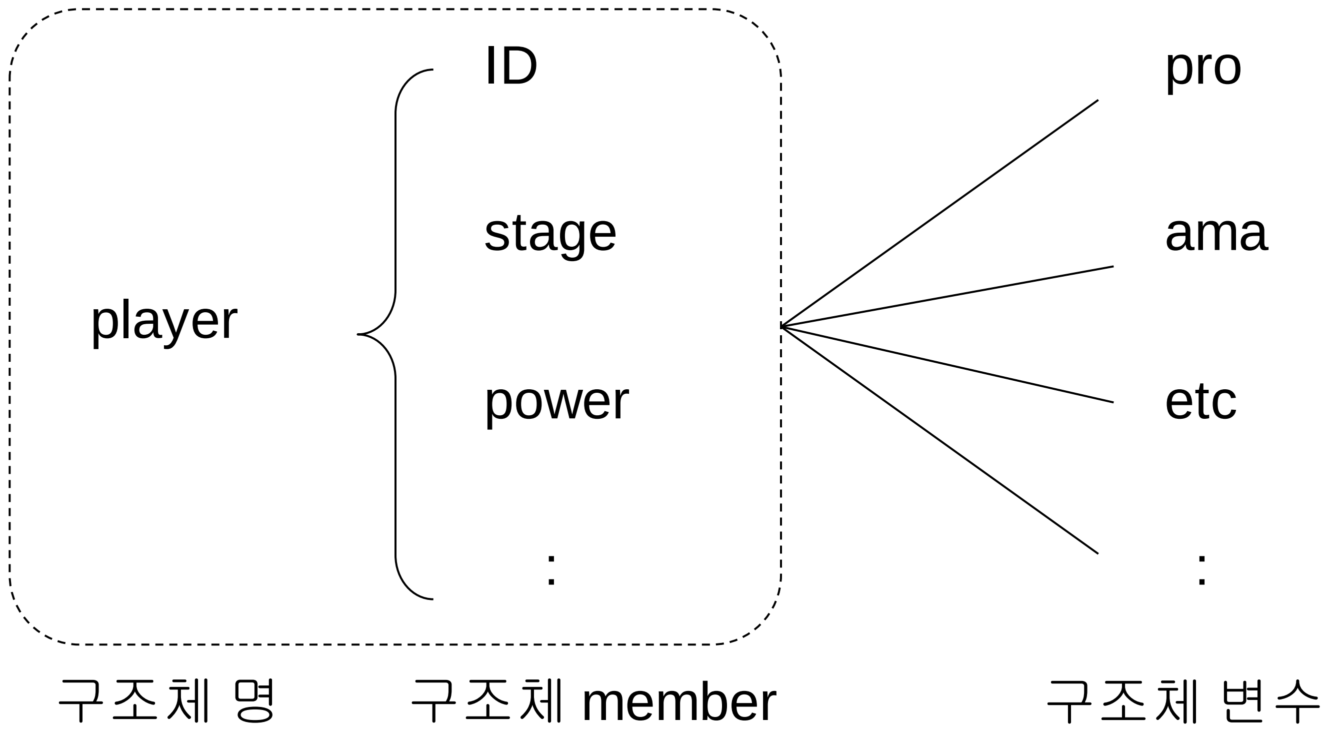
복합데이터형2

구조체(structure)

연관된 data(구조체 member)를 하나로 묶을 수 있는 Data type

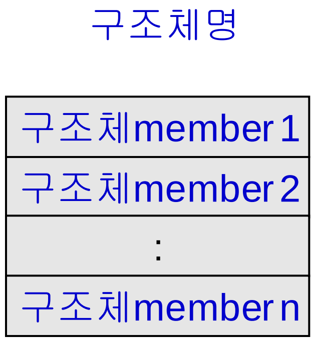


```
struct 구조체명 {
          구조체member 1;
          구조체member 2;
                   :          ;
          구조체member n;
};
```

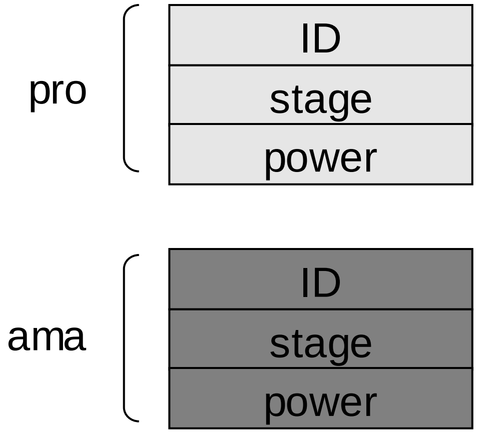
구조체명

구조체member 1
구조체member 2
:
구조체member n

```
struct 구조체명 구조체변수1, 구조체변수2,…;
struct 구조체명 {
            구조체member 1;
            구조체member 2;
                     :           ;
            구조체member n;
} 구조체변수1, 구조체변수2,…;
```

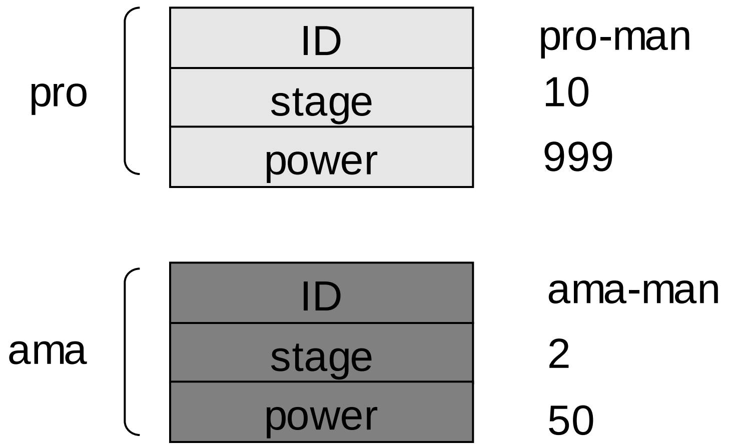


구조체(structure) -초기화-



```
struct player {
        char ID[20];
        int stage;
        double power;
};

struct player pro, ama;
```



~~pro.ID = "pro-man";~~

pro.stage = 10;

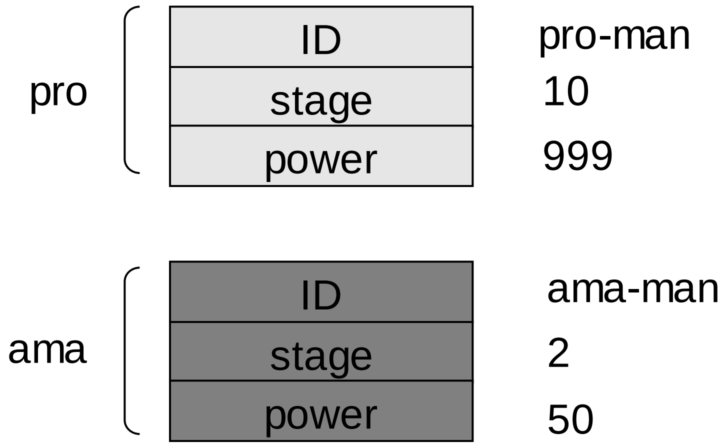
pro.power = 999;

~~ama.ID = "ama-man";~~

ama.stage = 2;

ama.power = 50;

```
struct player pro = {"pro-man", 10, 999},
              ama = {"ama-man", 2, 50};
```



```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

struct player {

              :      ;

} pro = {"pro-man", 10, 999},
  ama = {"ama-man", 2, 50};
```

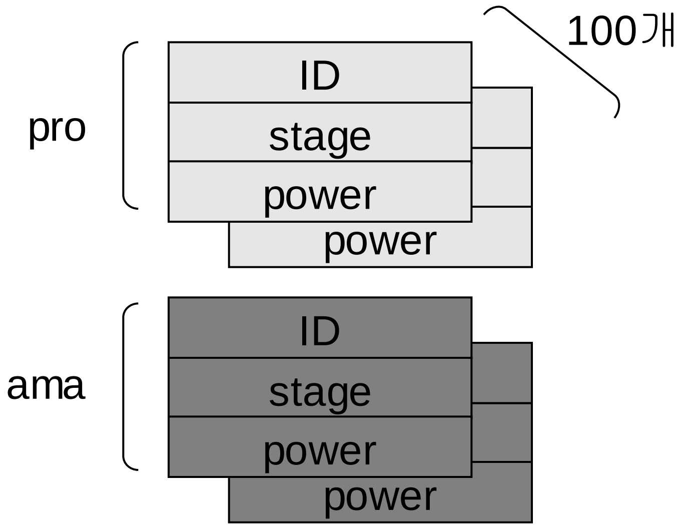
- example -

```
//structure_ini.cpp
#include <iostream>
using namespace std;

struct player {
            char ID[20];
            int stage;
            double  power;
};

void main()
{
  //struct player pro = {"pro-man",  10, 999};
  struct player pro;
  //cin.getline(pro.ID,  20);
  pro.ID[0] = 'p';
  pro.ID[1] = 'r';
  pro.ID[2] = 'o';
  pro.ID[3] = '-';
  pro.ID[4] = 'm';
  pro.ID[5] = 'a';
  pro.ID[6] = 'n';
  pro.ID[7] = '\0';
  pro.stage = 10;
  pro.power  = 999;
  cout << "pro_ID is " << pro.ID << '\n';
  cout << "pro_stage  is " << pro.stage << '\n';
  cout << "pro_power  is " << pro.power  << '\n';
}
```

구조체(structure) -배열-



```
struct player {
        char ID[20];
        int stage;
        double power;
};

struct player pro[100], ama[100];
```

~~pro[0].ID = "~~pro-man";

pro[0].stage = 10;

pro[0].power = 999;

~~ama[3].ID = "~~ama-man";

ama[3].stage = 2;

ama[3].power = 50;

```
struct player pro[100] =
{
   {"pro-man1", 10, 900},
   {"pro-man2", 11, 999}
};
```

- example -

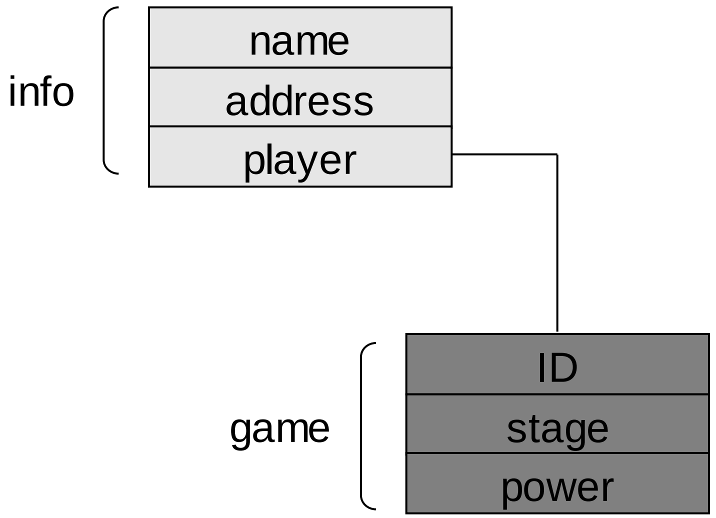
```
//structure_array.cpp
#include <iostream>
using namespace std;

struct player {
            char ID[20];
            int stage;
            double power;
};

void main()
{
  struct player pro[100] =
  {
    {"pro-man1", 10, 900},
    {"pro-man2", 11, 999}
  };

    cout << "pro_ID[0] is " << pro[0].ID << '\n';
    cout << "pro_stage[0] is " << pro[0].stage << '\n';
    cout << "pro_power[0] is " << pro[0].power << '\n';
    cout << "pro_ID[1] is " << pro[1].ID << '\n';
    cout << "pro_stage[1] is " << pro[1].stage << '\n';
    cout << "pro_power[1] is " << pro[1].power << '\n';
}
```

구조체(structure) -중첩(nested structure)-



```
struct game {
        char ID[20];
        int stage;
        double power;
};

struct info {
        char name[30];
        char address[50];
        struct game player;
};
```

```
struct info {
        char name[30];
        char address[50];
        struct {
                char ID[20];
                int stage;
                double power;
        };
};
```

```
struct info pro[100] =
{
   {"unknown", "seoul", "pro-man", 10, 900},
   {"known", "inchon", "super-man", 10, 999},
};
```

- example -

```
//structure_nested.cpp
#include <iostream>
using namespace std;

struct game {
            char ID[20];
            int stage;
            double power;
};
struct info {
            char name[30];
            char address[50];
            struct game player;
};

void main()
{
  struct info pro[100] =
  {
    {"unknown", "seoul", "pro-man", 10, 900},
    {"known", "inchon", "super-man", 10, 999},
  };
  cout << "pro_name[0] is " << pro[0].name << '\n';
  cout << "pro_address[0] is " << pro[0].address << '\n';
  cout << "pro_ID[0] is " << pro[0].player.ID << '\n';
  cout << "pro_stage[0] is " << pro[0].player.stage << '\n';
  cout << "pro_power[0] is " << pro[0].player.power << '\n';
  cout << "pro_name[1] is " << pro[1].name << '\n';
  cout << "pro_address[1] is " << pro[1].address << '\n';
  cout << "pro_ID[1] is " << pro[1].player.ID << '\n';
  cout << "pro_stage[1] is " << pro[1].player.stage << '\n';
  cout << "pro_power[1] is " << pro[1].player.power << '\n';
}
```

공용체(union)

각 member가 한 개의 memory를 공유

```
struct player {                                    union player {
        char ID;          ——   8bit   ————————          char ID;
        int stage;        ——   32bit  ————————          int stage;
        double power;     ——   64bit  ————————          double power;
};                                                 };
```

104 bit memory 를 변수에 할당

64 bit memory 를 변수에 할당

구조체 member에 값들을 넣으면 마지막 대입한 값만 남게 됨

```
union player pro;

pro.ID = 'A';
pro.stage = 7;
pro.power = 999;
```

- example -

```
//union.cpp
#include <iostream>
using namespace std;

union player {
            char ID;
            int stage;
            double  power;
};

void main()
{
  union player pro;
  pro.ID = 'p';
  pro.stage = 10;
  pro.power = 999;

  cout << "pro_ID is " << pro.ID << '\n';
  cout << "pro_stage is " << pro.stage << '\n';
  cout << "pro_power is " << pro.power << '\n';
}
```