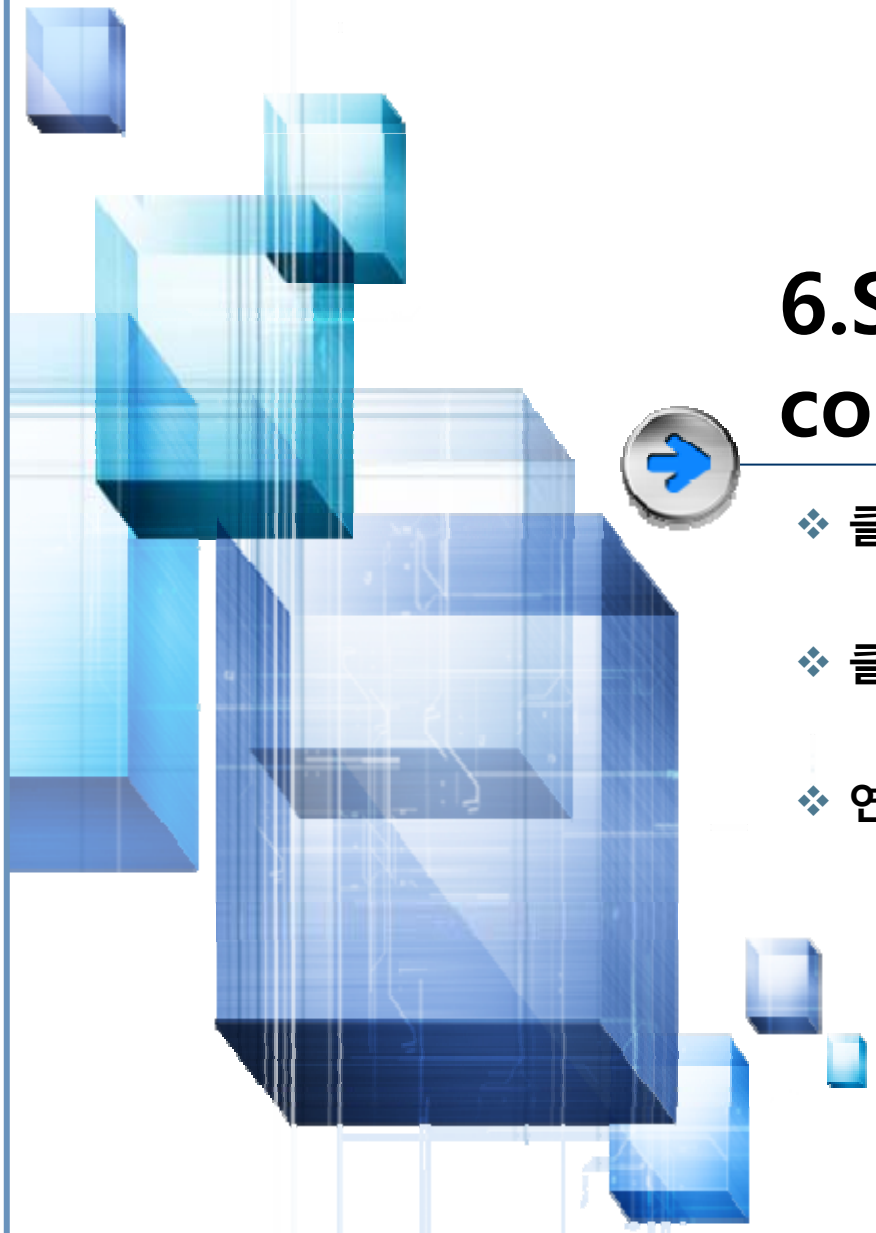


6.Static 멤버와 const 멤버

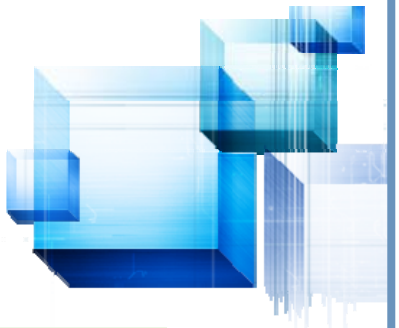
- 클래스와 const
- 클래스와 static
- 연결 리스트 프로그램 예

Jong Hyuk Park

클래스와 const

Jong Hyuk Park

C의 const (1)



```
const double PI=3.14;
PI=3.1415;                    // 컴파일 오류
```

```
const int val;
val=20;                       // 컴파일 오류
```

C의 const (1)

```
int n=10;
const int* pN=&n;
*pN=20;    // 컴파일 오류
```

```
int n1=10;
int n2=20;
int* const pN=&n1;
*pN=20;
pN=&n2;    //컴파일 오류
```

멤버변수의 상수화

```
#include<iostream>
using std::cout;
using std::endl;

class Student
{
	const int id;
	int age;
	char name[20];
	char major[30];
public:
	Student(int _id, int _age, char* _name, char*
	_major)
	{
		id=_id;
		age=_age;
		strcpy(name, _name);
		strcpy(major, _major);
	}

	void ShowData()
	{
		cout<<"이름: "<<name<<endl;
		cout<<"나이: "<<age<<endl;
		cout<<"학번: "<<id<<endl;
		cout<<"학과: "<<major<<endl;
	}
};
```

```
int main()
{
	Student Kim(200577065, 20, "Kim Gil Dong",
	                                    "Computer
	Eng.");
	Student Hong(200512065, 19, "Hong Gil Dong",
	                                    "Electronics Eng.");

	Kim.ShowData();
	cout<<endl;
	Hong.ShowData();

	return 0;
}
```

- **컴파일 에러**
  - 상수값을 생성자에서 초기화하여 발생
- **Member initializer 사용**
  - Const 멤버변수 초기화

멤버변수의 상수화 (2)

```
#include<iostream>
using std::cout;
using std::endl;

class Student
{
        const int id;
        int age;
        char name[20];
        char major[30];
public:
        Student(int _id, int _age, char* _name, char* _major) : id(_id), age(_age)
        {
                strcpy(name, _name);
                strcpy(major, _major);
        }

        void ShowData()
        {
                cout<<"이름: "<<name<<endl;
                cout<<"나이: "<<age<<endl;
                cout<<"학번: "<<id<<endl;
                cout<<"학과: "<<major<<endl;
        }
};
```

```
int main()
{
        Student Kim(200577065, 20, "Kim Gil Dong",
                                        "Computer Eng.");
        Student Hong(200512065, 19, "Hong Gil Dong",
                                        "Electronics Eng.");

        Kim.ShowData();
        cout<<endl;
        Hong.ShowData();

        return 0;
}
```

const 멤버 함수

❖ 멤버 함수의 상수화

- 함수 내부에서 멤버변수의 값을 변경할 수 없음
- 이 함수 내에서 상수화 되지 않은 함수의 호출을 허용하지 않음
- 멤버변수의 포인터의 리턴을 허용하지 않음

❖ 사용 예

```
void ShowData()  const
{
        cout<<"이름: "<<name<<endl;
        cout<<"나이: "<<age<<endl;
        cout<<"학번: "<<id<<endl;
        cout<<"학과: "<<major<<endl;
}
```

const 멤버 함수 예 (1)

```
#include <iostream>
using std::cout;
using std::endl;

class Count
{
	int cnt;
public :
	Count() : cnt(0){}
	int* GetPtr() const {
		return &cnt;	// Compile Error
	}

	void Increment(){
		cnt++;
	}

	void ShowData() const {
		ShowIntro();	// Compile Error
		cout<<cnt<<endl;
	}
	void ShowIntro() {
		cout<<"현재 count의 값 : "<<endl;
	}
};
```

```
int main()
{
	Count count;
	count.Increment();
	count.ShowData();

	return 0;
}
```

const 멤버 함수 예 (2)

```
#include <iostream>
using std::cout;
using std::endl;

class Count
{
	int cnt;
public :
	Count() : cnt(0){}
	const int* GetPtr() const {
		return &cnt;
	}

	void Increment(){
		cnt++;
	}

	void ShowData() const {
		ShowIntro();
		cout<<cnt<<endl;
	}
	void ShowIntro() const {
		cout<<"현재 count의 값 : "<<endl;
	}
};
```

```
int main()
{
	Count count;
	count.Increment();
	count.Increment();
	count.ShowData();

	return 0;
}
```

const 객체

❖ const 객체

- 데이터의 변경이 허용되지 않는 객체
- const 함수 이외에는 호출 불가

❖ const와 함수 오버로딩

- const도 함수 오버로딩 조건에 포함

```
void function(int n) const { . . . . . }
void function(int n) { . . . . . }
```

const 멤버 함수 예

```
#include <iostream>
using std::cout;
using std::endl;

class AAA
{
	int num;
public :
	AAA(int _num) : num(_num) {}
	void Add(int n){
		num+=n;
	}
	void ShowData(){
		cout<<num<<endl;
	}
};

int main()
{
	const AAA aaa(10);
	aaa.Add(10);		// Compile Error
	aaa.ShowData();	// Compile Error

	return 0;
}
```

```
#include <iostream>
using std::cout;
using std::endl;
class AAA
{
	int num;
public :
	AAA(int _num) : num(_num) {}

	void ShowData(){
		cout<<"void ShowData() 호출"<<endl;
		cout<<num<<endl;
	}
	void ShowData() const {
		cout<<"void ShowData() const 호출"<<endl;
		cout<<num<<endl;
	}
};
int main()
{
	const AAA aaa1(20);
	AAA aaa2(70);
	 aaa1.ShowData();	// void ShowData() const  호출
	aaa2.ShowData();	// void ShowData() 호출
	return 0;
}
```

클래스와 static

Jong Hyuk Park

정적멤버(static member)

❖ 정적멤버

- 같은 클래스 형으로 선언된 여러 객체들이 동일한 하나의 자료를 공유
- 클래스의 정의 시 멤버를 static이란 키워드로 정의
- 클래스 내에 선언된 정적멤버는 정적멤버가 선언된 클래스로 사용 영역이 제한된 전역변수(global variable)
- 클래스 내에서 정적멤버를 선언할 때 정적 멤버 자체가 정의되는 것은 아니기 때문에 클래스 밖에서 정적멤버를 정의하는 선언 필요
- 모든 정적멤버 변수는 특별히 초기값을 명시하지 않는 한 0으로 초기화

```
/*    PersonCount1.cpp    */
#include<iostream>
using std::cout;
using std::endl;

int count=1;

class Person
{
      char name[20];
      int age;
public:
      Person(char* _name, int _age)
      {
            strcpy(name, _name);
            age=_age;
            cout<<count++<<"번째 Person 객체생성"<<endl;
      }
      void ShowData(){
            cout<<"이름: "<<name;
            cout<<"나이: "<<age;
      }
};
```

```
int main(void)
{
      Person p1("Lee", 13);
      Person p2("Hong", 22);

      return 0;
}
```

```
1번째 Person 객체 생성
2번째 Person 객체 생성
```

```
/*    PersonCount2.cpp     */
#include<iostream>
using std::cout;
using std::endl;

class Person
{
      char name[20];
      int age;
      int count;
public:
      Person(char* _name, int _age)
      {
            count=1;
            strcpy(name, _name);
            age=_age;
            cout<<count++<<"번째 Person
      객체생성"<<endl;
      }
      void ShowData(){
            cout<<"이름: "<<name;
            cout<<"나이: "<<age;
      }
};
```

```
int main(void)
{
      Person p1("Lee", 13);
      Person p2("Hong", 22);

      return 0;
}
```

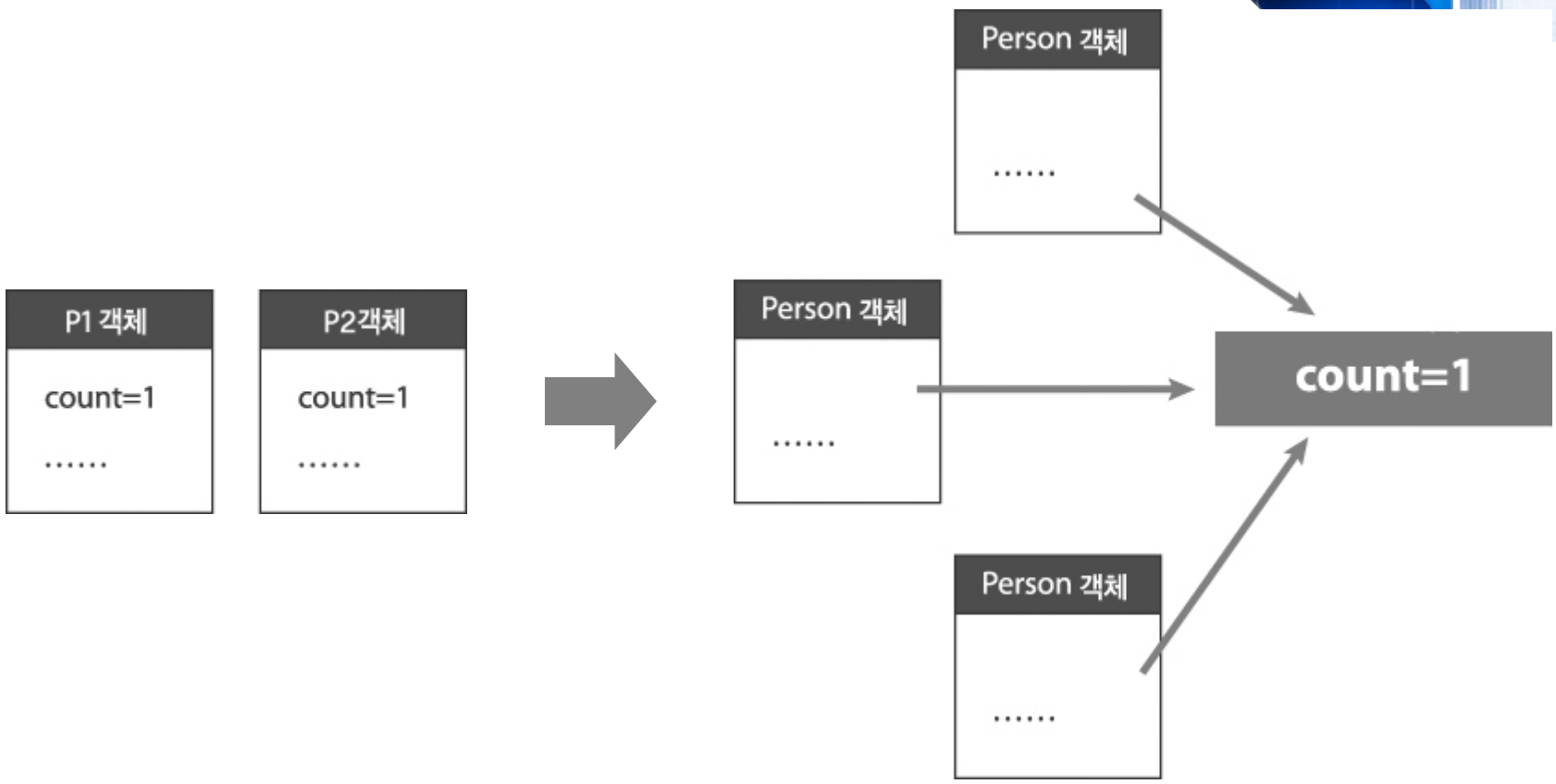
```
1번째 Person 객체 생성
1번째 Person 객체 생성
```

```
/*   PersonCount3.cpp   */
#include<iostream>
using std::cout;
using std::endl;

class Person
{
      char name[20];
      int age;
      static int count;
public:
      Person(char* _name, int _age)
      {
            strcpy(name, _name);
            age=_age;
            cout<<count++<<"번째 Person
      객체생성"<<endl;
      }
      void ShowData(){
            cout<<"이름: "<<name;
            cout<<"나이: "<<age;
      }
};

int Person::count=1; // static 멤버 초기화
```

```
int main(void)
{
      Person p1("Lee", 13);
      Person p2("Hong", 22);

      return 0;
}
```

```
1번째 Person 객체 생성
2번째 Person 객체 생성
```

정적멤버(static member)

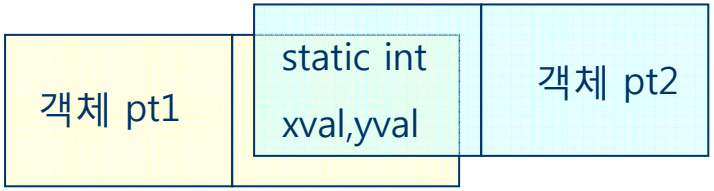


정적멤버 사용 예 (1)

```
#include <iostream>
using std::cout;

class Point {
        static int xval, yval;       // 정적멤버 선언
public:
        setpt(int x, int y)      { xval = x; yval = y; }
        void showxy()
           { cout << xval << " , " << yval << '₩n'; }
};

int Point::xval, Point::yval;     // 정적멤버 정의

int main()
{
   Point pt1, pt2;

   pt1.setpt(12,6);
   pt1.showxy();
   pt2.showxy();

   return 0;
}
```

```
12 , 6
12 , 6
```

전용부분 멤버 xval, yval을 정적멤버로 선언하고 클래스 밖에서 통용범위 지정 연산자 ::로 소속 클래스를 명시하여 정의하면 객체 pt1, pt2가 자료를 공유하여 같은 값을 출력한다.



정적멤버 사용 예 (2)

```
#include <iostream>
using std::cout;

class Test {
public:
        static int n;           // 정적멤버 선언
        void setn(int a)        { n = a; }
        void shown()            { cout << n << '₩n'; }
};

int Test::n;            // 정적멤버 정의

int main()
{
   Test x,y;

   Test::n = 88;        // 정적멤버 참조

   x.shown();
   y.shown();

   return 0;
}
```

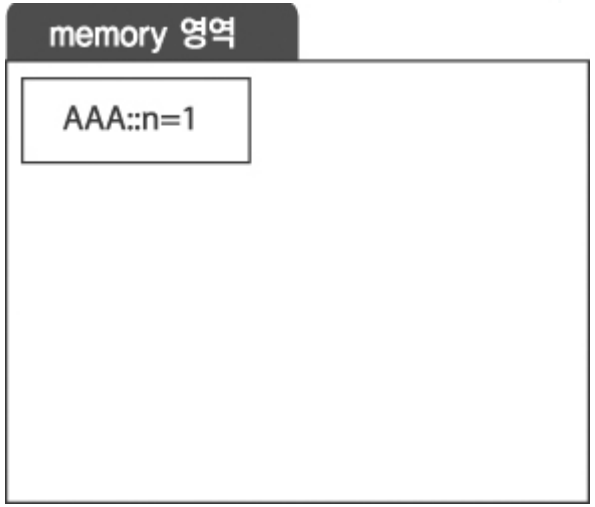
```
88
88
```

정적멤버는 클래스 밖에서 정의되므로 객체와 관계없이 직접 참조될 수 있다. 이 예에서 n을 특정 객체와 무관하게 88로 지정할 수 있다.

정적멤버(static member)

```
class AAA
{
    int val;
    static int n;
public:
    AAA(int a=0){
        val=a;
        n++;
    }
     void ShowData(){
        cout<<"val: "<<val<<endl;
        cout<<"n: "<<n<<endl;
    }
};
int AAA::n=1;
```

```
int main(void)
{
    AAA a1(10);
    a1.ShowData();

    AAA a2(20);
    a2.ShowData();
    return 0;
}
```



정적멤버(static member)

❖ static 멤버의 특징

- 클래스 변수, 클래스 함수라 한다.
- main 함수 호출 이전에 메모리 공간에 올라가서 초기화(전역변수와 동일)
- 선언된 클래스의 객체 내에 직접 접근 허용
- static 멤버 초기화문으로 초기화해야 함

explicit & mutable

❖ explicit

- 명시적 호출만 허용한다.

❖ mutable

- const에 예외를 둔다

explicit & mutable

```
/*   explicit.cpp   */

#include<iostream>
using std::cout;
using std::endl;

class AAA
{
public:
    explicit AAA(int n){
                cout<<"explicit AAA(int n)"<<endl;
     }
};

int main(void)
{
     AAA a1=10;

     return 0;
}
```

explicit & mutable

```
/*    mutable.cpp    */
#include<iostream>
using std::cout;
using std::endl;

class AAA
{
private:
      mutable int val1;
      int val2;
public:
      void SetData(int a, int b) const
      {
            val1=a;  // val1이 mutable이므로 OK!
            val2=b;  // Error!
      }
};
```

```
int main(void)
{
      AAA a1;
      a1.SetData(10, 20);
      return 0;
}
```

과제

1. 5장의 은행계좌 관리 과제 프로그램에 다음을 추가하여라
   - ❖ 6-4절과 같이 멤버함수 상수화

Q & A

Jong Hyuk Park