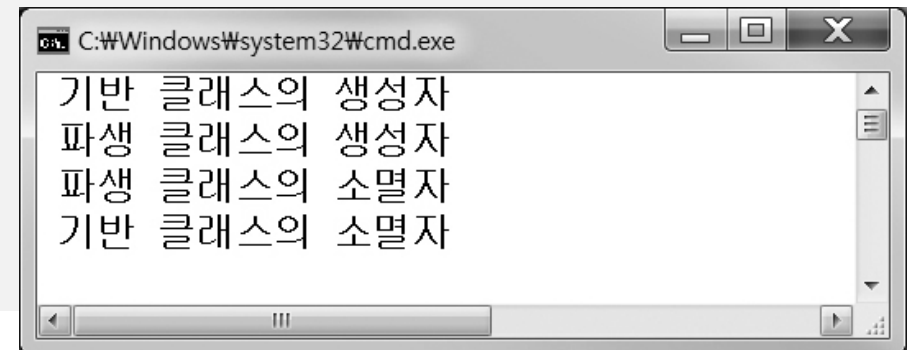


- 상속 관계에 있는 클래스에서 생성자 특징
 - ① 생성자는 멤버함수지만 상속할 수 없음
 - ② 파생 객체가 생성되어도 기반 클래스의 생성자까지 연속적으로 자동 호출
- 생성자와 소멸자 호출 순서
 - ① 파생 클래스의 객체가 생성될 때 기반 클래스의 멤버변수에 대한 메모리 영역에 파생 클래스의 멤버변수에 대한 메모리 영역을 포함한 형태로 메모리를 할당
 - ② 상속 관계에 있는 파생 클래스로 객체를 생성하면 자신의 생성자 뿐만 아니라 기반 클래스의 생성자까지 연속으로 자동 호출되는데, 기반 클래스의 생성자가 먼저 호출되고 파생 클래스의 생성자가 나중에 호출

상속 관계에서의 생성자와 소멸자 알아보기

```
01 #include<iostream>
02 using namespace std;
03 class Base{
04     public:
05         Base();
06         ~Base();
07 };
08 Base::Base()
09 {
10     cout<<" 기반 클래스의 생성자 "<<endl;
11 }
12 Base::~~Base()
13 {
14     cout<<" 기반 클래스의 소멸자 "<<endl;
15 }
16 class Derived : public Base{
17     public :
18         Derived();
19         ~Derived();
20 };
```

```
21 Derived::Derived()
22 {
23     cout<<" 파생 클래스의 생성자 "<<endl;
24 }
25 Derived::~~Derived()
26 {
27     cout<<" 파생 클래스의 소멸자 "<<endl;
28 }
29
30 void main()
31 {
32     Derived obj;
33 }
```

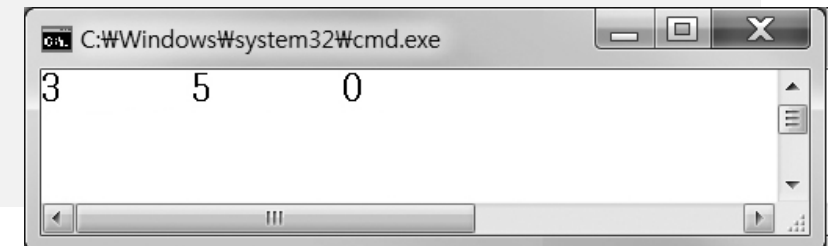


```
C:\Windows\system32\cmd.exe
기반 클래스의 생성자
파생 클래스의 생성자
파생 클래스의 소멸자
기반 클래스의 소멸자
```

상속 관계에서의 생성자와 소멸자 알아보기

```
01 #include<iostream>
02 using namespace std;
03 class Calc{
04     protected:
05         int a;
06         int b;
07         int c;
08     public:
09         void Prn();
10         Calc(int new_A,int new_B);
11         // 기반 클래스에 생성자 추가
12 };
13 void Calc::Prn()
14 {
15     cout<<a<<"\t"<<b<<"\t"<<c<<endl;
16 }
17
```

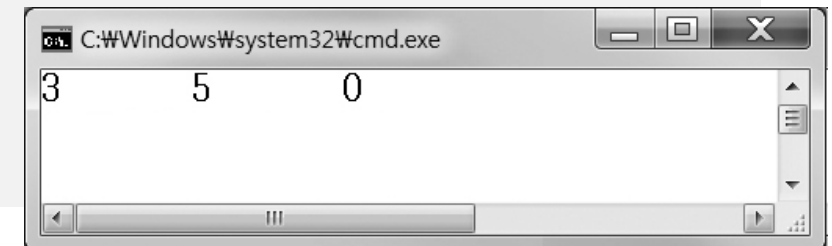
```
18 Calc::Calc(int new_A,int new_B)
19 // 기반 클래스에 생성자 추가
20 {
21     a=new_A;
22     b=new_B;
23     c=0;
24 }
25 void main()
26 {
27     Calc x(3, 5);
28     x.Prn();
29 }
```



Calc 클래스에서 생성자 정의하기

```
01 #include<iostream>
02 using namespace std;
03 class Calc{
04     protected:
05         int a;
06         int b;
07         int c;
08     public:
09         void Prn();
10         Calc(int new_A,int new_B);
           // 기반 클래스에 생성자 추가
11 };
12
13 void Calc::Prn()
14 {
15     cout<<a<<"\t"<<b<<"\t"<<c<<endl;
16 }
17
```

```
18 Calc::Calc(int new_A,int new_B)
   // 기반 클래스에 생성자 추가
19 {
20     a=new_A;
21     b=new_B;
22     c=0;
23 }
24
25 void main()
26 {
27     Calc x(3, 5);
28     x.Prn();
29 }
```



생성자를 정의한 Calc 클래스의 파생 클래스 설계

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;    int b;    int c;
public:
    void Prn();
    Calc(int new_A,int new_B); // 기반 클래스에 생성자 추가
};

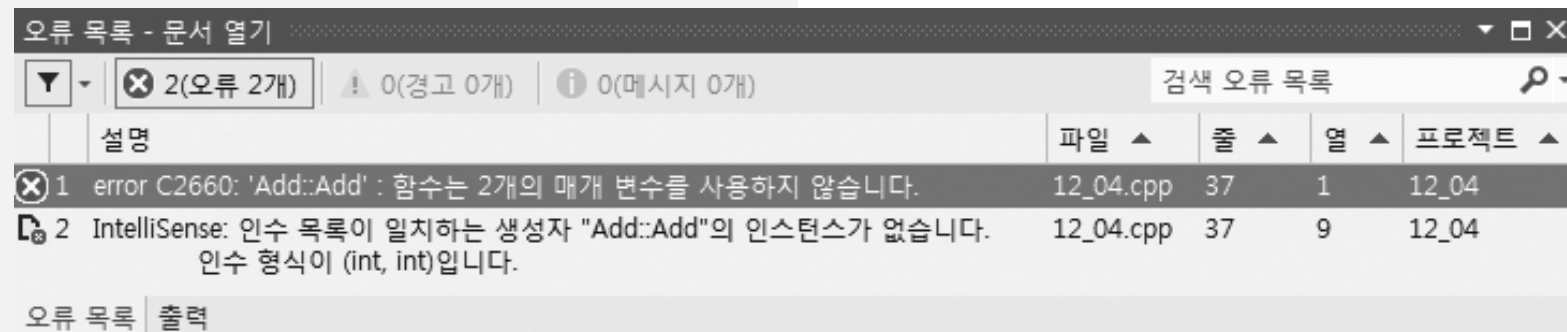
void Calc::Prn()
{
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}

Calc::Calc(int new_A,int new_B) // 기반 클래스에 생성자 추가
{
    a=new_A;
    b=new_B;
    c=0;
}
```

```
class Add : public Calc{
public:
    void Sum();
};

void Add::Sum()
{
    c=a+b;
}

void main()
{
    Add y(3, 5);
    y.Prn();
}
```



생성자를 정의한 Calc 클래스의 파생 클래스 설계

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;    int b;    int c;
public:
    void Prn();
    Calc(int new_A,int new_B); // 기반 클래스에 생성자 추가
};
```

```
void Calc::Prn()
{
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}
```

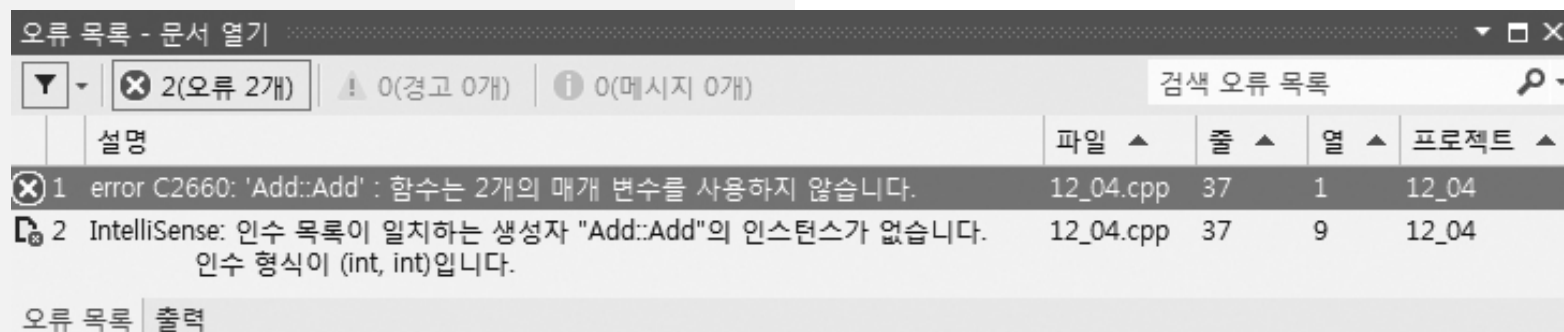
```
Calc::Calc(int new_A,int new_B)
{
    a=new_A;
    b=new_B;
    c=0;
}
```

```
class Add : public Calc{
public:
    void Sum();
};

void Add::Sum()
{
    c=a+b;
}

void main()
{
    Add y(3, 5);
    y.Prn();
}
```

Add y(3,5)에서 Add 객체를 생성할 때 매개변수 2개를 주었더니 매개변수가 2개인 생성자가 없다는 에러 메시지 출력.
매개변수 2개인 생성자가 기반(Calc) 클래스에 분명히 정의되어 있는 데도 불구하고 이를 Add클래스에서 사용하지 못하고 Add라는 이름의 생성자를 호출하고 있음. 생성자가 상속되지 않기 때문에 발생하는 에러이므로, Add 클래스에 매개변수가 2개인 생성자 정의 필요.



Add 클래스에 생성자 정의하기

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;    int b;    int c;
public:
    void Prn();
    Calc(int new_A,int new_B); // 기반 클래스에 생성자 추가
};

void Calc::Prn()
{
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}

Calc::Calc(int new_A,int new_B) // 기반 클래스에 생성자 추가
{
    a=new_A;
    b=new_B;
    c=0;
}
```

```
class Add : public Calc{
public :
    void Sum();
    Add(int new_A, int new_B);
    // 파생클래스에 생성자 추가
};

void Add::Sum()
{
    c=a+b;
}

Add::Add(int new_A,int new_B)
// 파생클래스에 생성자 추가
{
    a=new_A;
    b=new_B;
    c=0;
}

void main()
{
    Add y(3, 5);
    y.Prn();
}
```

Add 클래스에 생성자 정의하기

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;    int b;    int c;
public:
    void Prn();
    Calc(int new_A,int new_B); // 기반 클래스에 생성자 추가
};

void Calc::Prn()
{
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}

Calc::Calc(int new_A,int new_B) // 기반 클래스에 생성자 추가
{
```

```
class Add : public Calc{
public :
    void Sum();
    Add(int new_A, int new_B);
    // 파생클래스에 생성자 추가
};

void Add::Sum()
{
    c=a+b;
}

Add::Add(int new_A,int new_B)
// 파생클래스에 생성자 추가
{
    a=new_A;
    b=new_B;
    c=0;
}

void main()
{
    Add y(3, 5);
    y.Prn();
}
```

오류 목록 - 문서 열기

2(오류 2개) | 0(경고 0개) | 0(메시지 0개) | 검색 오류 목록

	설명	파일	줄	열	프로젝트
1	error C2512: 'Calc' : 사용할 수 있는 적절한 기본 생성자가 없습니다.	12_05.cpp	37	1	12_05
2	IntelliSense: "Calc" 클래스의 기본 생성자가 없습니다.	12_05.cpp	37	1	12_05

오류 목록 | 출력

Add 클래스에 생성자 정의하기

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;    int b;    int c;
public:
    void Prn();
    Calc(int new_A,int new_B); // 기반 클래스에 생성자 추가
};
```

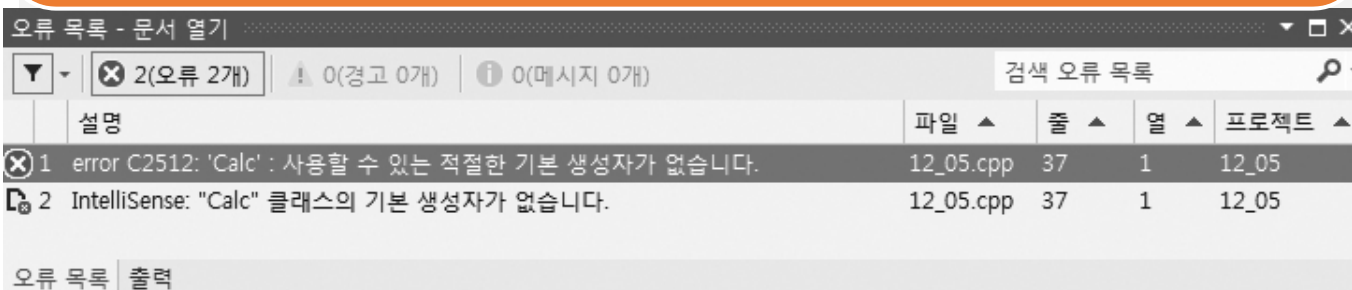
Add :Addy(int new_A, int new_B)와 같이 Add 클래스에 매개변수가 2개인 생성자를 추가했음에도 여전히 Add 클래스로 객체를 생성하는 과정에서 에러 발생.
이번에는 Add y(3,5)의 객체 생성에서 에러가 발생하는 것이 아닌 파생클래스의 생성자 추가에서 발생.
발생한 에러 메시지는 Add 클래스의 생성자를 호출하는 과정에서 기반(Calc) 클래스의 매개변수를 갖지 않는 기본 생성자를 호출하지만 기본 생성자가 없다는 에러 메시지 출력.

```
class Add : public Calc{
public :
    void Sum();
    Add(int new_A, int new_B);
    // 파생클래스에 생성자 추가
};

void Add::Sum()
{
    c=a+b;
}

Add::Add(int new_A,int new_B)
// 파생클래스에 생성자 추가
{
    a=new_A;
    b=new_B;
    c=0;
}

void main()
{
    Add y(3, 5);
    y.Prn();
}
```



● 상속 관계에서 생성자 문제

- ❶은 파생 클래스의 생성자가 기반 클래스의 암시적으로 컴파일러에 의해 제공되는 기본 생성자를 호출
- ❷처럼 매개변수를 한 개라도 갖는 생성자를 정의하면 더 이상 C++ 컴파일러가 제공하지 않음

❶

```
Calc::Calc()
```

```
{  
}  
}
```

```
Add::Add(int new_A, int new_B)  
{  
}  
}
```

```
Add y;
```

❷

```
Calc::Calc()
```

```
{  
}  
}
```

```
Calc::Calc(int new_A, int new_B)  
{  
}
```

```
Add::Add(int new_A, int new_B)  
{  
}
```

```
Add y;
```

● 파생 클래스에서 기반 클래스의 생성자를 명시적으로 호출하기

- 기반 클래스의 생성자를 파생 클래스에서 명시적으로 호출하는 것은 콜론(:) 초기화의 형태
- 파생 클래스의 생성자를 정의할 때 함수의 머리 부분 맨 마지막에 :을 기술했 후 기반 클래스의 생성자를 호출
- 명시적으로 기술했지 않아도 Add 클래스의 생성자는 자신의 기반 클래스인 Calc 클래스의 생성자를 자동 호출

```
Add::Add() : Calc()
```

```
{  
....  
}
```

기반 클래스의 생성자가
명시적으로 호출

```
Add::Add()
```

```
{  
....  
}
```

명시적으로 호출하지 않아도
기반 클래스의 생성자가 암시적으로 호출

- 기반 클래스 내에 매개변수를 갖는 생성자를 추가로 정의할 경우
 - 매개변수가 없는 기본 생성자를 프로그래머가 반드시 명시적으로 정의하는 습관을 들여야 함

```
class Calc{  
public :  
    Calc(int new_A, int new_B)  
    {  
        a=new_A;  
        b=new_B;  
        c=0;  
    }  
    Calc::Calc()  
    {  
        a=0;  
        b=0;  
        c=0;  
    }  
}
```

기본 생성자를 프로그래머가
명시적으로 정의하는 습관을 들인다.

- 기반 클래스의 생성자에 매개변수 전달하기

- 파생 클래스(Add)에도 동일한 작업이 중복되어 기술

```
Calc::Calc(int new_A, int new_B)
{
    a=new_A;
    b=new_B;
    c=0;
}
```

```
Add::Add(int new_A, int new_B)
{
    a=new_A;
    b=new_B;
    c=0;
}
```

- 파생 클래스의 생성자가 받은 매개변수의 값을 기반 클래스의 생성자를 호출하면서 전달

기반 클래스(Calc)의 생성자에 중복되는 내용은 생략.

파생 클래스 생성자의 형식 매개변수 값이
기반 클래스의 생성자를 호출할 때 실 매개변수로 넘겨줌

```
Add:: Add (int new_A, int new_B) : Calc(new_A, new_B)
{
    // a=new_A;
    // b=new_B;
    // c=0;
}
```

기반 클래스의 생성자를
명시적으로 호출

파생 클래스 생성자의 정의

객체 생성할 때 준 값을 저장하는 형식 매개변수이다.

● 그 외 주의 사항

- : 다음에 기술된 기반 클래스의 생성자는 함수의 호출임. 함수를 호출할 때 소괄호 내부에 기술된 내용은 실 매개변수이므로 변수나 값을 적어주어야 함. 함수의 머리 부분에 기술되었다고 해서 자료형을 기술하는 경우가 많은데 잘못된 것

```
Add:: Add (int new_A, int new_B) : Calc(int new_A, int new_B)
{
    잘못된 표현
}
```

- 함수를 정의할 때는 소괄호 안에 형식 매개변수를 기술해야 하므로 자료형과 함께 변수 선언문 형태를 취하고 : 다음에 나오는 부분은 분명히 함수 호출이므로 변수만 기술. 자료형을 기술해서는 안됨

상속 관계에서 생성자 문제 해결

```
#include<iostream>
using namespace std;
class Calc{
protected:
    int a;   int b;   int c;
public:
    void Prn();
    Calc(int new_A,int new_B);
    Calc();
};

void Calc::Prn()
{
    cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}

Calc::Calc(int new_A,int new_B)
{
    a=new_A;
    b=new_B;
    c=0;
}

Calc::Calc()
{
    a=0;   b=0;   c=0;
}
```

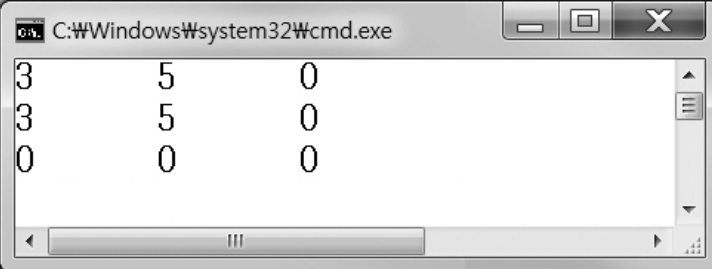
```
class Add : public Calc{
public :
    void Sum();
    Add(int new_A, int new_B);
    Add()
};

void Add::Sum()
{
    c=a+b;
}

Add::Add(int new_A,int new_B) : Calc(new_A, new_B)
{
    // a=new_A;
    // b=new_B;
    // c=0;
}

Add::Add() : Calc()
{
}

void main()
{
    Calc x(3, 5);
    x.Prn();
    Add y(3, 5);
    y.Prn();
    Add z;
    z.Prn();
}
```



```
C:\Windows\system32\cmd.exe
3      5      0
3      5      0
0      0      0
```