원리를 알면 IT가 맛있다
IT COOKBOOK

IT CookBook, C++ 하이킹 객체지향과 만나는 여행

[강의교안 이용 안내]

C++ 하이킹

객체지향과 만나는 여행

Chapter 08. 구조체

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- struct를 사용해서 새로운 자료형을 정의한다.
- 구조체를 다양하게 사용한다.
- 매개변수가 구조체인 함수를 정의한다.
- 구조체 포인터를 이용해서 구조체를 간접 참조한다.
- 구조체로 배열을 선언하고 사용한다.
- 공용체를 정의하고 사용하는 방법을 학습한다.
- 열거체를 정의하고 사용하는 방법을 학습한다.
- typedef를 정의하는 방법을 학습한다.

01 구조체의 이해

구조체 선언

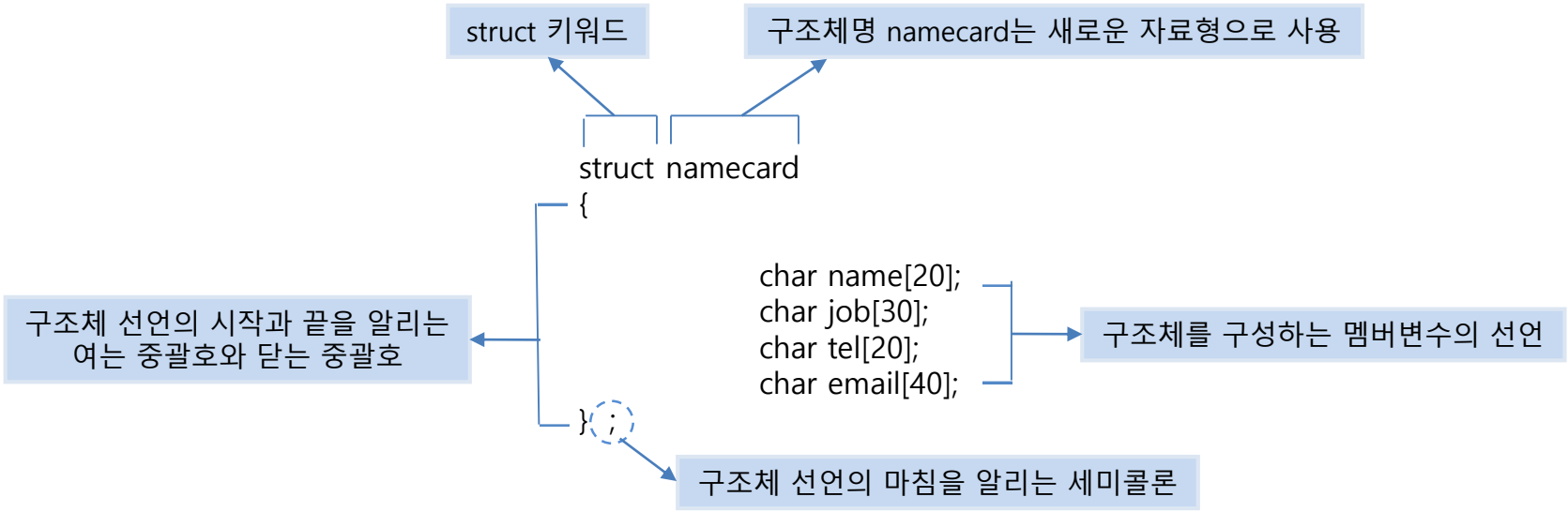
구조체 선언 기본 형식

```
struct 구조체명{
    자료형 멤버변수;
    자료형 멤버변수;
    ...
    ...
};
```

❶ 구조체명 : struct라는 예약어로 서로 다른 구조체 여러 개를 선언할 경우, 이들을 구분하려고 지정한다. 구조체 선언 후에 구조체 변수를 선언할 때 이 구조체명을 사용한다.

❷ 구조체 멤버변수 선언 : 구조체 선언문은 멤버 여러 개로 구성되므로 중괄호로 묶어서 그 내부에 멤버변수를 선언한다. 구조체 멤버변수는 일반 변수나 배열을 선언하는 방법과 동일하다.

01 구조체의 이해



[그림 8-1] **명함 관리 프로그램의 구조체 예**

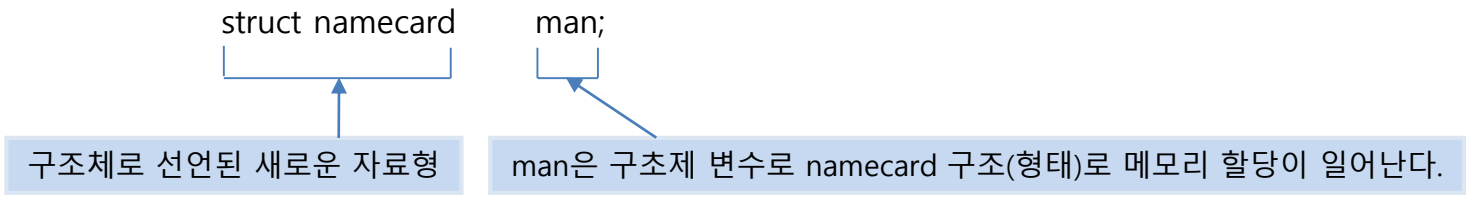
- 구조체 선언은 새로운 자료형을 만들어 내는 것을 의미하는데, 자료형은 변수를 선언하는 용도로 사용될 때 진정한 의미가 있다.

구조체 변수 선언 기본 형식

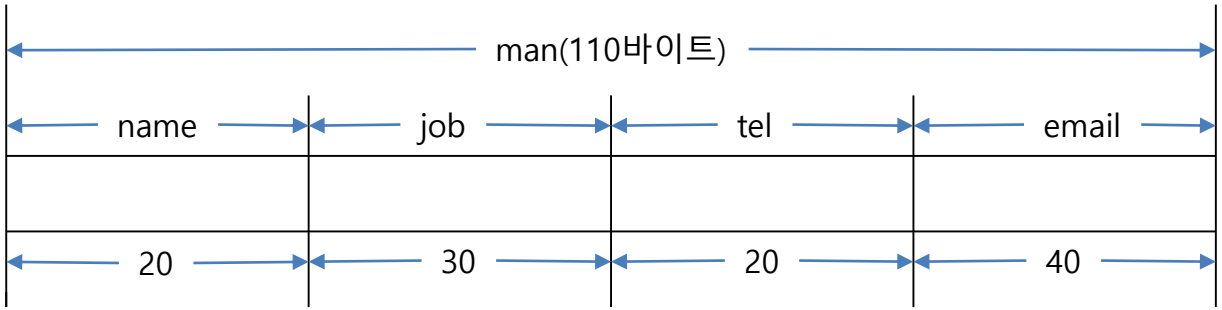
```
struct 구조체명 구조체변수명1, 구조체변수명2, ..., 구조체변수명n;
```

01 구조체의 이해

- 구조체 선언문은 일반적으로 파일의 머리 부분에 기술한다. 구조체 선언이 되어 있지 않은 상태에서 구조체 변수를 선언하면 "구조체 namecard가 선언되어 있지 않다."는 컴파일 에러가 발생한다.



- 구조체 변수 man이 메모리에 어떻게 할당되는지 그림으로 표현해 보자.



01 구조체의 이해


```
struct namecard man; // ------ ❶
namecard man;        // ------ ❷
```

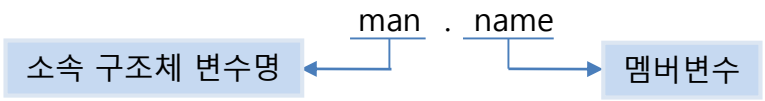
- 구조체 변수를 선언하는 방법은 ❶과 같이 예약어 struct를 붙이기도 하지만 ❷와 같이 예약어 struct를 생략하고 구조체명만으로 구조체 변수를 선언하는 것을 허용한다.

- 선언된 구조체 변수를 사용해 보자.

구조체 변수 기본 형식

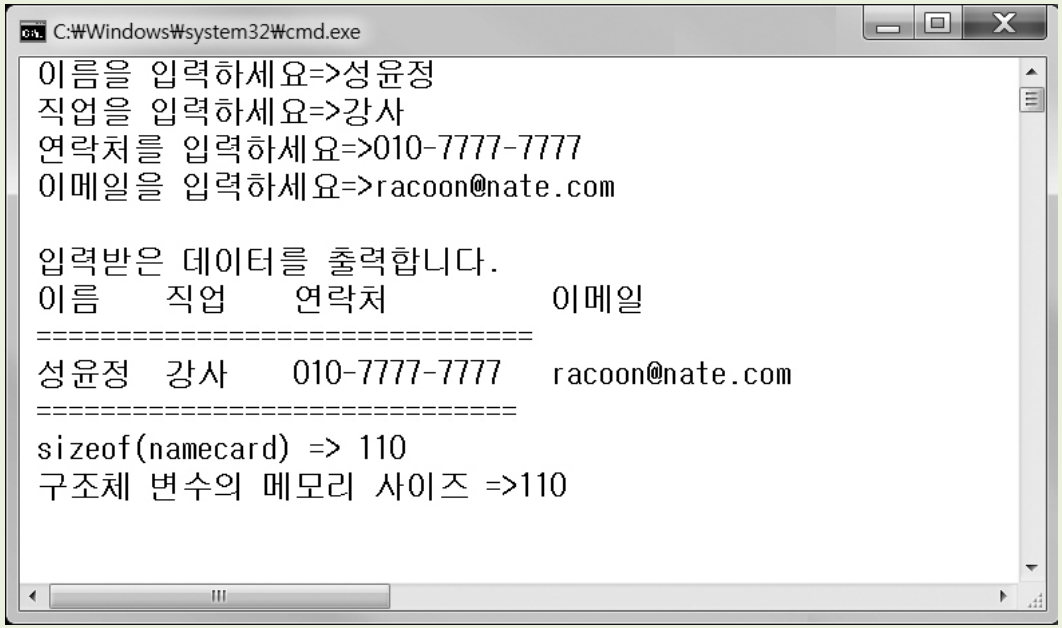
```
구조체 변수명.멤버변수;
```

- 멤버들의 집합체인 구조체는 멤버변수 단위로 값을 저장하는데, 멤버변수는 특정 구조체에 소속된 구성원이어서 단독적으로 사용하지 못하므로 다음 예처럼 . 연산자 앞에 반드시 구조체 변수명을 지정해야 한다.



예제 8-1. 구조체를 정의하고 사용하기(08_01.cpp)

교재의 소스코드 참고



01 구조체의 이해

구조체 변수의 초기화

- 배열처럼 값 4개를 콤마(,)로 구분해서 나열하고 전체를 중괄호로 묶는다.

```
namecard x={ "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"};
```

- 일반 변수에서 한꺼번에 변수 2개 이상을 선언하면서 초깃값을 줄 수 있듯이 구조체 변수도 다음과 같이 콤마(,)를 사용해서 변수에 여러 초깃값을 구분해서 나열하고 중괄호로 묶을 수 있다.

  - **일반 변수 초깃값 설정 예**

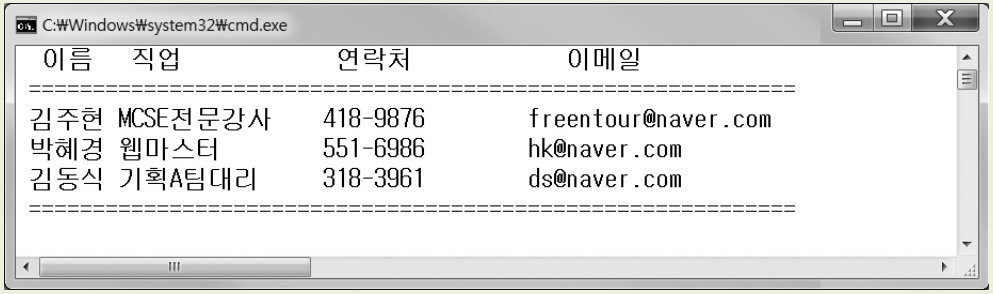
```
int   a=10, b=20, c=30;
```

  - **구조체 변수 초깃값 설정 예**

```
namecard x={ "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"},
y={ "박혜경", "웹마스터", "551-6986", "hk@naver.com"},
z={ "김동식", "기획A팀대리", "318-3961", "ds@naver.com"};
```

예제 8-2. 구조체 초기화 하기(08_02.cpp)


```
01 #include <iostream>
02 using namespace std;
03 struct namecard{
04 char name[20];
05 char job[30];
06 char tel[20];
07 char email[40];
08 };
09 void main()
10 {
11 namecard x={ "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"};
12 namecard y={ "박혜경", "웹마스터", "551-6986", "hk@naver.com"};
13 namecard z={ "김동식", "기획A팀대리", "318-3961", "ds@naver.com"};
14
15 cout<<" 이름 ₩t 직업 ₩t₩t 연락처 ₩t 이메일 ";
16 cout<<"₩n ========================================";
17 cout<<"₩n "<<x.name <<"₩t"<< x.job <<"₩t"<< x.tel <<"₩t"<< x.email;
18 cout<<"₩n "<<y.name <<"₩t"<< y.job <<"₩t"<< y.tel <<"₩t"<< y.email;
19 cout<<"₩n "<<z.name <<"₩t"<< z.job <<"₩t"<< z.tel <<"₩t"<< z.email;
20 cout<<"₩n ========================================₩n";
21 }
```



01 구조체의 이해

구조체 단위로 값 대입하기

- 필드와 레코드 개념을 사용해서 구조체를 사용하는 이유를 알아보자.
  - 명함 관리에 필요한 항목 즉, 이름, 직장명, 연락처, 이메일을 '필드'라 한다. 이런 필드들은 개별적으로 의미 있는 단위다. 하지만 예로, '김주현'이란 특정 사람에 대한 모든 정보가 필요하다면 항목(필드) 4개를 한꺼번에 읽어올 수 있어야 하는데, 이렇게 항목 4개를 한꺼번에 묶은 것을 '레코드'라 한다. 즉, 필드가 모여서 레코드가 된다.

  - 각각의 항목(필드)을 개별적으로 사용할 경우도 있지만 항목 4개(레코드)를 한꺼번에 사용할 수 있어야 하는데, C++에서는 구조체가 레코드에 해당되며 멤버가 필드와 동일한 개념으로 사용된다. 즉, 구조체를 정의해서 프로그램을 작성하면 레코드 단위로도 작업할 수 있다.

  - 구조체 변수에 저장된 값을 다른 구조체 변수에 저장하고자 할 때 다음과 같이 표현할 수 있다. 대입 연산자로 동일한 구조체로 선언된 변수에 대해서 구조체 단위로 값을 복사할 수 있기 때문이다. 만약 이것이 불가능했다면 일일이 멤버변수, 즉 필드 단위로 값을 복사해야 했을 것이다.

```
namecard x={ "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"};
namecard y;
y = x;
```

=

```
strcpy( y.name , x.name);
strcpy( y.job , x.job );
strcpy( y.tel , x.tel );
strcpy( y.email , x.email);
```

예제 8-3. 구조체 단위로 값 복사하기(08_03.cpp)


```
01 #include <iostream>
02 using namespace std;
03 struct namecard{
04 char name[20];
05 char job[30];
06 char tel[20];
07 char email[40];
08 };
09 void main()
10 {
11 namecard x={ "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"};
12 namecard y;
13
14 y = x;
15
16 cout<<"₩t₩t 이름 ₩t 직업 ₩t₩t 연락처 ₩t 이메일 ";
17 cout<<"₩n₩t₩t ========================================";
18 cout<<"₩n<구조체 변수 x>";
19 cout<<"₩t"<<x.name <<"₩t"<< x.job <<"₩t"<< x.tel <<"₩t"<< x.email;
20 cout<<"₩n<구조체 변수 y>";
21 cout<<"₩t"<<y.name<<"₩t"<<y.job<<"₩t"<<y.tel<<"₩t"<< y.email<<"₩n";
22 }
```

함수의 매개변수로 사용하는 구조체

- 프로그램을 작성하다보면 동일한 작업을 반복적으로 기술하는 부분이 생긴다.

```
17   cout<<"₩n"<< x.name <<"₩t"<< x.job <<"₩t"<< x.tel <<"₩t"<< x.email;
18   cout<<"₩n"<< y.name <<"₩t"<< y.job <<"₩t"<< y.tel <<"₩t"<< y.email;
19   cout<<"₩n"<< z.name <<"₩t"<< z.job <<"₩t"<< z.tel <<"₩t"<< z.email;
```

- 중복되는 문장을 여러 번 기술하기보다는 그 문장을 함수로 작성하고 반복되는 부분에서 함수를 호출해주면 훨씬 간단할 것이다. main 함수에서 선언되어 있는 구조체 변수를 함수에서 출력하려면 구조체가 함수의 매개변수가 된다.

```
structPrn(x);
structPrn(y);
structPrn(z);
```

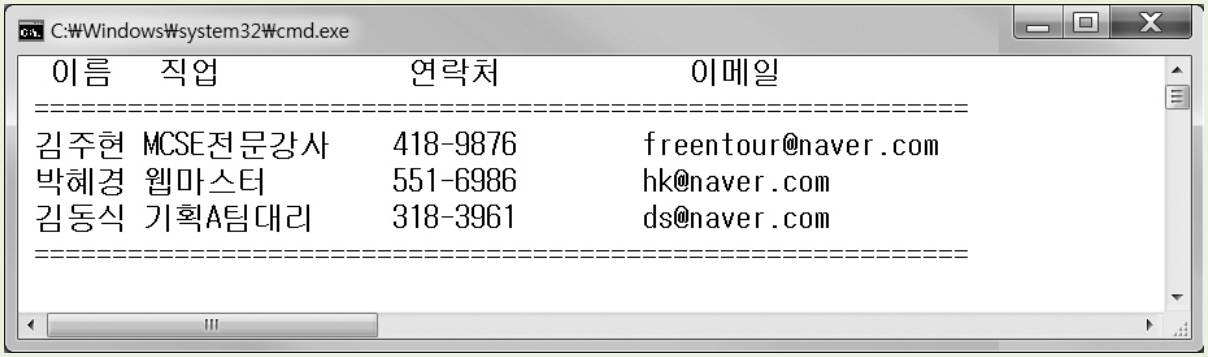
- 구조체 변수 x를 실 매개변수로 적어 주면 structPrn 함수가 호출될 당시에는 실 매개변수의 값만을 전달하므로 structPrn 함수를 정의한 측의 형식 매개변수가 이를 전달받아야 한다.

```
void structPrn(namecard temp) // 함수의 정의
{
cout<<"₩n"<<temp.name<<"₩t"<<temp.job<<"₩t"<<temp.tel<<" ₩t"<<temp.email;
}
```

- structPrn 함수의 형식 매개변수 temp는 호출 시 넘겨준 값을 저장한다. 형식 매개변수 temp는 함수 호출 시 x를 넘겨주면 x값을, y를 넘겨주면 y값을, z를 넘겨주면 z값을 복사해서 멤버변수 단위로 접근해서 출력한다.

예제 8-4. 값에 의한 전달 방식으로 구조체를 매개변수로 사용하는 함수(08_04.cpp)

교재의 소스코드 참고



구조체를 반환값으로 갖는 함수

- 키보드에서 입력받은 값을 구조체 변수에 저장하려고 다음과 같이 프로그램을 작성했다.

```
11  namecard man;
12
13  cout<<" 이름을 입력하세요=>";
14  cin>>man.name;
15  cout<<" 직업을 입력하세요=>";
16  cin>>man.job;
17  cout<<" 연락처를 입력하세요=>";
18  cin>>man.tel;
19  cout<<" 이메일을 입력하세요=>";
20  cin>>man.email;
```

- 반복되는 내용이 있다면 함수로 작성하면 프로그램이 간단해진다. 이제 구조체 값을 입력하는 부분도 함수로 작성해 보자.

01 구조체의 이해

- 함수 structPrn은 형식 매개변수가 실 매개변수 값을 전달받아 출력했다. 함수의 매개변수는 호출한 함수가 호출당한 함수 쪽으로 값을 전달하기만 하면 되므로 단 방향(한쪽)으로만 통신이 일어난다.

```
void main()
{
  namecard x={"김동식", "기획A팀대리"
    "318-3961", "ds@naver.com"};

  structPrn( x );
}
```

구조체 변수 x의 값을 형식 매개변수 temp가 복사한다.

```
void structPrn(namecard temp)
{
    cout<<"₩n"<<temp.name;
...
}
```

- 구조체값을 입력하는 부분을 함수로 구현할 경우에는 호출당한 함수가 호출한 함수로 값을 반환해야 하므로 구조체값을 전달해야 하는 방향이 반대가 되어야 한다. 다음은 키보드에서 입력한 값을 구조체에 저장하기 위해 함수로 작성한 후 이를 호출한 문장이다.

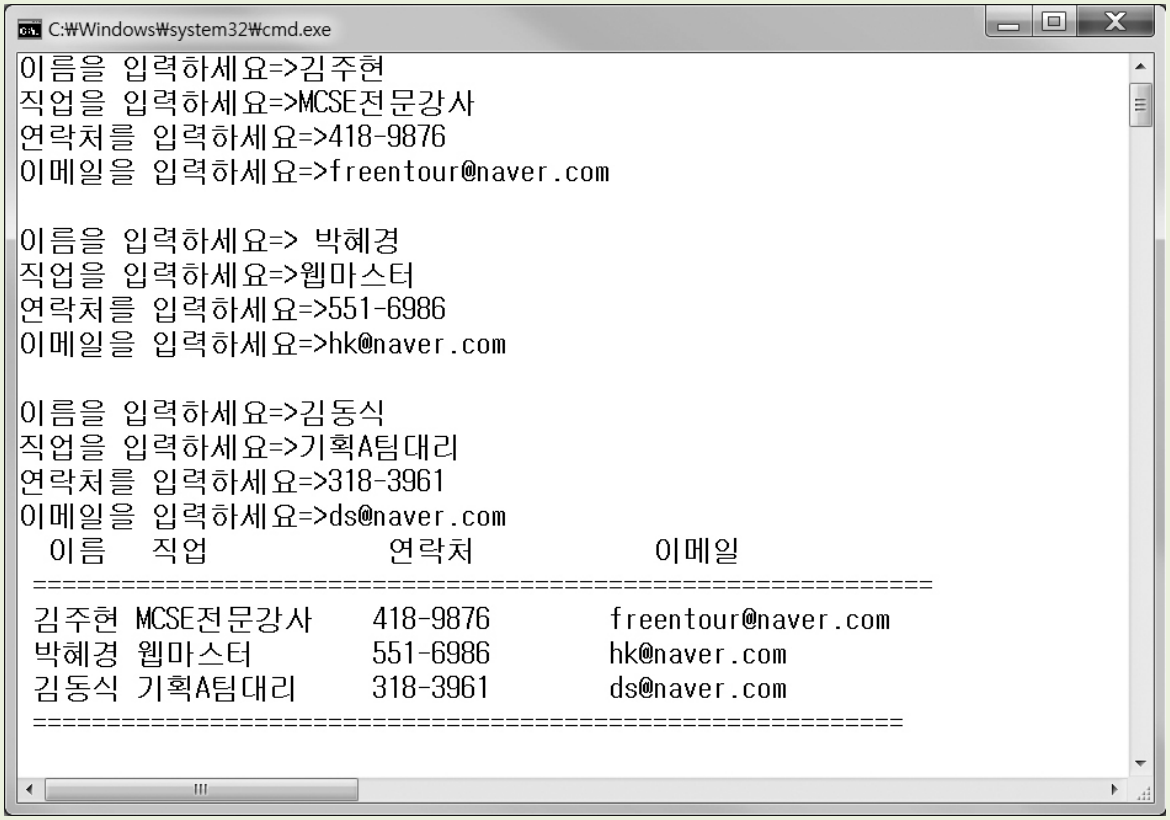
```
void main()
{
  namecard a;

  a=structInput();
}
```

```
namecard structInput()
{
  namecard temp;
  cin>>temp.name;
        ...
  return temp;
}
```

예제 8-5. 구조체 단위의 입력 함수 작성하기(08_05.cpp)

교재의 소스코드 참고



02 구조체 포인터와 배열

구조체 변수와 포인터의 관계

- 포인터 변수를 선언할 때의 구조는 자료형 다음에 *를 기술한 후 포인터 변수명을 적는다. 구조체 포인터인 경우에는 자료형으로 구조체를 기술하면 된다.

포인터 변수 선언 기본 형식

```
자료형 * 포인터 변수;
```

- 구조체 포인터도 구조체 변수를 선언할 때처럼 구조체 선언문이 먼저 나와야 한다.

```
struct namecard{
        char name[20];
        char job[30];
        char tel[20];
        char email[40];
};
namecard *p;
```

구조체 선언

구조체 포인터 선언

- 포인터 변수 p는 특정 기억공간의 주소를 저장하는 용도로 사용되기 때문에 포인터 변수를 선언만 하고 곧바로 사용하면 실행 도중에 에러가 발생한다. 포인터 변수는 반드시 대입 연산자를 이용해서 특정 주소를 저장하고 있어야 하며, 이때 저장할 주소는 namecard 구조체 변수의 주소가 되어야 한다.

02 구조체 포인터와 배열


```
namecard x={ "전수빈",
             "치과의사",
             "356-0868",
             "eldy@naver.com"
           };
p=&x;
```

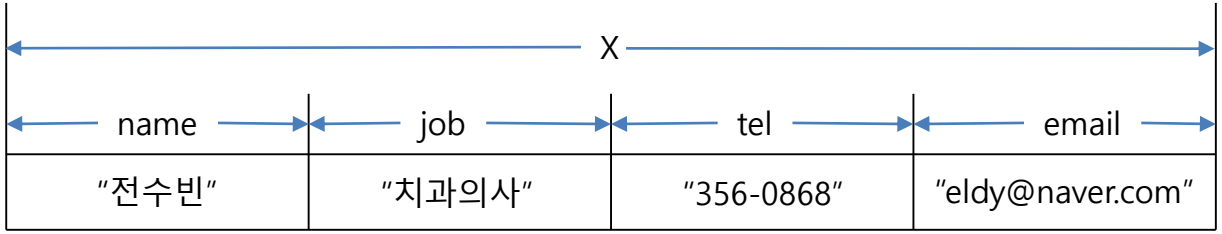
구조체 변수의 선언 및 초기화

구조체 포인터에 구조체 변수의 주소값 저장

■ 구조체 변수와 포인터의 메모리 할당

```
cout<<"sizeof(x) = > " << sizeof(x);
cout<<"sizeof(p) = > " << sizeof(p);
```

- 구조체 변수 x의 메모리 크기를 출력해보면 110바이트다. 이는 실질적인 값을 저장하는 기억공간이기 때문이다. 반면 포인터 변수 p는 4바이트인데, 이는 구조체 변수 x를 가리키기 위한 주소값만을 저장하기 때문이다.



02 구조체 포인터와 배열

- 구조체 포인터 변수는 * 연산자로 해당 위치의 구조체 전체를 얻을 수 있다. 다음과 같이 표현하면 . 연산자가 우선순위가 높으므로 p에 대한 멤버변수 name을 먼저 찾으려 하므로 에러가 발생한다.

```
*p.name; // 에러 발생
```

- 따라서 포인터 변수 p가 가리키는 구조체를 먼저 얻어야 하므로 *p를 먼저 연산할 수 있도록 괄호 연산자를 사용해서 다음과 같이 표현해야 한다.

```
(*p).name;
```

[표 8-1] **구조체와 멤버참조 연산자**

종류	사용법
. 연산자	구조체 변수명.멤버
-> 연산자	구조체 포인터 변수명 -> 멤버

예제 8-6. 구조체 변수와 포인터의 관계 알아보기(08_06.cpp)

교재의 소스코드 참고

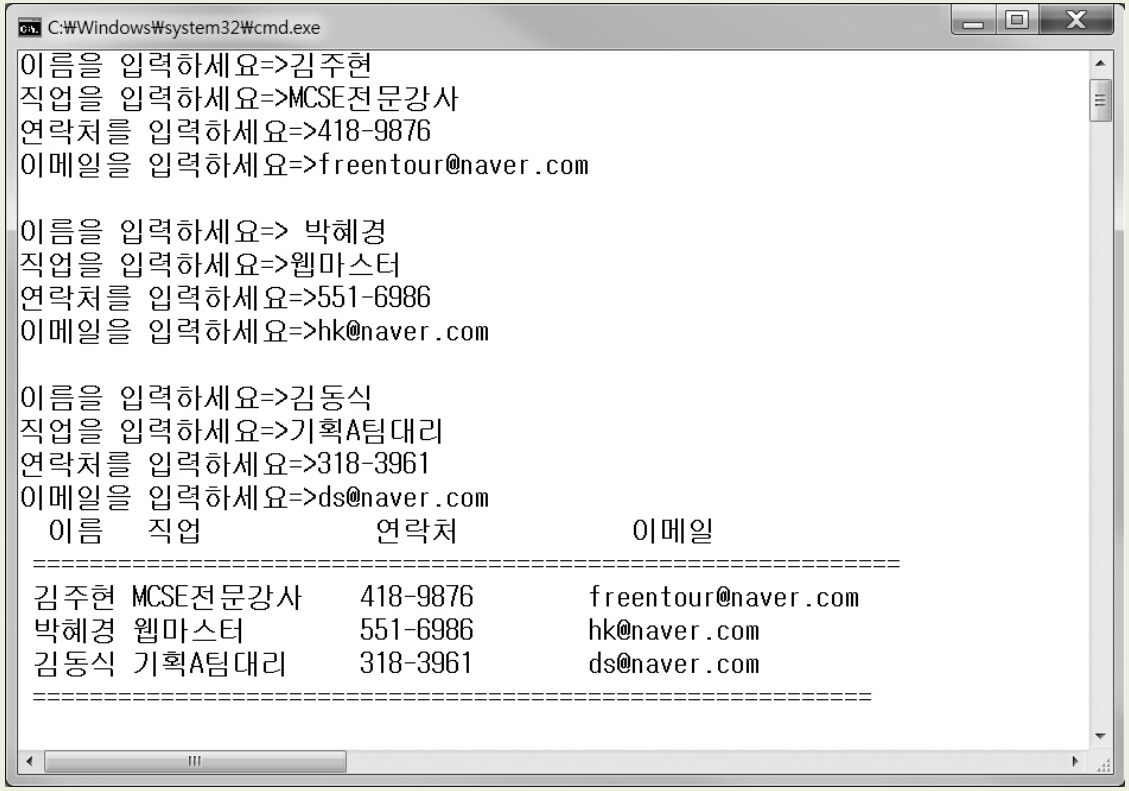
02 구조체 포인터와 배열

구조체 포인터의 용도

- 구조체를 매개변수로 하는 '주소에 의한 전달 방식'의 함수
  - 키보드에서 입력받은 값을 구조체 변수에 저장하는 함수를 작성하되 구조체를 형식 매개변수로 하는 함수를 작성해 보자. 이 함수는 '값에 의한 전달 방식'으로는 불가능하므로 '주소에 의한 전달 방식'으로 작성해야 한다. 함수 호출 후에 실 매개변수에 기술한 구조체 변숫값이 변경되어야 하기 때문이다. '주소에 의한 전달 방식'으로 함수를 정의하면 main함수에서 선언한 변수를 제어할 수 있다.

예제 8-7. 구조체 포인터를 매개변수로 사용하는 함수 작성하기(08_07.cpp)

교재의 소스코드 참고

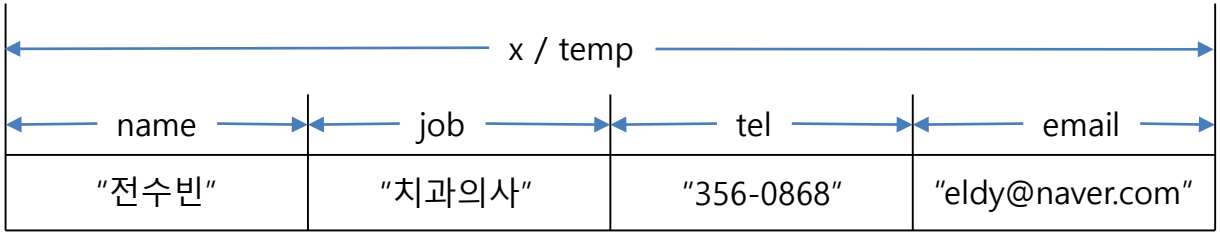


02 구조체 포인터와 배열

- 구조체를 매개변수로 하는 '참조에 의한 전달 방식'의 함수
  - 참조변수는 이미 할당되어진 기억공간을 다른 이름으로 접근할 수 있도록 별칭만을 지정해 주는 것이다.

```
namecard x;
namecard &temp = x;
```

  - x는 일반 구조체 변수로 기억공간 110바이트가 할당된다. 참조 변수(temp)는 이미 할당된 기억공간(x)을 다른 이름(temp)으로 접근할 수 있도록 하며 참조 변수 선언문에 의해서는 별도의 메모리 할당이 일어나지 않는다.



  - main 함수에서 지역 변수로 선언된 x는 참조 변수(별칭) 형태로 선언한 함수의 형식 매개변수 temp로 접근하게 되면 temp가 x의 별칭이므로 동일한 기억공간에 접근하게 된다. 그리고 입력 함수에서 temp로 멤버변수 값을 변경하면 main 함수에 선언된 구조체 변수 x의 값도 변경된다.

```
void structInput(namecard &temp )
```

  - 함수의 실 매개변수 값을 변경하려고 참조 변수를 사용하면 선언할 때만 & 기호를 붙이면 되고 함수 내부에서는 일반 변수와 동일하게 사용하면 되므로 포인터 변수를 사용하는 것보다 훨씬 편리하고 간단하다.

02 구조체 포인터와 배열

구조체 배열

- 동일한 자료형으로 선언된 기억공간이 같은 목적으로 사용될 경우 사용하는 것이 배열이다.

```
17 cout<<"₩n"<< x.name <<"₩t"<< x.job <<"₩t"<< x.tel <<"₩t"<< x.email;
18 cout<<"₩n"<< y.name <<"₩t"<< y.job <<"₩t"<< y.tel <<"₩t"<< y.email;
19 cout<<"₩n"<< z.name <<"₩t"<< z.job <<"₩t"<< z.tel <<"₩t"<< z.email;
```

- x, y, z는 namecard라는 동일한 자료형으로 선언한 구조체 변수다. 잃게 동일한 용도로 사용되는 변수들을 배
- 열로 선언해서 사용하면 간단하게 표현할 수 있다.

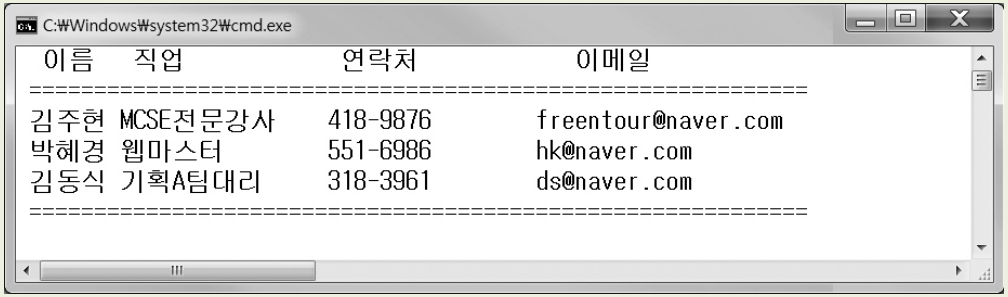
```
namecard x[3] = { { "김주현", "MCSE전문강사", "418-9876", "freentour@naver.com" },
                  { "박혜경", "웹마스터", "551-6986", "hk@naver.com" },
                  { "김동식", "기획A팀대리", "318-3961", "ds@naver.com" }
                };
```

- 구조체 배열로 할당된 기억공간을 그림으로 그려보면 다음과 같다.

	name	job	tel	email
x[0]	"김주현"	"MCSE전문강사"	"418-9876"	"freentour@naver.com"
x[1]	"박혜경"	"웹마스터"	"551-6986"	"hk@naver.com m"
x[2]	"김동식"	"기획A팀대리"	"318-3961"	"ds@naver.com m"

예제 8-9. 구조체 배열 사용하기(08_09.cpp)


```
01 #include <iostream>
02 using namespace std;
03 struct namecard{
04 char name[20];
05 char job[30];
06 char tel[20];
07 char email[40];
08 };
09 void main()
10 {
11 namecard x[3]={{"김주현", "MCSE전문강사", "418-9876", "freentour@naver.com"},
12 {"박혜경", "웹마스터", "551-6986", "hk@naver.com"},
13 {"김동식", "기획A팀대리", "318-3961", "ds@naver.com"}
14 };
15 cout<<" 이름 ₩t 직업 ₩t₩t 연락처 ₩t 이메일 ₩n";
16 cout<<"=======================================₩n";
17 for(int i = 0 ; i < 3; i++)
18 cout<<x[i].name<<"₩t"<<x[i].job<<"₩t"
19 <<x[i].tel<<"₩t"<<x[i].email<<"₩n ";
20 cout<<"=======================================₩n";
21 }
```



03 공용체와 열거형

공용체

- 공용체는 구조체처럼 다양한 데이터형으로 구성된 항목 여러 개가 모여 집단 항목을 이루는 복합 자료형이다. 공용체는 모든 멤버변수가 동일한 기억공간에 중복되어 할당된다. 이러한 공용체를 선언하려면 예약어 union을 사용해야 한다.

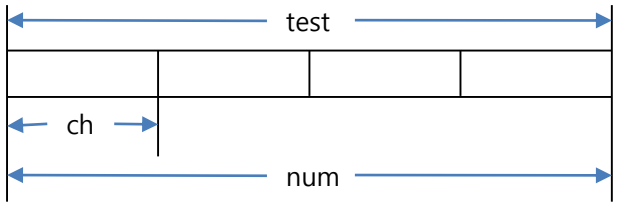
공용체 선언 예

```
union u_data{
  char ch; // 문자형 데이터 멤버
  int num; // 정수형 데이터 멤버
};
```

- 선언된 공용체는 템플릿만 만든 것이기 때문에 실질적인 메모리 할당은 하지 않는다. 메모리 할당은 공용체로 변수선언을 한다.

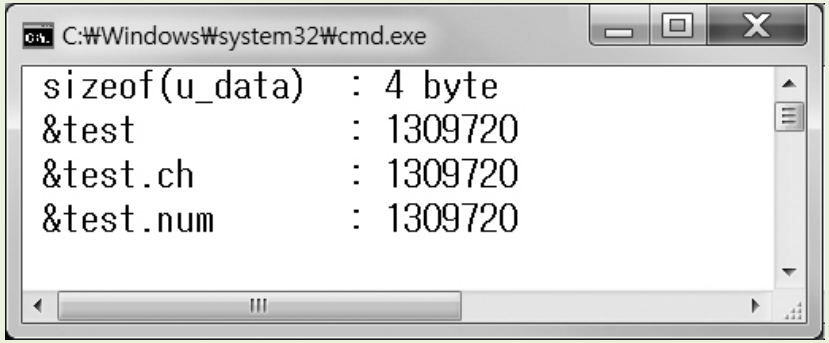
```
u_data test;
```

- 공용체는 멤버 중에서 가장 큰 자료형의 크기로 메모리가 할당된다. 그리고 멤버들이 동일한 시작주소를 갖도록 메모리가 할당된다.



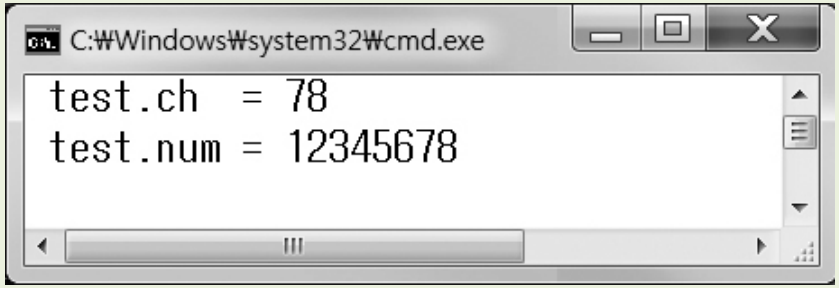
예제 8-10. 공용체 분석하기(08_10.cpp)


```
01 #include <iostream>
02 using namespace std;
03 // 공용체 정의하기
04 union u_data{
05 char ch; // 문자형 데이터 멤버
06 int num; // 정수형 데이터 멤버
07 };
08
09 void main()
10 {
11 u_data test; // 공용체 변수 선언
12 // 공용체의 크기 출력
13 cout<<" sizeof(u_data)₩t : "<<sizeof(u_data) <<" byte ₩n";
14 // 공용체 변수의 주소
15 cout<<" &test ₩t : " << (int)&test <<"₩n";
16 // 공용체 멤버의 주소
17 cout<<" &test.ch ₩t : " << (int)&test.ch <<"₩n";
18 cout<<" &test.num ₩t : " << (int)&test.num <<"₩n";
19 }
```



예제 8-11. 공용체에 값을 저장한 후 저장된 값을 출력하기(08_11.cpp)


```
01 #include <iostream>
02 using namespace std;
03 // 공용체 정의하기
04 union u_data{
05 char ch; // 문자형 데이터 멤버
06 int num; // 정수형 데이터 멤버
07 };
08
09 void main()
10 {
11 u_data test; // 공용체 변수 선언
12 test.num=0x12345678;
13
14 cout<<" test.ch = "<< hex <<(int) test.ch <<"₩n";
15 cout<<" test.num = "<< hex << test.num <<"₩n";
16 }
```



03 공용체와 열거형

열거형

- 열거형은 예약어 enum을 사용한다. 열거형은 숫자나 색깔과 같이 일정한 패턴이 있는 상수들을 집합으로 갖도록 선언한다. 열거형으로 선언된 변수는 집합에 포함되는 상숫값만을 갖게 된다.

```
enum COLOR { RED, GREEN, BLUE, WHITE, BLACK };
```

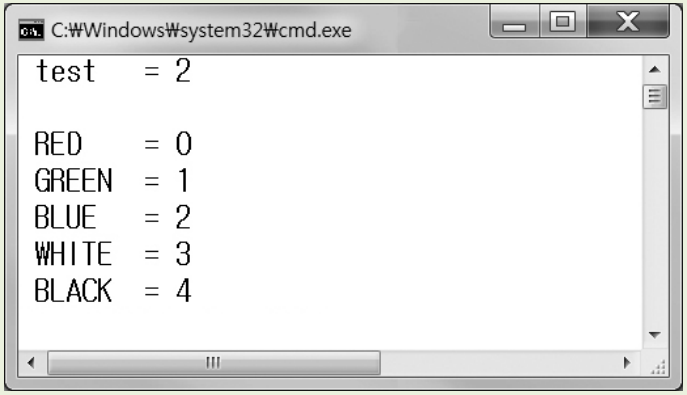
- 열거형 COLOR은 새로운 자료형이다. 그리고 { } 안에 기술된 구성원을 '열거 상수'라고 한다. 처음에 기술된 열거 상수의 값은 0이고 값이 차례로 1씩 증가한다.

- 열거형 COLOR은 템플릿만 정의한 것이기 때문에 메모리는 할당되지 않는다. 메모리를 할당하려면 다음처럼 열거형 변수를 선언해야 한다.

```
enum COLOR test;
```

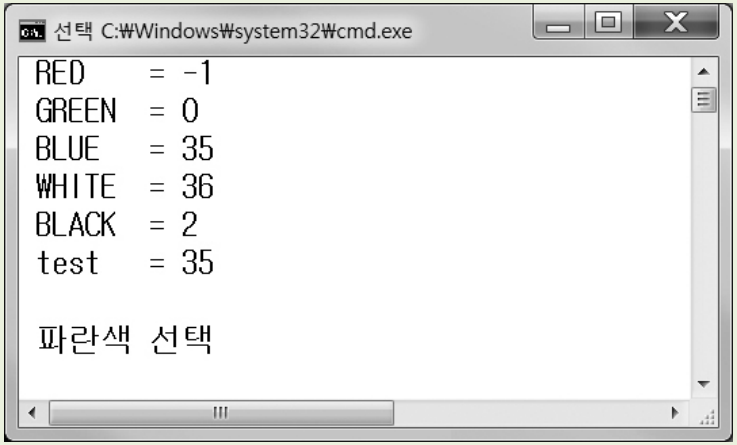
예제 8-12. 열거형 사용하기(08_12.cpp)


```
01 #include <iostream>
02 using namespace std;
03
04 enum COLOR { RED, GREEN, BLUE, WHITE, BLACK };
05 void main()
06 {
07 enum COLOR test;
08
09 test = BLUE;
10 cout<<" test = "<<test <<"₩n₩n"; // test는 정수값 2로 정의되어 있다.
11
12 cout<<" RED = "<< RED<<"₩n";
13 cout<<" GREEN = "<< GREEN<<"₩n";
14 cout<<" BLUE = "<< BLUE<<" ₩n";
15 cout<<" WHITE = "<<WHITE <<" ₩n";
16 cout<<" BLACK = "<< BLACK<<" ₩n";
17 }
```



예제 8-13. 열거 상수에 수치 정하기(08_13.cpp)

교재의 소스코드 참고



04 typedef

- typedef는 이미 사용하고 있는 자료형의 이름을 새롭게 지정할 때 사용하는 예약어로 자료형에 대해서만 적용할 수 있는 명령어다.

  ⓐ typedef int * PTR ⓑ PTR ptr1, ptr2, ptr03;

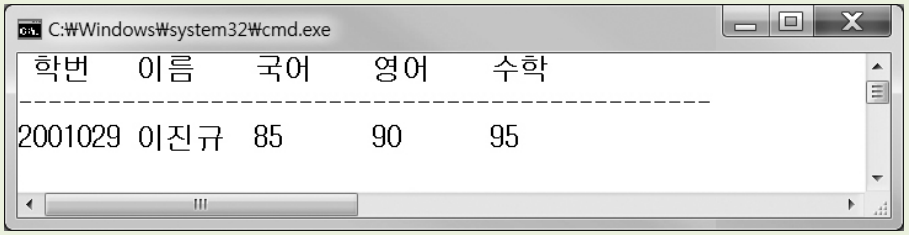
- typedef문을 사용해서 PTR를 int *로 정의해 두었기에 ⓑ와 같이 변수를 선언하면 ptr1, ptr2, ptr03 모두 포인터 변수로 인식한다.

- typedef를 사용하면 구조체 변수를 선언할 때 struct란 예약어를 사용하지 않을 수 있다.

  typedef struct sungjuck SJ;
  SJ s; // struct란 예약어를 사용하지 않고 태그명으로 구조체 변수 선언을 할 수 있다.

예제 8-14. 열거형 사용하기(08_14.cpp)


```
01 #include <iostream>
02 using namespace std;
03 // 성적 관리를 위한 구조체(탬플릿) 정의
04 struct sungjuck{
05 char no[8]; char name[16]; // 학번과 이름
06 int kor, eng, mat, tot; // 국어, 영어 수학 점수, 총점
07 double ave; // 평균
08 char level; // 학점
09 int grade; // 등수
10 };
11
12 typedef struct sungjuck SJ;
13
14 void main()
15 {
16 SJ s={"2001029", "이진규", 85, 90, 95};
17
18 cout<<" 학번 ₩t이름 ₩t국어 ₩t영어 ₩t수학 ₩n";
19 cout<<"------------------------------------------------------ ₩n";
20 cout<<s.no<<"₩t"<<s.name<<"₩t"<<s.kor<<"₩t"
21 <<s.eng<<"₩t"<< s.mat<<"₩n";
22 }
```



04 typedef

- typedef는 구조체 변수를 선언하는 것과 동시에 typedef 선언도 할 수 있다.

❶
```
struct sungjuck{
    char no[8];
    char name[16];
    int kor, eng, mat, tot;
    double ave;
    char level;
    int grade;
};
typedef struct sungjuck SJ;
```

❷
```
typedef struct sungjuck{
    char no[8];
    char name[16];
    int kor, eng, mat, tot;
    double ave;
    char level;
    int grade;
} SJ;
```

❸
```
typedef struct {
    char no[8];
    char name[16];
    int kor, eng, mat, tot;
    double ave;
    char level;
    int grade;
} SJ;
```

- ❶의 문장을 ❷의 문장으로 기술해도 같은 의미가 된다. 그리고 구조체 선언과 동시에 typedef 선언을 할 경우 ❸처럼 struct 다음에 태그명을 생략하고 선언할 수 있다. 그렇지만 태그명을 생략해서 선언할 경우에는 ❸은 ❷에 비해 차이가 있다. ❸은 구조체를 정의할 때 struct 다음에 태그명을 명시하지 않았으므로 오로지 SJ로만 구조체 변수를 선언할 수 있다.