9. 구조체

♲ 구조체란?

구조체란 서로 관련이 있는 여러 종류의 data를 하나로 묶어서 관리 할 수 있도록 해주는 도구이다. 즉, 사용자가 원하는 여러 종류의 자료를 하나로 묶어 놓은 형태의 자료형(data type)을 만들 할 수 있도록 해줌으로 각 종류별로 별도의 변수를 만들어 사용하는 것보다 편리하게 프로그래밍을 할 수 있게 해준다. 따라서 사용자가 원하는 자료형을 새롭게 만들 수 있도록 해주는 것이 구조체이며 UDT(User Define Datatype) 라고도 부른다. (C언어에서 구조체는 완벽한 하나의 data type으로 인정되지 않지만 편의를 위해 data type으로 설명한다.)

♲ 구조체의 정의 및 사용방법

구조체를 만들기 위해서는 struct 이라는 키워드를 이용한다. 구조체를 만드는 방법은 아래와 같다.

```
struct 구조체명 {     // 구조체명은 사용자가 부여한다. 변수명 작성규칙과 동일하다.
    멤버1;           // 구조체를 구성하고 있는 하나 하나의 변수를 멤버변수라 한다.
    멤버2; .......
};
```

구조체는 일반변수뿐 아니라 포인터 변수, 배열, 구조체형 변수 등을 모두 자신의 멤버로 가질 수 있다.

구조체는 하나의 자료형이라 생각하면 됨으로 만든 구조제를 직접 사용하는 것이 아니라 이 구조체를 자료형으로 하는 변수를 만들어 사용한다. 구조체로 변수를 만드는 방법은 다음과 같다.

```
① struct 구조체명 변수명;                  // 구조체 변수를 만드는 방법
② struct 구조체명 변수명={초기값들..}; // 구조체 변수를 선언과 동시에 초기화하는 방법
```

만들어진 구조체형 변수는 사용자가 만든 자료형을 가지고 있음으로 대입을 제외한 모든 연산이 불가능하다. 따라서 구조체형 변수는 해당 변수를 직접 사용하는 것이 아니라 구조체형 변수 안에 있는 멤버를 호출해 사용한다. 멤버는 . (도트) 연산자를 사용해 호출한다. . (도트) 연산자는 해석 시 “안에 있는”으로 해석한다. (A.B 는 해석 시 A 안에 있는 멤버 B로 해석한다.)

```
예) struct std ob;          // std라는 구조체 형으로 변수 ob를 만든다.
    ob.nai = 10;            // ob 안에 있는 멤버 nai를 불러와 10을 넣는다.
    ob.ki = 115.3           // ob 안에 있는 멤버 ki를 불러와 115.3을 넣는다.
```

구조체형 변수도 변수임으로 지역변수, 전역변수, 정적변수, 외부변수로 모두 선언될 수 있으며 주소값을 가지고 있다면 포인터 변수와 배열로도 생성 할 수 있다.

예제 9-1 : 구조체를 만들고 사용하는 예제

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};

void main ( ) {
    struct std ob;          // 구조체 std형 변수 ob의 선언
    ob.a = 10;                      // ob안에 있는 멤버 a에 10을 대입
    ob.b = 20;                      // ob안에 있는 멤버 b에 20을 대입
    printf("ob.a = %d₩n", ob.a);
    printf("ob.b = %d₩n", ob.b);
}
```

예제 9-2 : 구조체형 변수의 크기를 알아보는 예제

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};

void main ( ) {
    struct std ob;          // 구조체 std형 변수 ob의 선언
    printf("sizeof(ob) = %d₩n", sizeof(ob) );
}
```

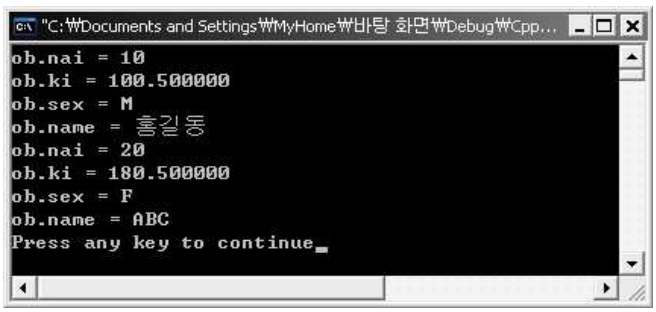
설명 : 구조체형 변수의 크기는 멤버변수들 크기의 총합과 같다. 단, 컴파일러의 종류에 따라 변수와 변수 사이에 여휴공간이 발생하여 멤버변수들 크기의 총 합보다 구조체형 변수의 크기가 더 크게 나오는 경우가 있다. (컴파일러의 종류 및 설정에 따라 변수의 엑세스 속도 향상을 위해 일정 간격으로 변수를 배열하려는 특성이 있다.)

예제 9-3 : 구조체형 변수를 만들면서 초기값을 부여 하는 예제

```
#include <stdio.h>
struct std {                  // std라는 이름으로 구조체 정의
    int a;                   // 멤버변수 a의 선언
    int b;                   // 멤버변수 b의 선언
};
void main ( ) {
    struct std ob = { 10, 20 };     // 구조체형 변수를 만들면서 넣은 초기값은
                                    // 순서대로 멤버변수에 들어간다.
    printf("ob.a = %d₩n", ob.a);
    printf("ob.b = %d₩n", ob.b);
}
```

예제 9-4 : 여러가지 종류의 멤버 변수를 가지는 구조체에 대한 예제

```
#include <stdio.h>
struct std {                  // std라는 이름으로 구조체 정의
    int nai;
    double ki;
    char sex;
    char name[11];
};
void main ( ) {
    struct std ob = { 10, 100.5, 'M', "홍길동" }; // 여러종류의 멤버를 초기화 하는 방법
    printf("ob.nai = %d₩n", ob.nai);
    printf("ob.ki = %lf₩n", ob.ki);
    printf("ob.sex = %c₩n", ob.sex);
    printf("ob.name = %s₩n", ob.name);
    ob.nai = 20;
    ob.ki = 180.5;
    ob.sex = 'F';
    ob.name[0] = 'A';
    ob.name[1] = 'B';
    ob.name[2] = 'C';
    ob.name[3] = NULL;
    printf("ob.nai = %d₩n", ob.nai);
    printf("ob.ki = %lf₩n", ob.ki);
    printf("ob.sex = %c₩n", ob.sex);
    printf("ob.name = %s₩n", ob.name);
}
```

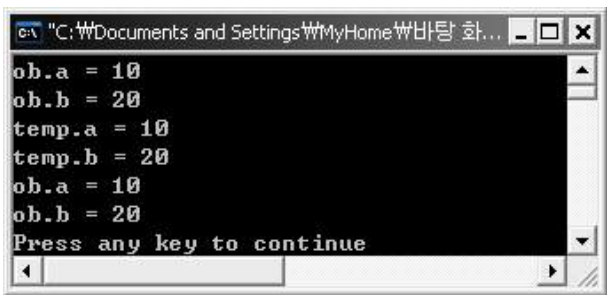


예제 9-5 : 구조체형 변수의 직접 연산은 대입만 가능하다는 것을 확인하는 예제

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};
void main ( ) {
    struct std ob = { 10, 20 }, pb;
    pb = ob;            // 같은 자료형을 가지는 구조체형 변수끼리는 대입을 할 수 있다.
    printf("pb.a = %d₩n", pb.a);
    printf("pb.b = %d₩n", pb.b);
    // ob = ob + 3;    // 에러 : struct std형 변수와 int는 더할 수 없다.
    // ob++;           // 에러 : struct std형 변수는 ++연산을 할 수 없다.
}
```

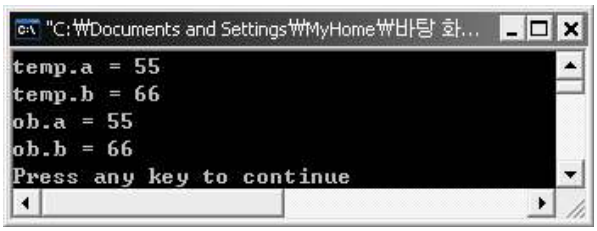
예제 9-6 : 구조체형 변수를 함수의 매개변수로 넘겨주는 방법에 대한 예제 (Call By Value)

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};
void func(struct std temp) {          // 구조체형 변수를 매개변수로 받아오는 방법
    printf("temp.a = %d₩n", temp.a);
    printf("temp.b = %d₩n", temp.b);
    temp.a = 55;                      // temp안에 있는 멤버변수의 값을 변경
    temp.b = 66;
}
void main ( ) {
    struct std ob = { 10, 20 };
    printf("ob.a = %d₩n", ob.a);      // 10
    printf("ob.b = %d₩n", ob.b);      // 20
    func( ob );
    printf("ob.a = %d₩n", ob.a);      // 10
    printf("ob.b = %d₩n", ob.b);      // 20
}
```



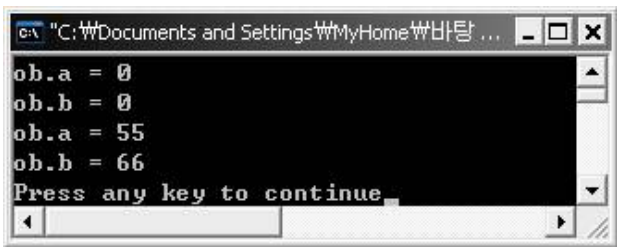
예제 9-7 : 구조체형 변수를 리턴하는 방법

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};
struct std func( ) {            // 구조체형 변수를 리턴하는 방법
    struct std temp;
    temp.a = 55;                        // temp안에 있는 멤버변수의 값을 변경
    temp.b = 66;
    printf("temp.a = %d₩n", temp.a);
    printf("temp.b = %d₩n", temp.b);
    return temp;
}
void main ( ) {
    struct std ob;
    ob = func( );
    printf("ob.a = %d₩n", ob.a);        // 55
    printf("ob.b = %d₩n", ob.b);        // 66
}
```



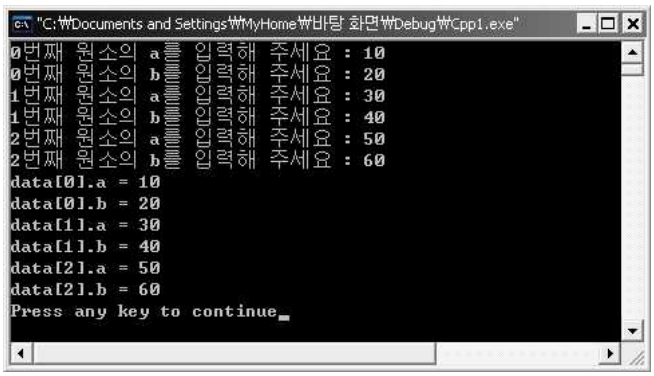
예제 9-8 : 구조체 선언과 동시에 전역변수로 구조체형 변수를 만드는 방법

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
} ob;

void main ( ) {
    printf("ob.a = %d₩n", ob.a);        // 0
    printf("ob.b = %d₩n", ob.b);        // 0
    ob.a = 55;
    ob.b = 66;
    printf("ob.a = %d₩n", ob.a);        // 55
    printf("ob.b = %d₩n", ob.b);        // 66
}
```



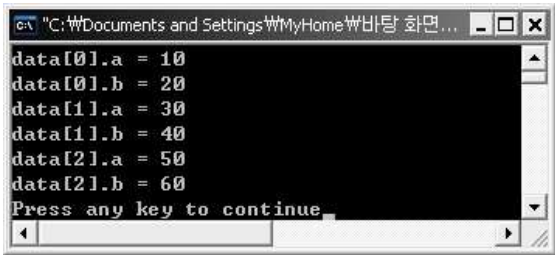
예제 9-9 : 구조체 변수를 배열로 만드는 방법

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};

void main ( ) {
    struct std data[3];      // 구조체 std형으로 크기가 [3]인 배열 data를 만든다.
    int i;
    for(i=0; i<=2; i++ ) {   // 반복문을 이용해 구조체 배열에 자료를 입력 받는다.
        printf("%d번째 원소의 a를 입력해 주세요 : ", i);
        scanf("%d", &data[i].a);
        printf("%d번째 원소의 b를 입력해 주세요 : ", i);
        scanf("%d", &data[i].b);
    }
    for(i=0; i<=2; i++ ) {   // 반복문을 이용해 구조체 배열의 자료를 출력한다.
        printf("data[%d].a = %d₩n", i, data[i].a);
        printf("data[%d].b = %d₩n", i, data[i].b);
    }
}
```



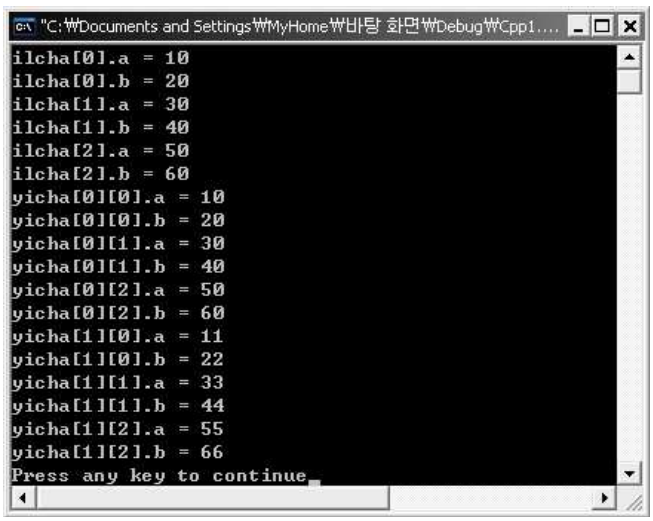
예제 9-10 : 구조체 변수를 배열로 만들면서 초기값을 넣어주는 방법

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};

void main ( ) {
    struct std data[3] = { {10,20},{30,40},{50,60} };      // 구조체 배열의 초기화 방법
    int i;
    for(i=0; i<=2; i++ ) {   // 반복문을 이용해 구조체 배열의 자료를 출력한다.
        printf("data[%d].a = %d₩n", i, data[i].a);
        printf("data[%d].b = %d₩n", i, data[i].b);
    }
}
```



예제 9-11 : 구조체 변수를 이차원 배열로 만들면서 초기값을 넣어주는 방법

```
#include <stdio.h>
struct std {                // std라는 이름으로 구조체 정의
    int a;                  // 멤버변수 a의 선언
    int b;                  // 멤버변수 b의 선언
};

void main ( ) {
    struct std ilcha[3] = { 10,20,30,40,50,60 };      // 구조체 배열의 초기화 방법
    struct std yicha[2][3] = { {{10,20},{30,40},{50,60}},{{11,22},{33,44},{55,66}} };
    int i, j;
    for(i=0; i<=2; i++ ) {   // 반복문을 이용해 구조체 배열의 자료를 출력한다.
        printf("ilcha[%d].a = %d₩n", i, ilcha[i].a);
        printf("ilcha[%d].b = %d₩n", i, ilcha[i].b);
    }
    for(i=0; i<=1; i++) {   // 반복문을 이용해 구조체 배열의 자료를 출력한다.
        for(j=0; j<=2; j++) {
            printf("yicha[%d][%d].a = %d₩n", i, j, yicha[i][j].a);
            printf("yicha[%d][%d].b = %d₩n", i, j, yicha[i][j].b);
        }
    }
}
```



설명 : 구조체형 이차원 배열을 초기화 시키는 방법

```
struct std data[2][3]={ {{10,20},{30,40},{50,60}},{{11,22},{33,44},{55,66}} };
struct std data[2][3]={ {10,20,30,40,50,60},{11,22,33,44,55,66} };
struct std data[2][3]={ 10,20,30,40,50,60,11,22,33,44,55,66 };
```

♲ 구조체형 포인터를 만들고 사용하는 방법

구조체로 만든 변수도 변수임으로 메모리에 생성되었다면 주소값을 가지게 된다. 따라서 이러한 구조체형 변수의 주소값을 저장 할 수 있는 포인터 변수도 생성 할 수 있다.

구조체형 포인터 변수를 만드는 방법

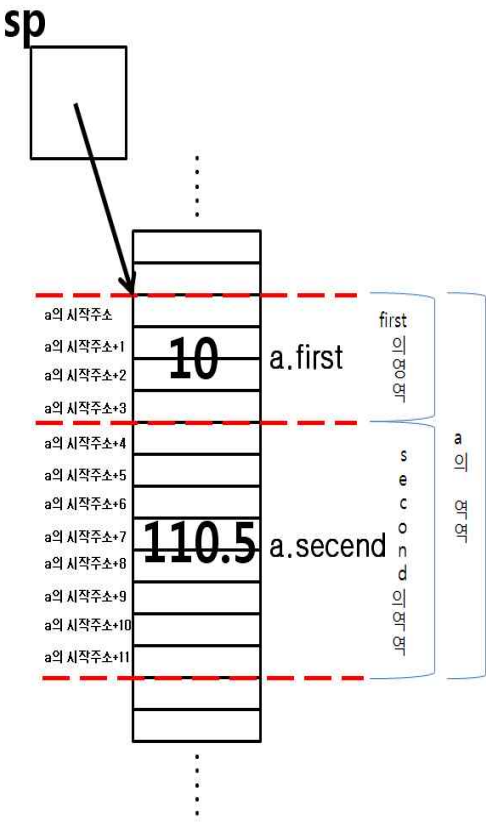
```
struct 구조체명 *포인터변수명;     // 구조체형 변수의 주소값을 보관 할 수 있는 포인터 변수
struct 구조체명 *배열명[크기];     // 구조체형 포인터 변수를 여러개(배열) 만드는 방법
```

구조체형 변수는 자신과 같은 자료형을 가지는 구조체형 변수의 주소값 만을 보관 할 수 있다.

```
struct std ob;
struct std *sp = &ob; // ob의 자료형이 struct std형 임으로 sp가 ob의 주소를 보관할 수 있다.
```

구조체형 포인터는 구조체형 변수의 시작 주소값을 보관한다. 이때 구조체형 변수의 시작 주소값은 첫 번째 멤버변수의 시작주소와 동일 하지만 자료형이 다름으로 오해하지 말아야 한다. 구조체형 포인터를 역참조하면 구조체형 변수가 나온다.

```
struct myst {
    int first;
    double second;
};
void main ( ) {
    struct myst a = { 10, 110.5};
    struct myst *sp = &a;
    printf("&a = %p₩n", &a);
    printf("&a.first = %p₩n", &a.first);
    printf("&a.second = %p₩n", &a.second);
    printf("sp = %p₩n", sp);
}
```



위의 예제에서 a의 주소값과 a.first의 주소값은 동일하다. 하지만 a의 주소값인 &a는 struct myst형 포인터 이고 &a.first는 int형 포인터이다.

포인터 변수를 역참조시 포인터와 같은 자료형의 data가 나오므로 struct myst형 포인터인 sp를 역참조하면 struct myst형인 a가 나온다. (100이 나오는 것이 아니다! 100이 나오려면 포인터가 int형이어야 한다. 즉, &a.first의 값을 가지는 int형 포인터 변수를 역참조 해야 100이 나오게 된다.)

역참조를 통해 포인터 변수가 가리키는 구조체 변수 안의 값을 꺼내오는 방법

```
(*구조체형포인터).호출할멤버변수명;
구조체형포인터->호출할멤버변수명;
```

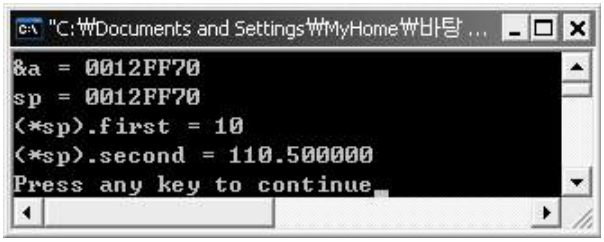
-> 연산자는 구조체형 포인터를 통해 구조체 안에 있는 변수를 호출하기 위해 사용하는 연산자이며 A->B로 사용시 "A가 가르키는 구조체 변수 안에있는 B를 호출해라" 라는 뜻이된다.
(->는 “가리키는곳 안에 있는” 이라고 읽으면 된다.)

예제 9-12 : 구조체 변수의 주소값과 멤버의 주소값을 알아보는 예제

```
#include <stdio.h>
struct myst {
    int first;
    double second;
};
void main ( ) {
    struct myst a = { 10, 110.5};
    printf("&a = %p₩n", &a);
    printf("&a.first = %p₩n", &a.first);
    printf("&a.second = %p₩n", &a.second);
}
```

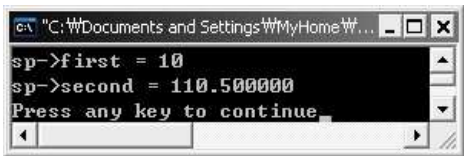
예제 9-13 : 구조체 포인터 변수를 만들고 사용하는 방법

```
#include <stdio.h>
struct myst {
    int first;
    double second;
};
void main ( ) {
    struct myst a = { 10, 110.5};
    struct myst *sp;
    sp = &a;  // struct myst형 포인터 sp에는 struct myst형 변수의 주소만 보관할 수 있다.
    printf("&a = %p₩n", &a);
    printf("sp = %p₩n", sp);
    // printf("*sp = %d₩n", *sp);  // sp를 역참조하면 a가 나옴으로 %d로 출력 할 수 없다.
    printf("(*sp).first = %d₩n", (*sp).first);
    printf("(*sp).second = %lf₩n", (*sp).second);
}
```



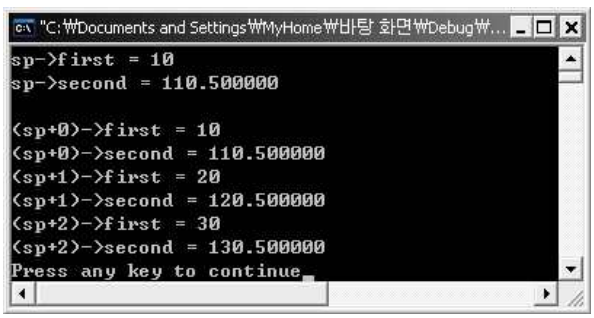
예제 9-14 : ->연산자를 이용해 구조체형 포인터를 역참조 하는 방법

```
#include <stdio.h>
struct myst {
    int first;
    double second;
};
void main ( ) {
    struct myst a = { 10, 110.5};
    struct myst *sp;
    sp = &a;  // struct myst형 포인터 sp에는 struct myst형 변수의 주소만 보관할 수 있다.
    printf("sp->first = %d₩n", sp->first);
    printf("sp->second = %lf₩n", sp->second);
}
```



예제 9-15 : 구조체형 포인터를 통한 구조체형 배열의 접근

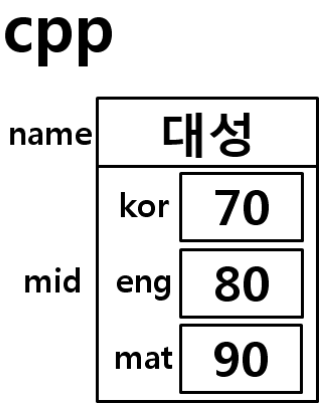
```
#include <stdio.h>
struct myst {
    int first;
    double second;
};
void main ( ) {
    struct myst k[3] = { {10,110.5}, {20,120.5}, {30,130.5} };
    struct myst *sp;
    sp = k;  // k는 struct myst형 배열의 이름임으로 struct myst형 포인터 이다.
    printf("sp->first = %d₩n", sp->first);
    printf("sp->second = %lf₩n", sp->second);
    printf("₩n");
    printf("(sp+0)->first = %d₩n", (sp+0)->first);
    printf("(sp+0)->second = %lf₩n", (sp+0)->second);
    printf("(sp+1)->first = %d₩n", (sp+1)->first);
    printf("(sp+1)->second = %lf₩n", (sp+1)->second);
    printf("(sp+2)->first = %d₩n", (sp+2)->first);
    printf("(sp+2)->second = %lf₩n", (sp+2)->second);
}
```



♲ 포함구조체

구조체의 멤버로 또 다른 구조체가 들어 있는 구조체를 포함 구조체라 한다. 따라서 포함구조체는 멤버 중 구조체형 변수가 멤버로 있는 구조체이며 이렇게 멤버로된 구조체도 그 안의 멤버를 호출해 사용한다.

```
struct score {      // 일반 구조체
    int kor;
    int eng;
    int mat;
};
struct sung {       // 멤버로 struct score형 변수가
                    // 들어 있는 포함구조체
    char name[11];
    struct score mid;
};
void main ( ) {
    struct sung cpp = { "대성",70,80,90} ;
}
```



오른쪽 그림과 같이 struct sung으로 만든 구조체형 변수 cpp에는 두 개의 멤버 name과 mid가 있고 mid의 자료형이 struct score형임으로 mid안에 kor, eng, mat의 세 멤버가 들어 있다.

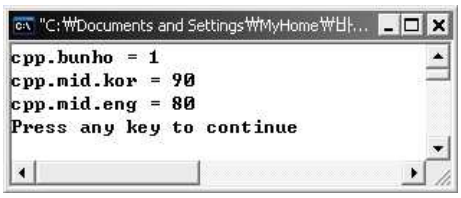
포함구조체에 있는 멤버 중 구조체형 멤버 안의 멤버는 . 연산자를 두 번 사용해 호출한다.

위의 예에서

```
cpp.mid.kor = 10;   // cpp 안에 있는 mid 안에있는 kor에 10을 호출해 10을 넣는다.
```

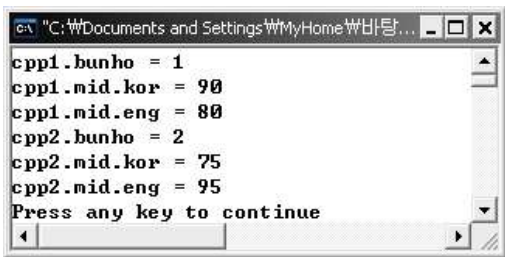
예제 9-16 : 포함 구조체의 선언 및 사용 방법

```
#include <stdio.h>
struct score {       // 일반 구조체
    int kor;
    int eng;
};
struct sung { // 멤버로 struct score형 변수가 들어 있는 포함구조체
    int bunho;
    struct score mid;
};
void main ( ) {
    struct sung cpp;
    cpp.bunho = 1;
    cpp.mid.kor = 90;
    cpp.mid.eng = 80;
    printf("cpp.bunho = %d₩n", cpp.bunho);
    printf("cpp.mid.kor = %d₩n", cpp.mid.kor);
    printf("cpp.mid.eng = %d₩n", cpp.mid.eng);
}
```



예제 9-17 : 포함 구조체형 배열의 초기화방법

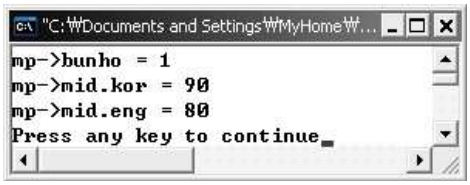
```
#include <stdio.h>
struct score {       // 일반 구조체
    int kor;
    int eng;
};
struct sung { // 멤버로 struct score형 변수가 들어 있는 포함구조체
    int bunho;
    struct score mid;
};
void main ( ) {
    struct sung cpp1 = { 1,{90, 80}};
    struct sung cpp2 = { 2, 75, 95};
    printf("cpp1.bunho = %d₩n", cpp1.bunho);
    printf("cpp1.mid.kor = %d₩n", cpp1.mid.kor);
    printf("cpp1.mid.eng = %d₩n", cpp1.mid.eng);
    printf("cpp2.bunho = %d₩n", cpp2.bunho);
    printf("cpp2.mid.kor = %d₩n", cpp2.mid.kor);
    printf("cpp2.mid.eng = %d₩n", cpp2.mid.eng);
}
```



예제 9-18 : 포인터를 통해 포함 구조체 안에 있는 구조체 멤버를 호출하는 예제

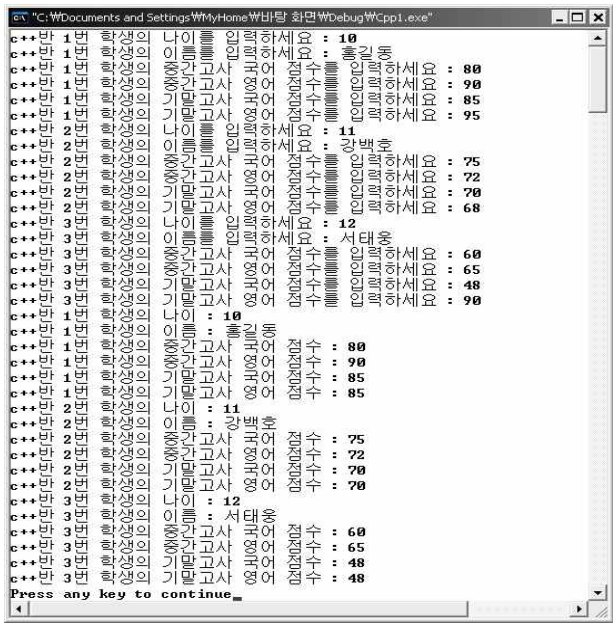
```
#include <stdio.h>
struct score {       // 일반 구조체
    int kor;
    int eng;
};
struct sung { // 멤버로 struct score형 변수가 들어 있는 포함구조체
    int bunho;
    struct score mid;
};
```

```
void main ( ) {
    struct sung cpp1 = { 1,{90, 80}};
    struct sung *mp;
    mp = &cpp1;
    printf("mp->bunho = %d₩n", mp->bunho);
    printf("mp->mid.kor = %d₩n", mp->mid.kor);
    printf("mp->mid.eng = %d₩n", mp->mid.eng);
}
```



예제 9-19 : 구조체형 멤버가 두 개 들어 있는 포함 구조체

```
#include <stdio.h>
struct score {       // 일반 구조체
    int kor;
    int eng;
};
struct sung { // 멤버로 struct score형 변수가 들어 있는 포함구조체
    int nai;
    char name[11];
    struct score mid;
    struct score final;
};
void main ( ) {
    struct sung cpp[3];
    int i;
    for(i=0; i<=2; i++) {
        printf("c++반 %d번 학생의 나이를 입력하세요 : ", i+1);
        scanf("%d", &cpp[i].nai);
        printf("c++반 %d번 학생의 이름를 입력하세요 : ", i+1);
        scanf("%s", cpp[i].name);
        printf("c++반 %d번 학생의 중간고사 국어 점수를 입력하세요 : ", i+1);
        scanf("%d", &cpp[i].mid.kor);
        printf("c++반 %d번 학생의 중간고사 영어 점수를 입력하세요 : ", i+1);
        scanf("%d", &cpp[i].mid.eng);
        printf("c++반 %d번 학생의 기말고사 국어 점수를 입력하세요 : ", i+1);
        scanf("%d", &cpp[i].final.kor);
        printf("c++반 %d번 학생의 기말고사 영어 점수를 입력하세요 : ", i+1);
        scanf("%d", &cpp[i].final.eng);
    }
    for(i=0; i<=2; i++) {
        printf("c++반 %d번 학생의 나이 : %d₩n", i+1, cpp[i].nai);
        printf("c++반 %d번 학생의 이름 : %s₩n", i+1, cpp[i].name);
        printf("c++반 %d번 학생의 중간고사 국어 점수 : %d₩n", i+1, cpp[i].mid.kor);
        printf("c++반 %d번 학생의 중간고사 영어 점수 : %d₩n", i+1, cpp[i].mid.eng);
        printf("c++반 %d번 학생의 기말고사 국어 점수 : %d₩n", i+1, cpp[i].final.kor);
        printf("c++반 %d번 학생의 기말고사 영어 점수 : %d₩n", i+1, cpp[i].final.kor);
    }
}
```



♲ 공용체

공용체란 구조체와 유사 하지만 구조체 처럼 각각의 멤버가 모두 메모리에 공간을 차지하고 만들어지는 것이 아니라 모든 멤버중 가장 크기가 큰 멤버의 크기만큼만 메모리를 할당받아 모든 멤버변수가 하나의 공간을 같이 사용하는 자료형이다. 메모리의 공간을 효율적으로 활용하기 위해 사용한다.

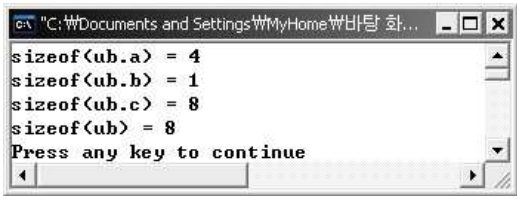
공용체를 만들고 사용하는 방법

구조체와 동일하지만 struct 대신에 union이라는 키워드를 사용한다.

```
union 공용체명 {                    // 공용체를 만드는 방법
    멤버들...
    ......
};
void main ( ) {
    union 공용체명 변수명;          // 공용체형 변수를 선언하는 방법
}
```

예제 9-20 : 공용체를 만드는 방법과 공용체형 변수의 크기

```
#include <stdio.h>
union ut {              // 공용체
    int a;
    char b;
    double c;
};
void main ( ) {
    union ut ub;
    printf("sizeof(ub.a) = %d₩n", sizeof(ub.a) );
    printf("sizeof(ub.b) = %d₩n", sizeof(ub.b) );
    printf("sizeof(ub.c) = %d₩n", sizeof(ub.c) );
    printf("sizeof(ub) = %d₩n", sizeof(ub) );
}
```



예제 9-21 : 공용체형 변수의 사용

```
#include <stdio.h>
union ut {              // 공용체
    int a;
    char b;
    double c;
};
void main ( ) {
    union ut ub;
    ub.a = 10;
    printf("ub.a = %d₩n", ub.a);     // 10이 출력된다.
    ub.b = 'R';
    printf("ub.b = %c₩n", ub.b);     // R이 출력된다.
    ub.c = 12.34;
    printf("ub.c = %lf₩n", ub.c);    // 12.34가 출력된다.

    printf("₩n");
    printf("ub.a = %d₩n", ub.a);     // a는 c 때문에 지워졌음으로 쓰레기 값이 나온다.
    printf("ub.b = %c₩n", ub.b);     // b는 c 때문에 지워졌음으로 쓰레기 값이 나온다.
    printf("ub.c = %lf₩n", ub.c);    // c만 남아있음으로 12.34가 그대로 나온다.

    ub.a = 123456;
    printf("₩n");
    printf("ub.a = %d₩n", ub.a);     // a는 그대로 123456이 나온다.
    printf("ub.b = %c₩n", ub.b);     // b는 a 때문에 지워졌음으로 쓰레기 값이 나온다.
    printf("ub.c = %lf₩n", ub.c);    // c는 a 때문에 지워졌음으로 쓰레기 값이 나온다.
}
```