C++ 클래스 &sscanf()

송허이

클 래 스 란 ?

≒

```c
struct student {
    int age;
    char name[20];
    int s_id;
}

struct student s1 = {22, "허송이", 91813274};
struct student s2 = {22, "홍길동", 91813275};
```

클래스

구조체

접 근 제 어 자

형식

```
class 클래스명 {
[ private:
protected:
public:]
   [멤버변수 선언;]
   [멤버함수 선언;]
};
```

“접근 제어자”

주요 정보는 외부에 참조할 수 없도록
보호하여 안전한 프로그램이 가능

보통, 데이터(멤버변수)를 private, protected로 하며,
멤버함수는 public으로 하여 멤버변수에 접근하도록 한다.

생 성 자

객체를 초기화하는 방법 중 하나로, 객체 생성을 준비하는 함수이다.

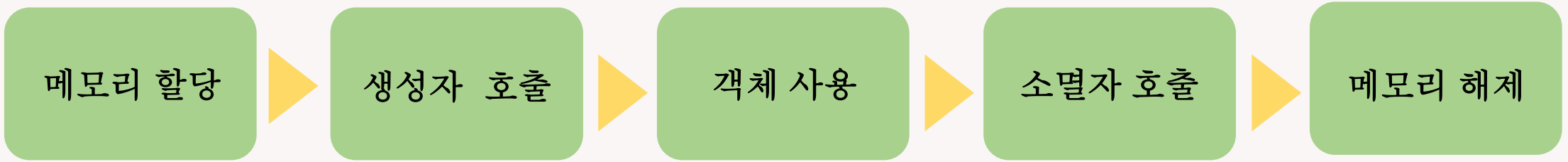
```
class Point {
private:
        int x;
        int y;
public:
        Point();
        Point(int xpos, int ypos);
};
```

```
Point::Point()
{
        x=0;
        y=0;
        cout << "디폴트 생성자 "
<< endl;
}
Point p;
```

```
Point::Point(int xpos, int ypos)
{
        x=xpos;
        y=ypos;
        cout << "인자 있는 생성자 "
<< endl;
}
Point obj(5, 10);
```

객체의 생성과 소멸

객체 생성



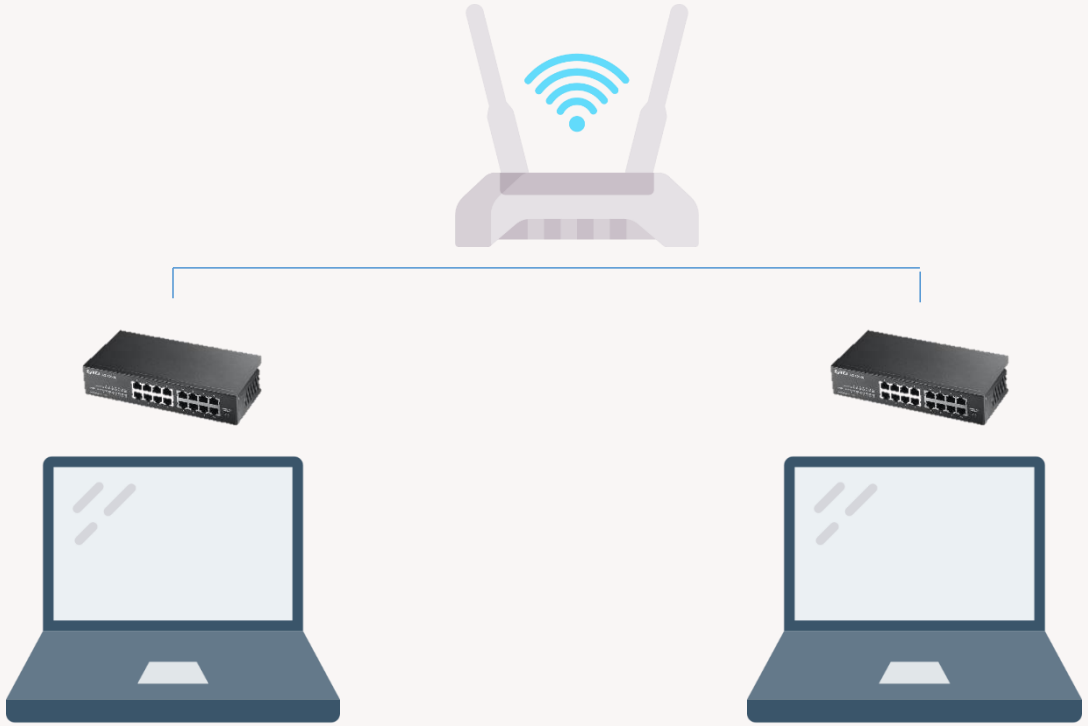
멤버변수를 초기화하거나
필요한 자원 할당

생성자에서 할당한
자원 해제

객체 소멸

클	래	스	예	제										

IP 주소를 기반으로 MAC 주소를 알아오는 ARP Protocol 구현



클 래 스 예 제

Arp request packet을 보낼 때 ip주소와 mac 주소를 패킷에 저장하는 기능 필요 ⟶ 클래스화

```
class Mac final {
public:
    static const int SIZE = 6;

    //
    // constructor
    //
    Mac() {}
    Mac(const uint8_t* r) { memcpy(this->mac_, r, SIZE); }
    Mac(const std::string r);
protected:
    uint8_t mac_[SIZE];
};
```

클 래 스 예 제

Arp request packet을 보낼 때 ip주소와 mac 주소를 패킷에 저장하는 기능 필요 ⟶ 클래스화

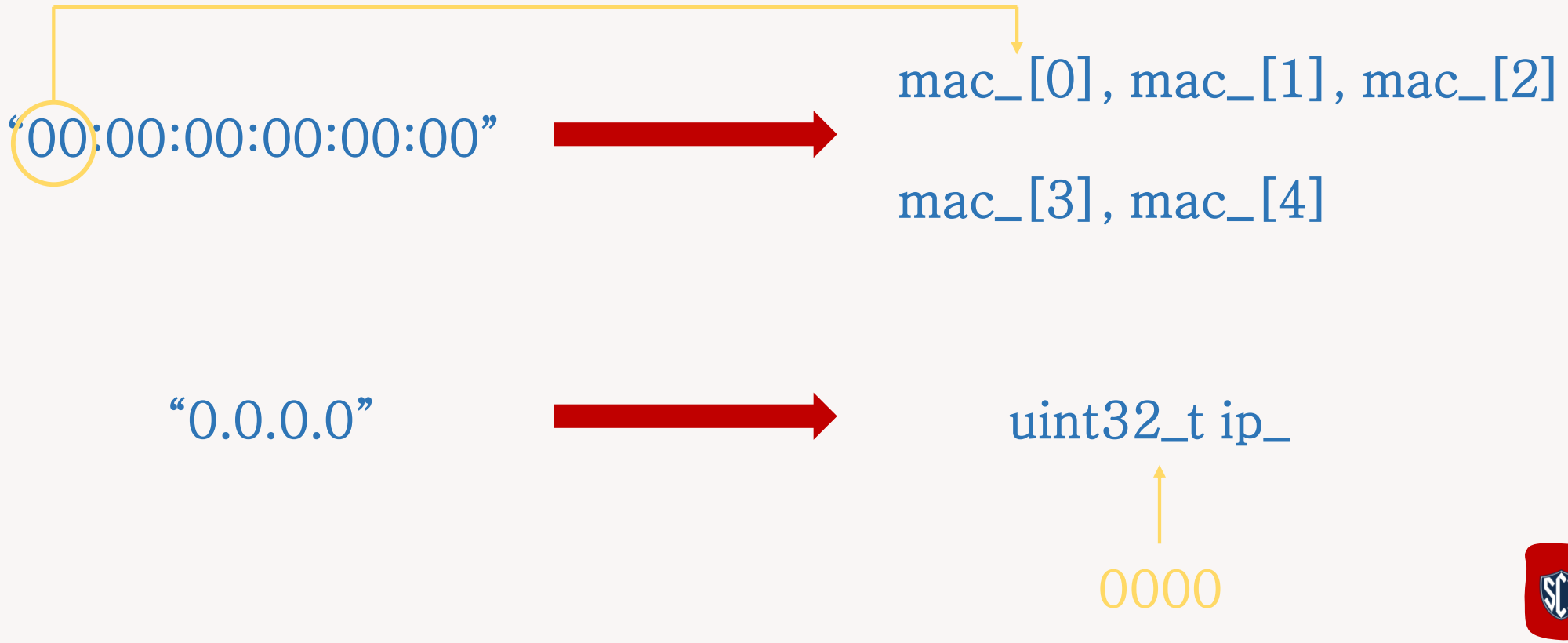
```
class Ip final {
public:
    static const int SIZE = 4;

    //
    // constructor
    //
    Ip() {}
    Ip(const uint32_t r) : ip_(r) {}
    Ip(const std::string r);

protected:
    uint32_t ip_;
};
```

sscanf()

문자열에서 데이터를 형식 문자열(format)에서 지정하는 바에 따라 읽어와 format 다음 인자들이 가리키는 메모리 공간에 저장한다.

```
int sscanf(const char* str, const char* format, ...);
```

s	s	c	a	n	f	()								



s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

① str

: sscanf 함수가 데이터를 얻어올 문자열 “00:00:00:00:00:00”

s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

② format

: 공백 문자는 비-공백 문자를 읽어 들이기 전까지 무시하며,
%를 제외한 비-공백 문자를 읽어 들여 형식과 비교한다.

“%02X:%02X:%02X:%02X:%02X:%02X:”

s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

② format “%02X:%02X:%02X:%02X:%02X:%02X:”

%[*][폭(width)][한정자(modifiers)]타입(type)

1) *: 데이터를 받아들이지만 무시하라는 의미이다.

ex: scanf(“%*d%d” , i, j);

s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

② format “%02X:%02X:%02X:%02X:%02X:%02X:”

%[*][폭(width)][한정자(modifiers)]타입(type)

2) 폭: stdin에서 읽어 들일 최대 문자 수를 지정

ex: scanf(“%10s” , str);

s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

② format “%02X:%02X:%02X:%02X:%02X:%02X:”

%[*][폭(width)][한정자(modifiers)]타입(type)

3) 한정자: 입력 받는 데이터의 크기를 지정

s	s	c	a	n	f	()								

“00:00:00:00:00:00” → 00 00 00 00 00 00

```
int sscanf(const char* str, const char* format, ...);
```

② format “%02X:%02X:%02X:%02X:%02X:%02X:”

%[*][폭(width)][한정자(modifiers)]타입(type)

4) 타입: 어떠한 값 만을 읽어 들여야 할 지 지정

ex: d – 십진법,
x – 16진법, …

s	s	c	a	n	f	()								

```
int sscanf(const char* str, const char* format, ...);
```

③ 부수적 인자

: 형식 문자열의 정의된 순서대로 각 형식 지정자는 이에 대응하는 인자가 가리키는 메모리 공간에 데이터를 집어넣는다.

sscanf(r.c_str(), “%02X:%02X:%02X:%02X:%02X:%02X”, &a, &b, &c, &d, &e, &f);

s	s	c	a	n	f	()								

```cpp
#include "mac.h"

Mac::Mac(const std::string r) {
    unsigned int a, b, c, d, e, f;
    int res = sscanf(r.c_str(), "%02X:%02X:%02X:%02X:%02X:%02X", &a, &b, &c, &d, &e, &f);
    if (res != SIZE) {
        fprintf(stderr, "Mac::Mac sscanf return %d r=%s\n", res, r.c_str());
        return;
    }
    mac_[0] = a;
    mac_[1] = b;
    mac_[2] = c;
    mac_[3] = d;
    mac_[4] = e;
    mac_[5] = f;
}
```

s	s	c	a	n	f	()								

```cpp
#include "ip.h"
#include <cstdio>

Ip::Ip(const std::string r) {
    unsigned int a, b, c, d;
    int res = sscanf(r.c_str(), "%u.%u.%u.%u", &a, &b, &c, &d);
    if (res != SIZE) {
        fprintf(stderr, "Ip::Ip sscanf return %d r=%s\n", res, r.c_str());
        return;
    }
    ip_ = (a << 24) | (b << 16) | (c << 8) | d;
}
```

클 래 스 & s s c a n f ()

```
Mac get_mac(char* dev)
{
    struct ifreq s;
    int fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_IP);

    strcpy(s.ifr_name, dev);
    if (0 != ioctl(fd, SIOCGIFHWADDR, &s)) {
        printf("Can't get mac address");
    }
    return Mac((unsigned char*) s.ifr_addr.sa_data);
}
rqpacket.ether.dmac_ = Mac("FF:FF:FF:FF:FF:FF");
rqpacket.ether.smac_ = get_mac(dev);
```

```
class Mac final {
public:
    static const int SIZE = 6;

    //
    // constructor
    //
    Mac() {}
    Mac(const uint8_t* r) { memcpy(this->mac_, r, SIZE); }
    Mac(const std::string r);

protected:
    uint8_t mac_[SIZE];
};
```

클 래 스 & s s c a n f ()

```
rqpacket.ether.dmac_ = Mac("FF:FF:FF:FF:FF:FF");
rqpacket.ether.smac_ = get_mac(dev);
```

```
class Mac final {
public:
    static const int SIZE = 6;

    //
    // constructor
    //
    Mac() {}
    Mac(const uint8_t* r) { memcpy(this->mac_, r, SIZE); }
    Mac(const std::string r);

protected:
    uint8_t mac_[SIZE];
};
```

```
#include "mac.h"

Mac::Mac(const std::string r) {
    unsigned int a, b, c, d, e, f;
    int res = sscanf(r.c_str(), "%02X:%02X:%02X:%02X:%02X:%02X", &a, &b,
    if (res != SIZE) {
        fprintf(stderr, "Mac::Mac sscanf return %d r=%s\n", res, r.c_str
        return;
    }
    mac_[0] = a;
    mac_[1] = b;
    mac_[2] = c;
    mac_[3] = d;
    mac_[4] = e;
    mac_[5] = f;
}
```

Q & A