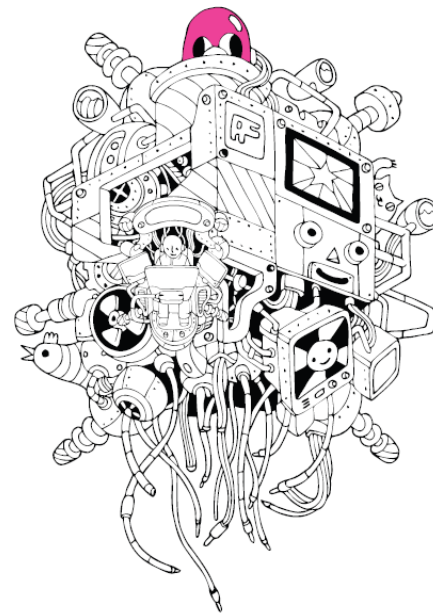


윤성우의 열혈 C++ 프로그래밍



윤성우 저 열혈강의 C++ 프로그래밍 개정판

Chapter 06. friend와 static 그리고 const

윤성우의 열혈 C++ 프로그래밍



Chapter 06-1. const와 관련해서 아직
못다한 이야기

윤성우 저 열혈강의 C++ 프로그래밍 개정판

const와 관련해서 아직 못다한 이야기

```
class SoSimple
{
private:
    int num;
public:
    SoSimple(int n) : num(n)
    { }
    SoSimple& AddNum(int n)
    {
        num+=n;
        return *this;
    }
    void ShowData() const
    {
        cout<<"num: "<<num<<endl;
    }
};
```

```
int main(void)
{
    const SoSimple obj(7);
    // obj.AddNum(20);
    obj.ShowData();
    return 0;
}
```

이 객체의 데이터 변경을 허용
하지 않겠다!

const로 선언된 객체를 대상으로는 const로 선언
되지 않는 멤버함수의 호출이 불가능하다.



const와 함수 오버로딩

```
class SoSimple
{
private:
    int num;
public:
    SoSimple(int n) : num(n)
    { }
    SoSimple& AddNum(int n)
    {
        num+=n;
        return *this;
    }
    void SimpleFunc()
    {
        cout<<"SimpleFunc: "<<num<<endl;
    }
    void SimpleFunc() const
    {
        cout<<"const SimpleFunc: "<<num<<endl;
    }
};
```

함수의 **const** 선언 유무는 함수
오버로딩의 조건이 된다!

const 객체 또는 참조자를 대상으로 멤버함수 호출
시 **const** 선언된 멤버함수가 호출된다!

```
SimpleFunc: 2
const SimpleFunc: 7
const SimpleFunc: 2
const SimpleFunc: 7
```

실행결과

```
void YourFunc(const SoSimple &obj)
{
    obj.SimpleFunc();
}

int main(void)
{
    SoSimple obj1(2);
    const SoSimple obj2(7);
    obj1.SimpleFunc();
    obj2.SimpleFunc();
    YourFunc(obj1);
    YourFunc(obj2);
    return 0;
}
```

윤성우의

열혈 C++ 프로그래밍



Chapter 06-2. 클래스와 함수에 대한
friend 선언

윤성우 저 열혈강의 C++ 프로그래밍 개정판

클래스의 friend 선언

```
class Boy
```

```
{
```

```
private:
```

```
int height;
```

```
friend class Girl;
```

```
public:
```

```
Boy(int len) : height(len)
```

```
{ }
```

```
.....
```

```
};
```

Girl 클래스에 대한 friend 선언!

이렇게

```
class Girl
```

```
{
```

```
private:
```

```
char phNum[20];
```

```
public:
```

```
Girl(char * num)
```

```
{
```

```
strcpy(phNum, num);
```

```
}
```

```
void ShowYourFriendInfo(Boy &frn)
```

```
{
```

```
cout<<"His height: "<<frn.height<<endl;
```

```
}
```

```
};
```

Girl이 Boy의 friend로 선언되었으므로,
private 멤버에 직접접근 가능

friend 선언은 private 멤버의 접근을 허용하
는 선언이다.

friend 선언은 정보은닉에 반하는 선언이기 때문에 매우 제한적으로
선언되어야 한다.

함수의 friend 선언

```
class Point
{
private:
    int x;
    int y;
public:
    Point(const int &xpos, const int &ypos) : x(xpos), y(ypos)
    { }
    friend Point PointOP::PointAdd(const Point&, const Point&);
    friend Point PointOP::PointSub(const Point&, const Point&);
    friend void ShowPointPos(const Point&);
};
```

직접 접근 가능

전역함수 대상의 friend 선언

이렇듯 클래스의 특정 멤버함수를 대상으로도 friend 선언이 가능하다.

가능

```
Point PointOP::PointAdd(const Point& pnt1, const Point& pnt2)
{
    opcnt++;
    return Point(pnt1.x+pnt2.x, pnt1.y+pnt2.y);
}
Point PointOP::PointSub(const Point& pnt1, const Point& pnt2)
{
    opcnt++;
    return Point(pnt1.x-pnt2.x, pnt1.y-pnt2.y);
}
```

private 멤버 접근

private 멤버 접근

```
void ShowPointPos(const Point& pos)
{
    cout<<"x: "<<pos.x<<" ";
    cout<<"y: "<<pos.y<<endl;
}
```

private 멤버 접근

윤성우의 열혈 C++ 프로그래밍



Chapter 06-3. C++에서의 static

윤성우 저 열혈강의 C++ 프로그래밍 개정판

C언어에서 이야기한 static

프로그래밍이 시작될 때 생성

- 전역변수에 선언된 static의 의미
 - ➔ 선언된 파일 내에서만 참조를 허용하겠다는 의미
- 함수 내에 선언된 static의 의미
 - ➔ 한번만 초기화되고, 지역변수와 달리 함수를 빠져나가도 소멸되지 않는다.

```
void Counter()
{
    static int cnt;
    cnt++;
    cout<<"Current cnt: "<<cnt<<endl;
}

int main(void)
{
    for(int i=0; i<10; i++)
        Counter();
    return 0;
}
```

실행결과

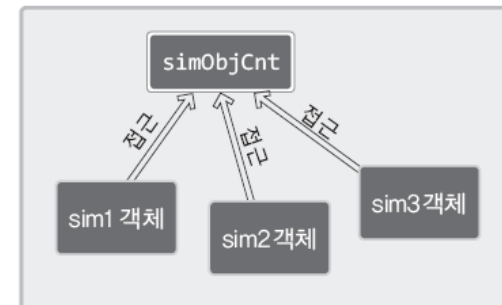
```
Current cnt: 1
Current cnt: 2
Current cnt: 3
Current cnt: 4
Current cnt: 5
Current cnt: 6
Current cnt: 7
Current cnt: 8
Current cnt: 9
Current cnt: 10
```

static 멤버변수(클래스 변수)

```
class SoSimple
{
private:
    static int simObjCnt; // static 멤버변수, 클래스 변수
public:
    SoSimple()
    {
        simObjCnt++;
        cout<<simObjCnt<<"번째 SoSimple 객체"<<endl;
    }
};

int SoSimple::simObjCnt=0; // static 멤버변수의 초기화
```

```
int main(void)
{
    SoSimple sim1;
    SoSimple sim2;
    SoSimple sim3;
    . . . .
}
```



static 변수는 객체 별로 존재하는 변수가 아닌, 프로그램 전체 영역에서 하나만 존재하는 변수이다.

프로그램 실행과 동시에 초기화되어 메모리 공간에 할당된다.

static 멤버 변수의 접근방법

```
class SoSimple
{
public:
    static int simObjCnt;
public:
    SoSimple()
    {
        simObjCnt++;
    }
};
int SoSimple::simObjCnt=0;
```

접근 case 1

```
int main(void)
{
    cout<<SoSimple::simObjCnt<<"번째 SoSimple 객체"<<endl;
    SoSimple sim1;
    SoSimple sim2;
    cout<<SoSimple::simObjCnt<<"번째 SoSimple 객체"<<endl;
    cout<<sim1.simObjCnt<<"번째 SoSimple 객체"<<endl;
    cout<<sim2.simObjCnt<<"번째 SoSimple 객체"<<endl;
    return 0;
}
```

접근 case 2

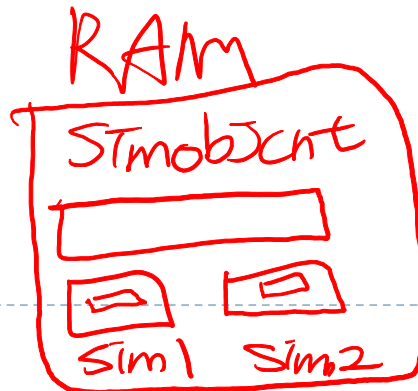
접근 case 3

static 변수를 외부에서 접근 가능하게 하려면, 해당 변수가 public으로 선언되어야 한다.

0번째 SoSimple 객체
2번째 SoSimple 객체
2번째 SoSimple 객체
2번째 SoSimple 객체

실행결과

$sim1.simObjCnt == sim2.simObjCnt$



static 멤버함수

static 멤버변수의 특징과 일치한다.

- 선언된 클래스의 모든 객체가 공유한다.
- public으로 선언이 되면, 클래스의 이름을 이용해서 호출이 가능하다.
- ✖ 객체의 멤버로 존재하는 것이 아니다.

```
class SoSimple
{
private:
    int num1;
    static int num2;
public:
    SoSimple(int n): num1(n)
    { }
    static void Adder(int n)
    {
        num1+=n;    // 컴파일 에러 발생
        num2+=n;
    }
};
int SoSimple::num2=0;
```

객체 생성시만
프로그램 실행시만

객체가 없어도 호출이 가능하므로

static 함수는 객체 내에 존재하는 함수가 아니기 때문에 멤버변수나 멤버함수에 접근이 불가능하다.

static 함수는 static 변수에만 접근 가능하고, static 함수만 호출 가능하다.

일반 함수
호출 X

const static 멤버와 mutable

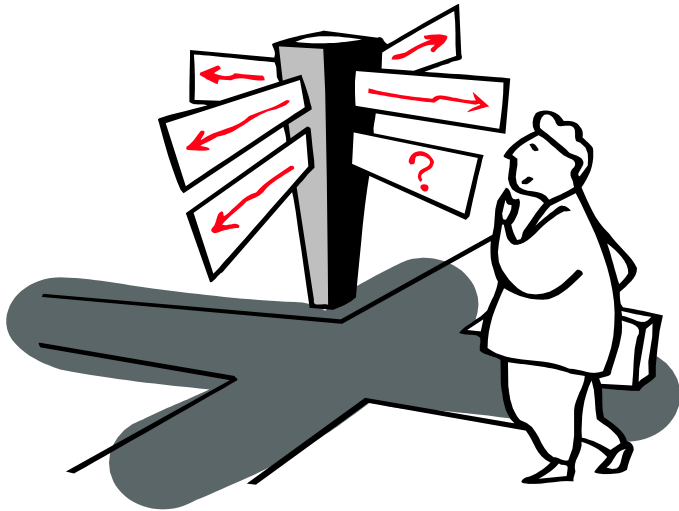
```
class CountryArea
{
public:
    const static int RUSSIA    =1707540;
    const static int CANADA    =998467;
    const static int CHINA     =957290;
    const static int SOUTH_KOREA =9922;
};

int main(void)
{
    cout<<"러시아 면적: "<<CountryArea::RUSSIA<<"km²"<<endl;
    cout<<"캐나다 면적: "<<CountryArea::CANADA<<"km²"<<endl;
    cout<<"중국 면적: "<<CountryArea::CHINA<<"km²"<<endl;
    cout<<"한국 면적: "<<CountryArea::SOUTH_KOREA<<"km²"<<endl;
    return 0;
}
```

const static 멤버변수는, 클래스가 정의될 때 지정된 값이 유지되는 상수이기 때문에, 위 예제에서 보이는 바와 같이 초기화가 가능하도록 문법으로 정의하고 있다.

mutable로 선언된 멤버변수는 const 함수 내에서 값의 변경이 가능하다.

```
class SoSimple
{
private:
    int num1;
    mutable int num2;
public:
    SoSimple(int n1, int n2)
        : num1(n1), num2(n2)
    { }
    void CopyToNum2() const
    {
        num2=num1;
    }
};
```



Chapter 06가 끝났습니다. 질문 있으신지요?