int i;와 같다.  
UINT32 k;   // unsigned int k;와 같다.
```

```
typedef struct point {  
    int x;  
    int y;  
} POINT;
```

- 2차원 공간에서의 점을 구조체로 표현한 다음에 이 구조체를 typedef을 이용하여 새로운 타입인 POINT로 정의한다.



```
C:\Windows\system32\cmd.exe
(2, 3)+(10, 10)->(12, 13)
계속하려면 아무 키나 누르십시오 . . .
```

2. 구조체 – typedef 예제

59 |

```
#include <stdio.h>

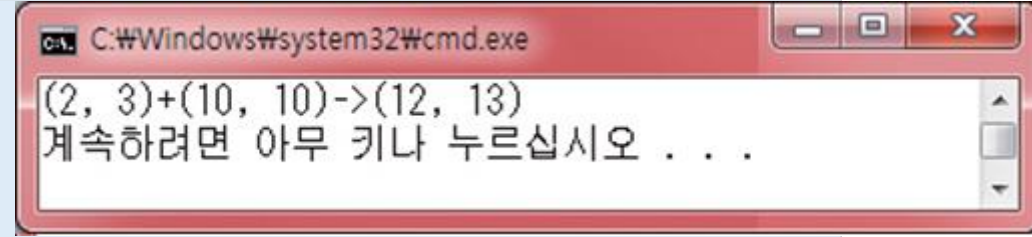
typedef struct point {
    int x;
    int y;
} POINT;

POINT translate(POINT p, POINT delta){
    POINT new_p;
    new_p.x = p.x + delta.x;
    new_p.y = p.y + delta.y;
    return new_p;
}

int main(void)
{
    POINT p = { 2, 3 };
    POINT delta = { 10, 10 };
    POINT result;

    result = translate(p, delta);
    printf("(%d, %d)+(%d, %d)->(%d, %d)\n", p.x, p.y, delta.x, delta.y, result.x, result.y);

    return 0;
}
```



■ 열거형의 이해

● 열거형의 형식

- 단순히 1, 2, 3, 4, ... 와 같은 숫자를 기호로 표현한 것

● 요일을 열거형으로 표현

- 0은 sun, 1은 mon, 2는 tue, ... 등과 같이 의미가 좀더 명확해짐
- 나열한 데이터의 값은 0에서부터 1씩 차례대로 증가함

```
enum 열거형_이름{
```

```
    기호_1;
```

```
    기호_2;
```

```
    :
```

```
};
```

```
enum 열거형_이름 열거형_변수;
```

```
enum week {
```

sun, → 0

mon, → 1

tue, → 2

wed, → 3

thu, → 4

fri, → 5

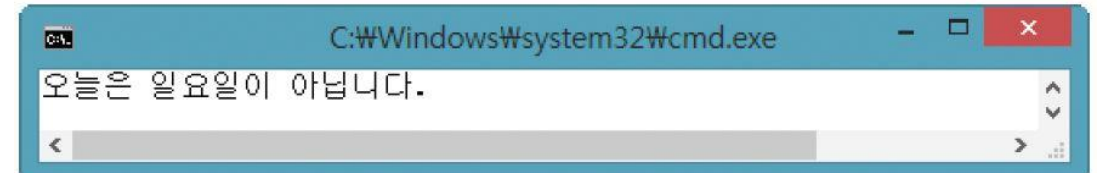
sat → 6

```
};
```

동일한 의미

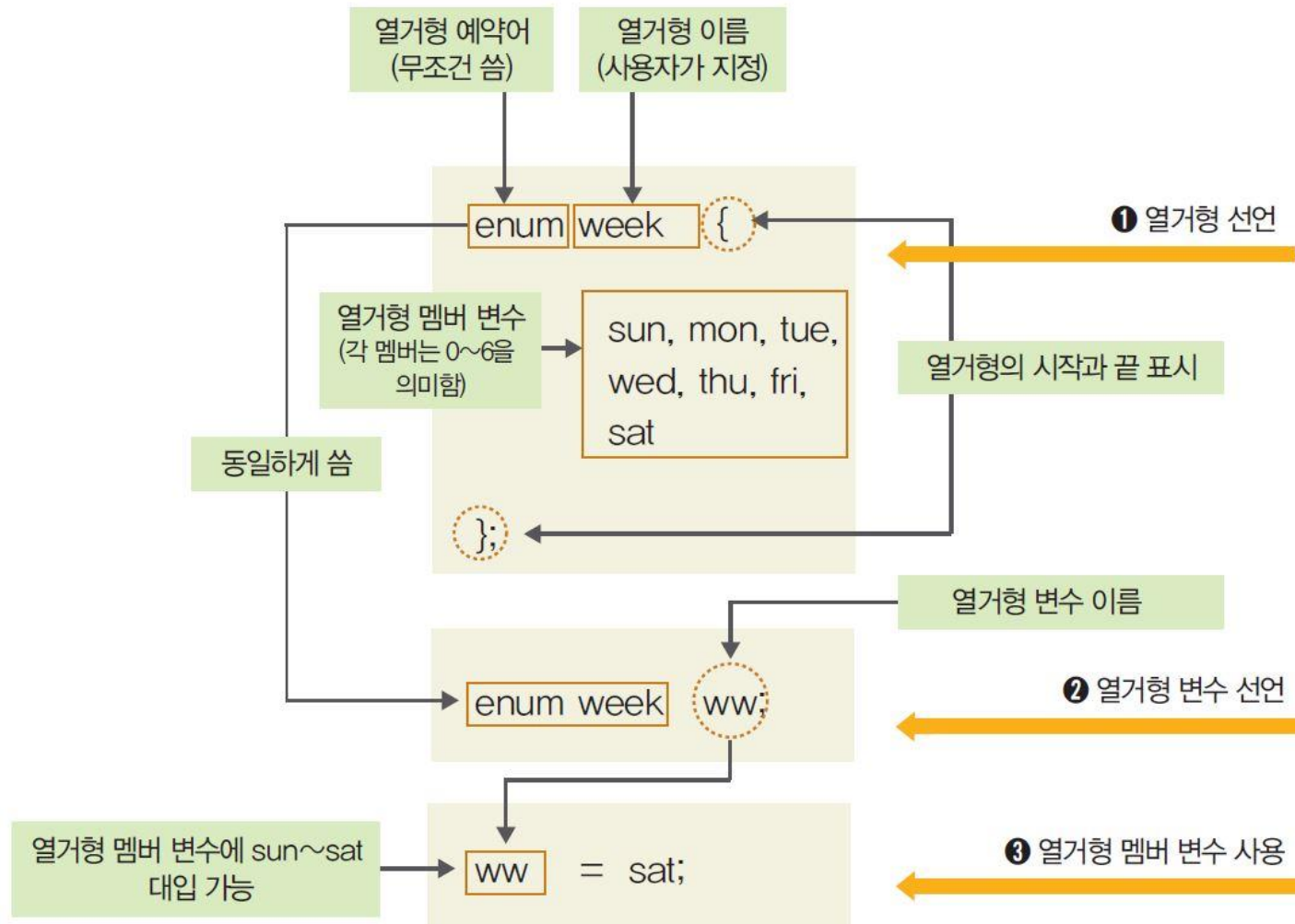
```
01 #include <stdio.h>
02
03 int main( )
04 {
05     enum week {sun, mon, tue, wed, thu, fri, sat}; ---0부터 6까지의 열거형이다.
06
07     enum week ww; ---열거형 변수 ww를 선언한다.
08
09     ww = sat; ---변수 ww에 값을 대입한다.
10
11     if (ww == sun)
12         printf("오늘은 일요일입니다.\n\n", ww); ---ww가 sun(0)인지 여부에 따라 출력한다.
13     else
14         printf("오늘은 일요일이 아닙니다.\n\n", ww);
15 }
```

실행 결과 ▼



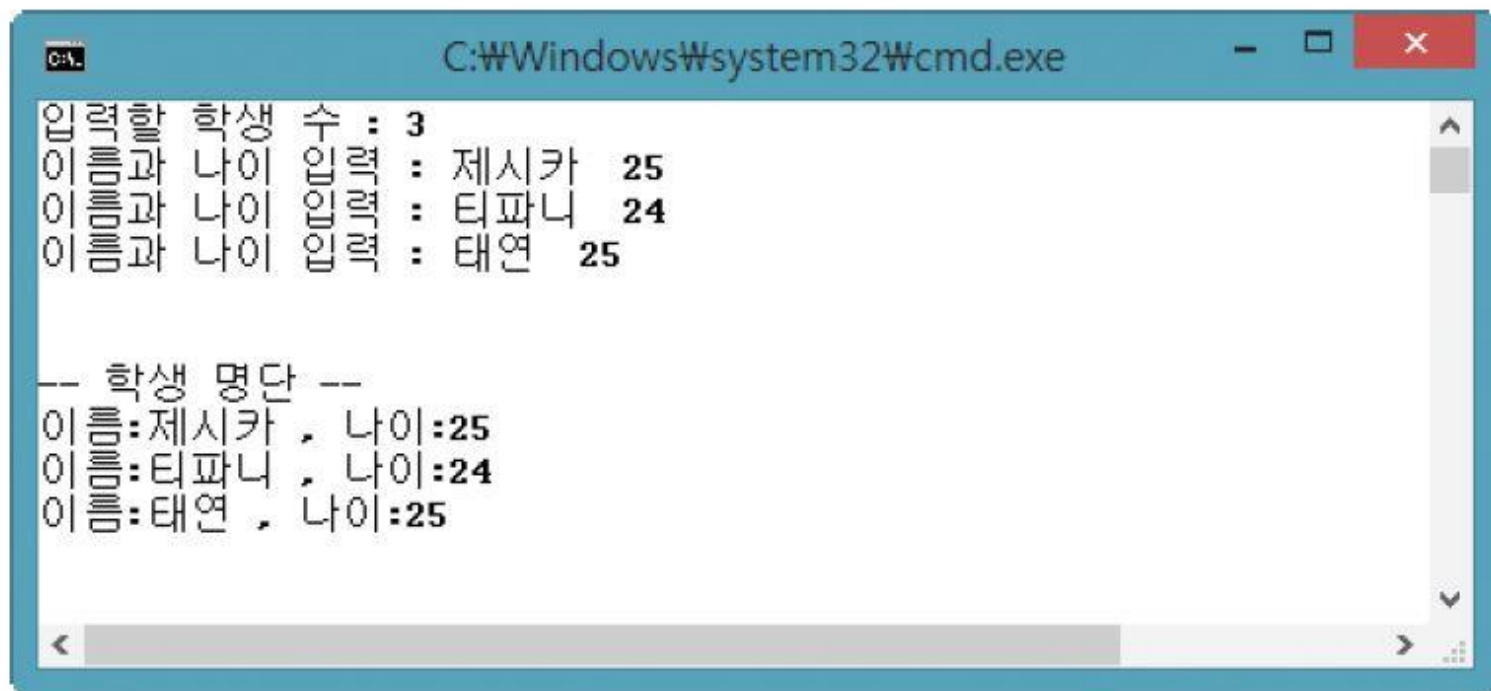
■ 주의할 점

- Sun, mon, ... 등은 변수가 아니라 정숫값 0, 1, 2,...를 의미
- 열거형 변수 ww에는 sun, mon, tue, wed, thu, fri, sat 중 하나만 대입 가능



예제 설명 구조체 포인터와 메모리 할당 함수를 이용하여 학생의 이름과 나이를 입력받아 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

입력할 학생 수 : 3
이름과 나이 입력 : 제시카 25
이름과 나이 입력 : 티파니 24
이름과 나이 입력 : 태연 25

-- 학생 명단 --
이름: 제시카 , 나이: 25
이름: 티파니 , 나이: 24
이름: 태연 , 나이: 25
```

[예제] 구조체 포인터를 활용한 학생 관리

```

01 #include <stdio.h>
02 #include <malloc.h>
03
04 int main( )
05 {
06     struct student{
07         char name[10];
08         int age;
09     };
10
11     struct student *s;
12
13     int cnt, i;
14
15     printf("입력할 학생 수 : ");
16     scanf_s("%d", &cnt);
17
18     s = (struct student *) malloc((sizeof(struct student)) * cnt);
19
20     for (i=0; i<cnt; i++)
21     {
22         printf("이름과 나이 입력 : ");
23         scanf_s("%s", s[i].name, sizeof(s[i].name));
24         scanf_s("%d", &s[i].age);
25     }
26     printf("\n\n-- 학생 명단 --\n");
27     for (i=0; i<cnt; i++)
28         printf("이름:%s , 나이:%d \n", s[i].name, s[i].age);
29
30     free(s);
31 }

```

----구조체형을 선언한다.

----구조체 포인터 변수를 선언한다.

----관리할 학생 수를 입력한다.

----학생 수만큼 메모리를 할당한다.

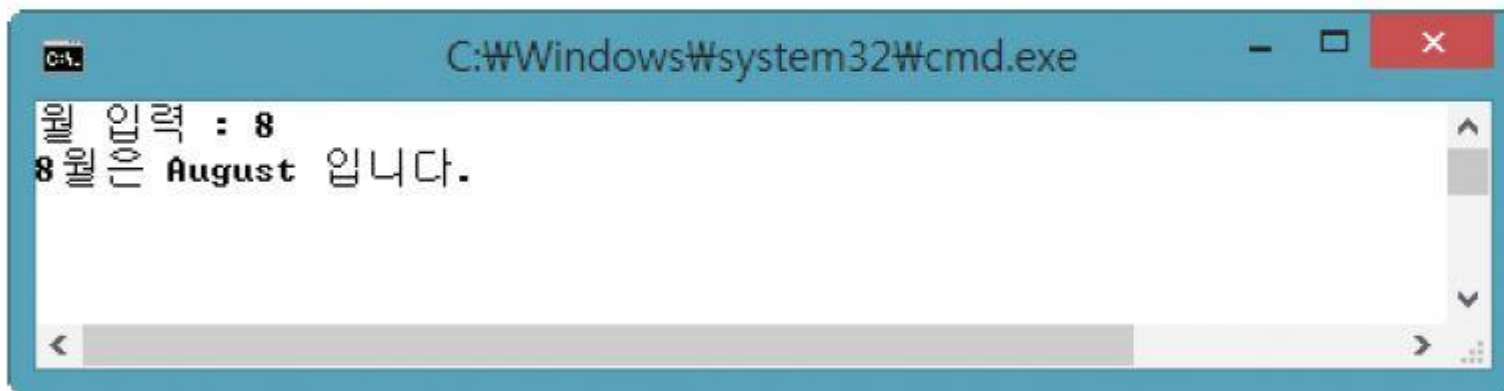
----학생 수만큼 반복하며 이름과 나이를 입력한다.

----학생 수만큼 반복하며 이름과 나이를 출력한다.

----메모리를 해제한다.

예제 설명 열거형을 활용해서 입력된 월의 이름을 출력하는 프로그램이다.

실행 결과



[예제] 열거형(enum)을 활용한 월 이름 출력

```

01 #include <stdio.h>
02 void main( )
03 {
04     enum month {
05         January=1, February, March, April, May, June,
06         July, August, September, October, November, December
07     };
08     enum month mm;
09     printf("월 입력 : ");
10     scanf_s("%d", &mm);
11     switch (mm) {
12         case January : printf("%d월은 January 입니다.", mm); break;
13         case February : printf("%d월은 February 입니다.", mm); break;
14         case March : printf("%d월은 March 입니다.", mm); break;
15         case April : printf("%d월은 April 입니다.", mm); break;
16         case May : printf("%d월은 May 입니다.", mm); break;
17         case June : printf("%d월은 June 입니다.", mm); break;
18         case July : printf("%d월은 July 입니다.", mm); break;
19         case August : printf("%d월은 August 입니다.", mm); break;
20         case September: printf("%d월은 September 입니다.", mm); break;
21         case October : printf("%d월은 October 입니다.", mm); break;
22         case November : printf("%d월은 November 입니다.", mm); break;
23         case December : printf("%d월은 December 입니다.", mm); break;
24         default :printf("잘못 입력했군요.");
25     }
26     printf("\n\n");
27 }

```

----열거형을 선언한다(1부터 시작).

----열거형 변수 mm을 선언한다.

----값을 입력한다.

----입력값에 따라 해당 월을 출력한다.