6장. static 멤버와 const 멤버

부산대학교
김미경
ddosun@pusan.ac.kr

- **const 키워드 복습 1**

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

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

- **const 키워드 복습 2**

```
int n1=10, n2=100;
const int* pN1=&n1;
*pN1=20;   // 컴파일 오류
pN1 = &n2; // o
```

```
int* const pN2=&n1;
*pN2=20;  // o
pN2=&n2;   //컴파일 오류
```

6-1 클래스와 const

- 멤버 변수의 상수화, 그리고 초기화
  - 멤버 이니셜라이저(member initializer)
  - ComstMember1.cpp

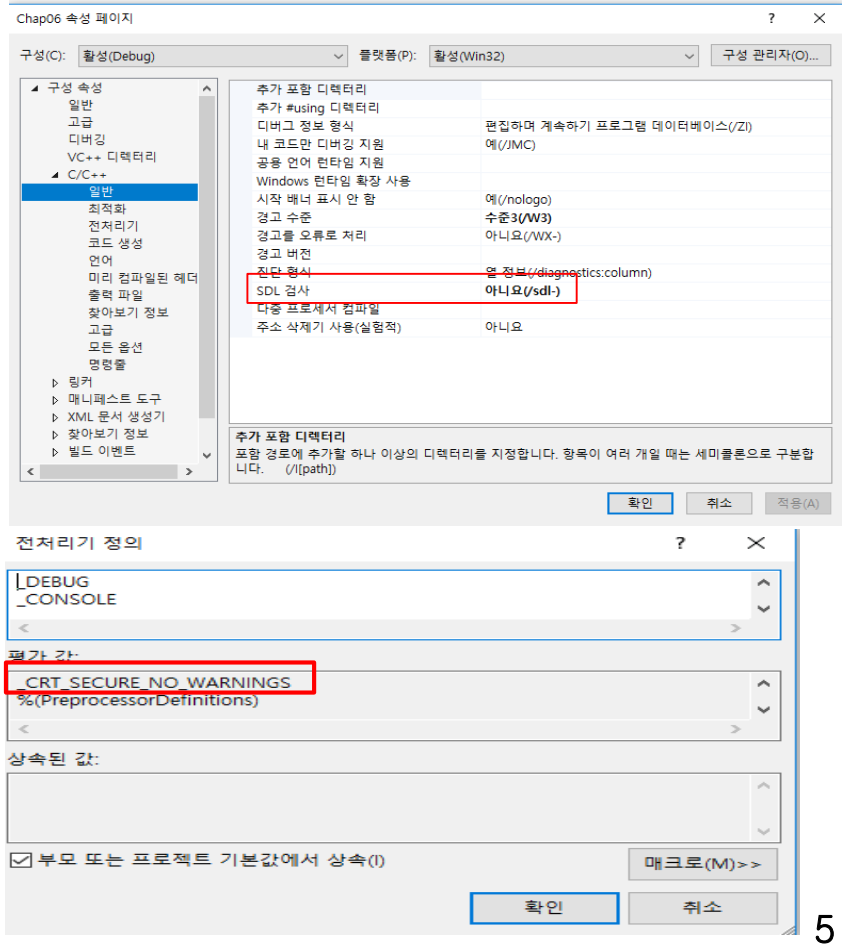
```
Student(…) : id(_id)
```

- const 멤버 함수
  - 멤버 변수의 값 변경 허용 않는다.
  - 멤버 변수 값의 변경에 대한 기회제공도 불가
  - ConstMember2.cpp, ConstMember3.cpp

Visual Studio에서 작업 시

- C언어 C4996 _CRT_SECURE_NO_WARNINGS
- ➔프로젝트명에서 오른쪽 버튼 속성
- ➔warning으로

- 또는
- 속성 ➔C/C++의 전처리기
- ➔전처리기 정의 편집



6-1 클래스와 const

멤버 변수가 const 이므로 생성자로 초기화 불가능

```
#include <iostream>
#include <cstring>
using namespace std;
class Student
{
    const int id;
    //int id;
    int age;
    char name[20];
    char major[30];
public:
    Student(const int _id, const int _age, const char* _name, const 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(201911111, 20, "Kim Gil Dong", "Computer Eng.");
    Student Lee(202011111, 19, "Lee Gil Dong", "Electronics Eng.");

    Kim.ShowData();
    Lee.ShowData();
}
```

6-1 클래스와 const

```
/* 2_멤버이니셜라이즈 */
#include <iostream>
#include <cstring>
using namespace std;
class Student
{
    const int id;
    int age;
    char name[20];
    char major[30];
public:
    Student(const int _id, const int _age, const char* _name, const char* majo
    :id(_id),age(_age)
    {
        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(201911111, 20, "Kim Gil Dong", "Computer Eng.");
    Student Lee(202011111, 19, "Lee Gil Dong", "Electronics Eng.");

    Kim.ShowData();
    Lee.ShowData();
}
```

```
/*   3_ConstMember3.cpp*/
#include <iostream>
using std::cout;using std::endl;
class Count{
    int cnt;
public :
    Count() : cnt(5){}
    int* GetPtr() const{
        return &cnt;  // Compile Error
    }
    void Increment(){
        cnt++;
    }
    void ShowData() const {
        ShowIntro();  // Compile Error
        cout<<cnt<<endl;
    }
    void ShowIntro() {
        cout<<"현 재  count의  값  :  "<<endl;
    }
};
int main()
{
    Count count;
    count.Increment();
    count.ShowData();
}
```

주소 return 하면 수정가능 o

Const 함수는 const함수만 call 가능

6-1 클래스와 const

- const 객체
  - 데이터의 변경이 허용되지 않는 객체
  - const 함수 이외에는 호출 불가
  - ConstObject.cpp

- const와 함수 오버로딩
  - const도 함수 오버로딩 조건에 포함
  - constOverloading.cpp

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

6-1 클래스와 const

```
1  /*   ConstObject.cpp*/
2  #include <iostream>
3  using std::cout;
4  using std::endl;
5
6  class AAA
7  {
8      int num;
9  public :
10     AAA(int _num) : num(_num) {}
11     void Add(int n){
12         num+=n;
13     }
14     void ShowData(){
15         cout<<num<<endl;
16     }
17 };
18
19 int main()
20 {
21     const AAA aaa(10);
22     aaa.Add(10);  // Compile Error
23     aaa.ShowData();  // Compile Error
24
25     return 0;
26 }
```

- showData const 만들기
- main수정

```
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(10);
//  aaa1.Add(10);  // Compile Error
//  aaa1.ShowData();  // Compile Error
    aaa1.ShowData();
    AAA aaa2(70);

    aaa2.ShowData();
    return 0;
}
```

Const 함수 없을 때는 error

Const 객체는 const함수 호출

6-2 클래스와 static

- static 멤버의 등장
  - 전역 변수와 전역 함수를 일부 대체하기 위해서 등장
  - PersonCount1.cpp

- static 키워드의 효과
  - 모든 객체가 공유할 수 있는 멤버
  - PersonCount2.cpp, PersonCount3.cpp

6-2 클래스와 static

```
1  /*   PersonCount1.cpp*/
2  #include<iostream>
3  #include <cstring>
4  using std::cout;using std::endl;
5  int count=1;
6  class Person
7  {
8       char name[20];
9       int age;
10 public:
11      Person(const char* _name, const int _age)
12      {
13           strcpy(name, _name);
14           age=_age;
15           cout<<count++<<"번 째  Person 객 체  생 성 "<<endl;
16      }
17      void ShowData(){
18           cout<<"이 름 : "<<name;
19           cout<<"나 이 : "<<age;
20      }
21 };
22 int main(void)
23 {
24      Person p1("Lee", 13);
25      Person p2("Hong", 22);
26 }
```

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

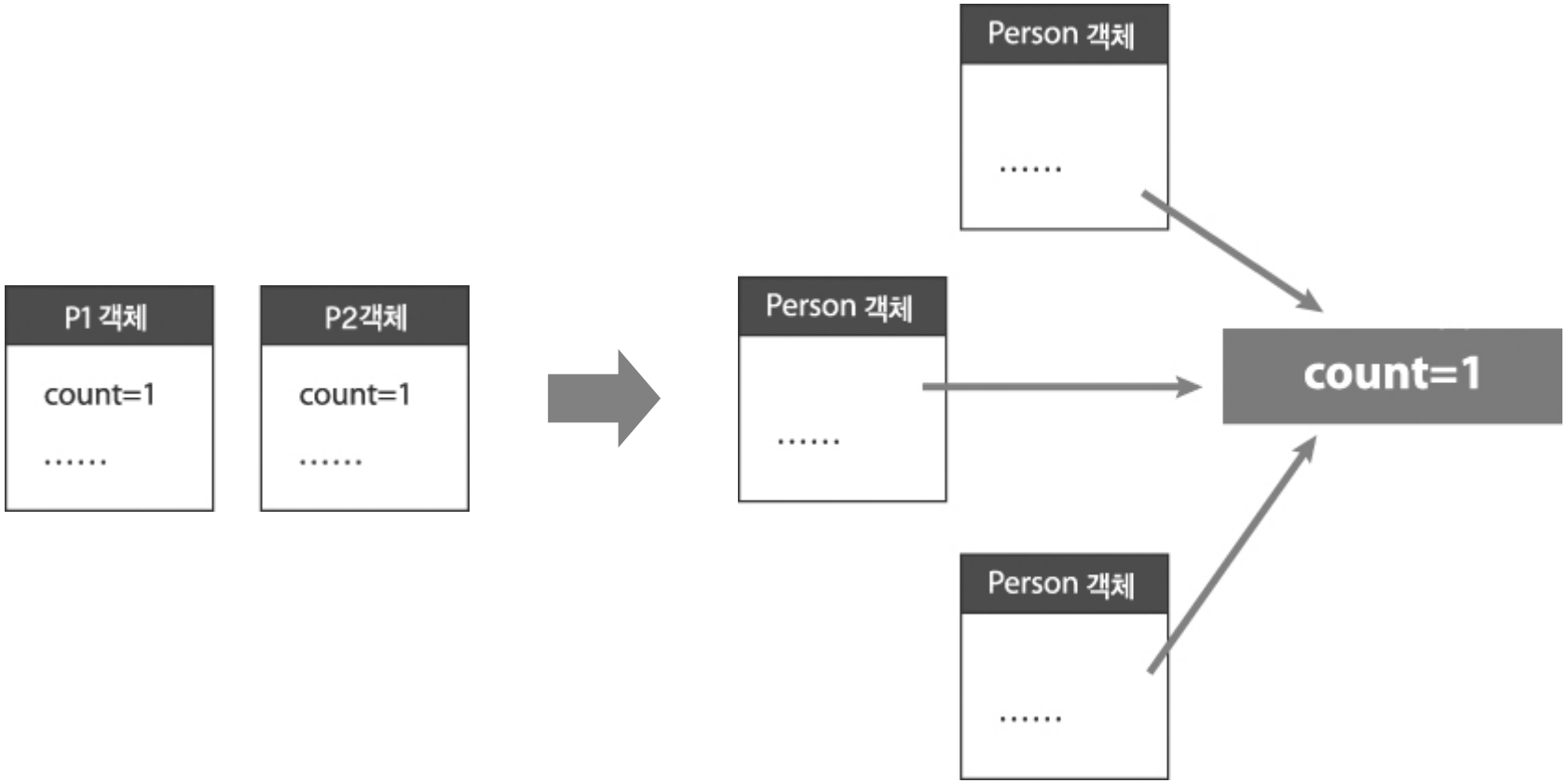
6-2 클래스와 static

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

```cpp
/*   PersonCount2.cpp*/
#include<iostream>
#include <cstring>
using std::cout;using std::endl;
class Person
{
    char name[20];
    int age;
    int count;
public:
    Person(const char* _name, const int _age)
    {
        count =1;
        strcpy(name, _name);
        age=_age;
        cout<<count++<<"번 째  Person 객 체  생 성 "<<endl;
    }
    void ShowData(){
        cout<<"이 름 : "<<name;
        cout<<"나 이 : "<<age;
    }
};
int main(void)
{
    Person p1("Lee", 13);
    Person p2("Hong", 22);
}
```

```cpp
/*   PersonCount3.cpp*/
#include<iostream>
#include <cstring>
using std::cout;using std::endl;
class Person
{
    char name[20];
    int age;
    static int count;
public:
    Person(const char* _name, const int _age)
    {
        strcpy(name, _name);
        age=_age;
        cout<<count++<<"번 째  Person 객 체  생 성 "<<endl;
    }
    void ShowData(){
        cout<<"이 름 : "<<name;
        cout<<"나 이 : "<<age;
    }
};
int Person::count =1;
int main(void)
{
    Person p1("Lee", 13);
    Person p2("Hong", 22);
}
```

6-2 클래스와 static



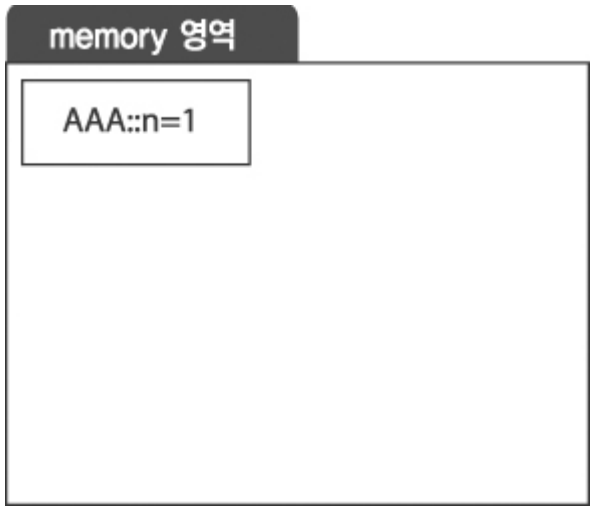
6-2 클래스와 static

- 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
  - 외울 생각 말자! 이해하자!

6-2 클래스와 static

```
1  /* static.cpp */
2  #include <iostream>
3  using std::cout;using std::endl;
4
5  class AAA
6  {
7      int val;
8      static int n;
9  public :
0      AAA(int a=0){
1          val=a;
2          n++;
3      }
4      void ShowData(){
5
6          cout << "val = " << val <<endl;
7          cout << "n = " << n <<endl;
8      }
9  };
0  int AAA::n=1;
```

```
21 int main()
22 {
23     AAA a1(10);
24     a1.ShowData();
25
26     AAA a2(20);
27     a2.ShowData();
28 }
```



6-3 explicit & mutable

- explicit
  - 명시적 호출만 허용한다.
  - explicit.cpp

- mutable- 사용 안 하는 것이 좋다.
  - const에 예외를 둔다
  - mutable.cpp

6-3 explicit & mutable

```cpp
////   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; //error

    return 0;
}
```

```cpp
////   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;
}
```

- 3단계
  - 복사 생성자의 정의
  - Account 클래스에 깊은 복사를 하는 복사 생성자를 삽입

- 4단계
  - const 키워드의 삽입
  - const 키워드를 많이 사용할수록 프로그램은 안정적으로 구성이 된다.