Member Initializer List

❖ 데이터 멤버 초기화 위치

```
class Rectangle {
  int leftTopX, leftTopY ;
  int rightBottomX, rightBottomY ;
public:
  // 모든 파라미터가 기본값을 가지는 default constructor
  Rectangle(int x1=0, int y1=0, int x2=0, int y2=0)
    : leftTopX(x1), leftTopY{y1} // 멤버 초기화 목록
  { // 함수 본문
    rightBottomX = x2 ; rightBottomY = y2 ;
  }
```

Good Design : 생성자에서는 할당(assignment)보다는 멤버 초기화 리스트(initialization) 를 더 선호하라 (C.49)

- initialization explicitly states that initialization is done (rather than assignment).
- more elegant and efficient
- prevents "use before set" errors

```
class A {   // Good
    string s1;
public:
    A(czstring p) : s1{p} { }    // GOOD: directly construct (and the C-string is explicitly named)
    // ...
};
```

```
class B {   // BAD
    string s1;
public:
    B(const char* p) { s1 = p; }   // BAD: default constructor followed by assignment
    // ...
};
```

Good Design :
멤버 변수는 선언된 순서대로 초기화하라! (C.47)

- 혼란과 에러를 최소화 함
- 멤버 변수의 선언된 순서가 초기화가 발생하는 순서이다 (m1, m2 순서로 초기화, member initializer list의 순서와 무관함)

```
class Foo {
    int m1;
    int m2;
public:
    Foo(int x) :m2{x}, m1{++x} { }   // BAD: misleading initializer order
    // ...
};

Foo x(1); // surprise: x.m1 == x.m2 == 2
```

Member Initializer의 용도:

객체 멤버 변수

```
class Point {
   int x, y ;
public:
  // 모든 파라미터가 기본값을 가지는 default constructor
   Point(int x=0, int y=0) : x{x}, y{y} { }
  // 복사 생성자가 없으므로 컴파일러가 자동으로 생성함
} ;
class Rectangle {
   // rightBottom, leftTop의 순으로 호출됨
   Point rightBottom, leftTop  ;
public:
   // 1) 번 생성자 (Point의 복사 생성자 이용)
   Rectangle(const Point& p1, const Point& p2=Point(0,0))
      : leftTop(p1), rightBottom(p2) {} //member initializer

   // 2) 번 생성자
   Rectangle(int x1, int y1, int x2=0, int y2=0)
      // member initializer list
      : leftTop{x1, y1}, rightBottom{x2, y2} {}
   // 기본 생성자
   Rectangle()
      // 생략가능함; default constructor를 호출
      // : leftTop(), rightBottom() {}
      {}
} ;
```

```
int main() {
   // 기본 생성자
   Rectangle r1 ;

   Point p ;
   // 1) 번 생성자 호출
   Rectangle r2(p, p) ;
   Rectangle r3(p) ;

   // 2) 번 생성자 호출
   Rectangle* const pR4 =
      new Rectangle(100, 200) ;
   Rectangle* const pR5 =
      new Rectangle(100) ;

   delete pR4 ;
   delete pR5 ;
}
```

Member Initializer의 용도:
const 멤버 변수

```
#include <string>
#include <vector>
using namespace std;

enum Grade { FRESH=1, SOPHOMORE, JUNIOR, SENIOR } ;

class Student ;

class School {
   // const 멤버는 반드시 member initializer로 초기화해야 함
   const string name ;
   // 멤버 객체는 반드시 member initializer로 초기화해야 함
   vector<Student*> students ;
   float budget ;
public:
   School(const string& _name, int size) :
     name{_name}, students{size} { budget = 0 ; }
} ;
```

Member Initializer의 용도 :
참조 멤버 변수

```
class Student {
    string name ;
    Grade grade ;
    // reference 멤버는 반드시 member initializer로 초기화 해야함
    const School& school ;
public:
    Student(const School& _school, const string& _name="")
        : school(_school), name(_name) { grade = FRESH ; }
} ;
```

Good Design: 데이터 멤버를 const 또는 참조로 만들지 말자! (C.12)

- They are not useful, and make types difficult to use by making them either uncopyable or partially uncopyable for subtle reasons.

```
class bad {
    const int i;    // bad
    string& s;      // bad
    // ...
};
```