7 클래스의 응용

클래스를 자유자재로 사용하자.

원리를 알면 IT가 맛있다
IT COOKBOOK for Beginner
한빛 교재 시리즈

이 장에서 다룰 내용

1 객체의 치환

2 함수와 클래스의 상관관계

01_객체의 치환

❖ **객체도 변수와 마찬가지로 치환이 가능하다.**

기본예제[7-1]
객체도 일반 변수와 마찬가지로 대입이 가능하다.

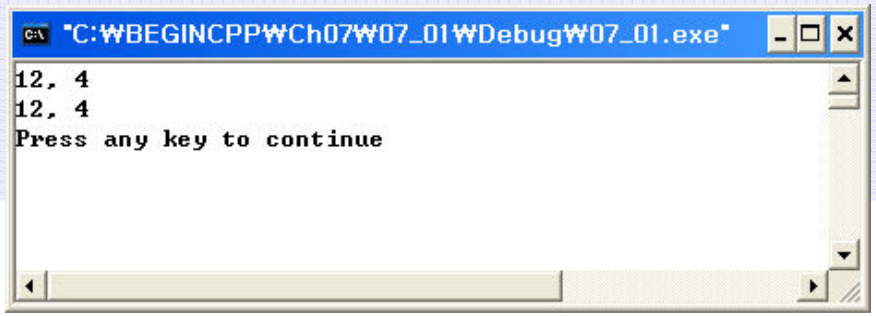
기본예제[7-2]
객체의 치환 시에는 조심해야 할 점이 있다.
복사생성자의 필요성에 대하여 알아보자.

[기본예제 7-1] 클래스의 치환

```
01  #include<iostream>
02  using namespace std;
03  class Unit {
04          int x,y;
05  public:
06          void set(int i, int j) { x=i; y=j; }
07          void show() { cout << x << ", " << y << "₩n"; }
08  };
09
10  int main(int argc, char* argv[])
11  {
12          Unit A, B;
13          A.set(12,4);
14          B = A;
15          A.show();
16          B.show();
17          return 0;
18  }
```

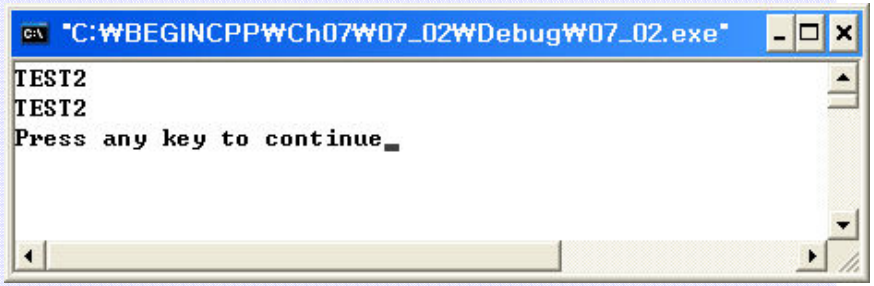
Unit 클래스를 정의한다.

A를 B에 치환한다.



[기본예제 7-2] 클래스의 치환 #2

```
01 #include<iostream>
02 #include<cstring>
03 using namespace std;
04 class Unit {
05    char* str;
06 public:
07    Unit()
08    {
09       str = NULL;
10    }
11
12    Unit(char* agStr)
13    {
14       int strLength = strlen(agStr);
15       str = new char[strLength + 1];
16       strcpy(str, agStr);
17    }
18
19    void set(char* agStr)
20    {
21       strcpy(str, agStr);
22    }
23
24    void show()
25    {
26       cout << str << "₩n";
27    }
28 };
29
30 int main(int argc, char* argv[])
31 {
32    Unit A("TEST1");
33    Unit B = A;
34
35    A.set("TEST2");
36
37    A.show();
38    B.show();
39    return 0;
```



Test1의 값을 갖는 A를 생성한다.
B에 A를 복사한다.

A의 내용을 Test1에서 B로 바꾼다.

01_객체의 치환

❖ 복사생성자

- 기본예제[7-2]에서 일어난 이상한 현상의 이유를 알려면, 복사생성자에 대해서 알아야 한다.
- B를 생성하는 데에는 기본 생성자가 아닌 복사 생성자가 사용된 것이다.
- 기본 생성자들이 호출되는 상황을 파악하기 위해서 기본예제[7-2]를 약간 수정한 기본예제[7-3]을 실행해 보자.
- 기본예제[7-4]에서 복사생성자를 추가하여 실행해 보자.

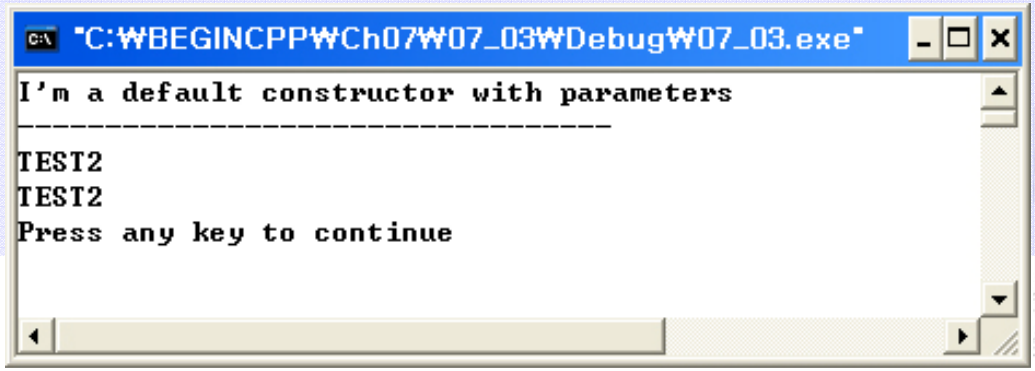
[기본예제 7-3] 클래스의 치환 #3

```
...
07    Unit()
08    {
09       cout << "I'm a default constructor₩n";
10       str = NULL;
11    }
12
13    Unit(char* agStr)
14    {
15       cout << "I'm a default constructor with parameters₩n";
16       int strLength = strlen(agStr);
17       str = new char[strLength + 1];
18       strcpy(str, agStr);
19    }
...
32  int main(int argc, char* argv[])
33  {
34    Unit A("TEST1");
35    Unit B = A;
36
37    cout << "--------------------------------₩n";
38    A.set("TEST2");
39
40    A.show();
41    B.show();
42
43    return 0;
44  }
```

기본 생성자가 실행된 것을 화면에 표시한다.

입력 인자를 받은 경우에 대한 생성자가 실행된 것을 화면에 표시한다.

Separator 표시.

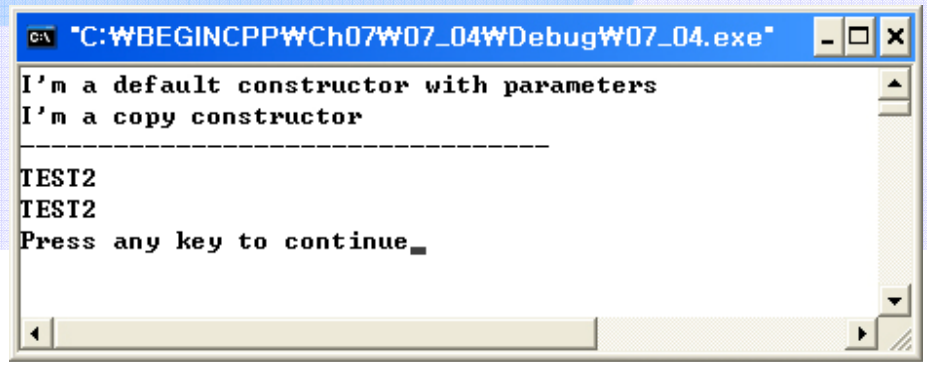


[기본예제 7-4] 클래스의 치환 #4 – 복사생성자 추가

```
...
07    Unit()
08    {
09        cout << "I'm a default constructor₩n";
10        str = NULL;
11    }
12
13    Unit(char* agStr)
14    {
15        cout << "I'm a default constructor with parameters₩n";
16        int strLength = strlen(agStr);
17        str = new char[strLength + 1];
18        strcpy(str, agStr);
19    }
20    Unit(Unit& agUnit)
21    {
22        cout <<  "I'm a copy constructor₩n";
23        str = agUnit.str;
24    }

...
```

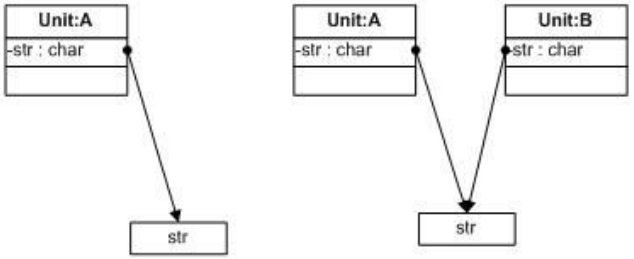
복사 생성자가 실행된 것을 화면에 표시한다.



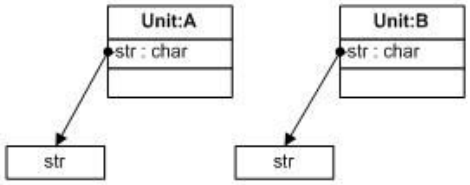
01_객체의 치환

❖ 얕은 복사와 깊은 복사

- 얕은 복사란, 우리가 앞에서 본 예제들과 같이 B 객체를 생성하면서, 이미 A가 만들어 놓은 메모리의 주소만을 참조하도록 복사하는 것을 의미한다.



- 깊은 복사란, 얕은 복사와는 달리 객체들간의 간섭이 일어나지 않는 아래와 같은 형태를 의미한다.

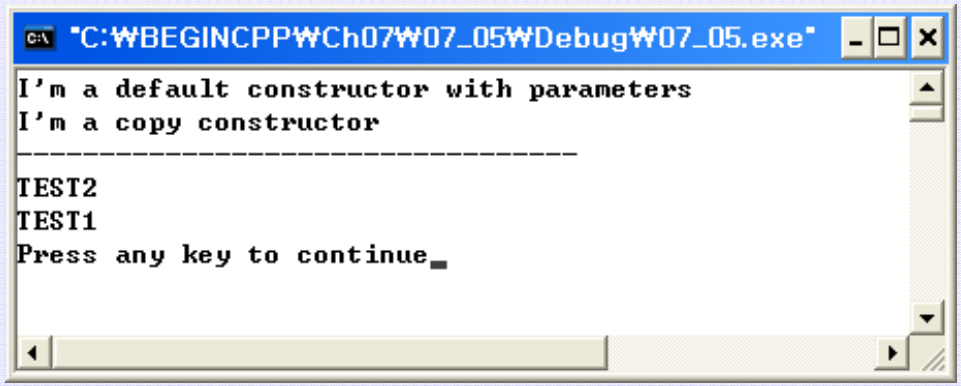


[기본예제 7-5] 깊은 복사 – 7-4에서 복사생성자만을 수정.

```
...
07    Unit()
08    {
09       cout << "I'm a default constructor₩n";
10       str = NULL;
11    }
12
13    Unit(Unit& agUnit)
14    {
15       cout <<  "I'm a copy constructor₩n";
16       str = new char[strlen(agUnit.str) + 1];
17       strcpy(str, agUnit.str);
18    }
...
```

깊은 복사를 하기 위해서 메모리를 새로 할당한다.

문자열을 복사한다.



02_함수와 클래스의 상관관계

❖ 함수의 입력인자로 전달하는 클래스 객체

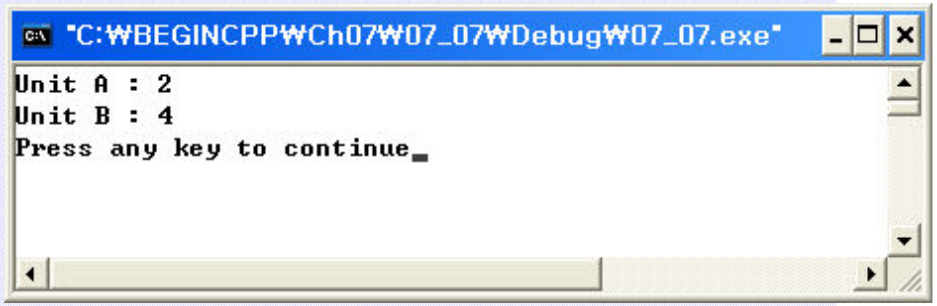
▪ 클래스 객체를 함수에 전달하려면?

- 함수의 입력인자로 다른 기본 변수형들을 전달하는 것과 동일하다.
- 즉, 함수의 입력변수에 대한 정의를 클래스형으로 선언한 다음, 함수를 호출할 때 그 클래스의 객체를 인자로 사용하면 된다.

[기본예제 7-7] 입력인자로 전달하는 클래스 객체

```
01  #include <iostream>
    using namespace std;
02  class Unit {
03          int defaultArmor;
04  public:
05          Unit(int armor) { defaultArmor = armor; }
06          int GetDefaultArmor() { return defaultArmor; }
07  };
08
09  int GetEnhancedArmor(Unit aUnit)
10  {
11          return aUnit.GetDefaultArmor() * 2;
12  }
13
14  int main(int argc, char* argv[])
15  {
16          Unit A(1), B(2);
17          cout << "Unit A : " << GetEnhancedArmor(A) << "₩n";
18          cout << "Unit B : " << GetEnhancedArmor(B) << "₩n";
19          return 0;
20  }
```

클래스가 입력인자인 함수를 정의한다.



A와 B 클래스 객체를 입력인자로 함수를 수행한다.

02_함수와 클래스의 상관관계

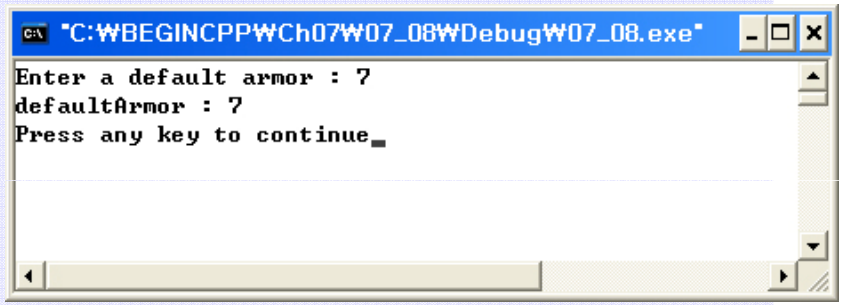
❖ 함수로부터 반환되는 클래스 객체

- 함수에게 객체를 입력인자로 전달할 수 있는 것과 마찬가지로, 함수도 클래스 객체를 반환할 수 있다.

- 객체를 반환하기 위해서는 미리 선언한 후, 정상적인 return 문을 사용하여 반환하면 된다.

[기본예제 7-8] 반환되는 클래스 객체

```
01 #include <iostream>
02  using namespace std;
03 class Unit {
04         int defaultArmor;
05 public:
06         void Show() { cout << "defaultArmor : " << defaultArmor << "₩n"; }
07         void Set(int armor) { defaultArmor = armor; }
08 };
09
10 Unit GetUnit()
11 {
12         int armor;
13         Unit temp;
14         cout << "Enter a default armor : ";
15         cin >> armor;
16         temp.Set(armor);
17         return temp;
18 }
19
20 int main(int argc, char* argv[])
21 {
22         Unit a;
23         a = GetUnit();
24         a.Show();
25         return 0;
26 }
```



클래스가 객체를 결과값으로 반환한다.

함수의 결과로 클래스 객체를 받아서 저장한다.

02_함수와 클래스의 상관관계

❖ Friend 함수

- 클래스의 멤버가 아니면서도, 클래스의 비공개 원소에 접근할 수 있게 해주는 함수이다.
- 하나의 함수가 두 개 이상의 서로 다른 클래스들의 비공개 멤버에 접근할 수도 있다.
- Friend 함수는 멤버함수가 아닌 일반 함수로 정의되며, 함수명 앞에 friend라는 키워드를 붙여서 선언한다.

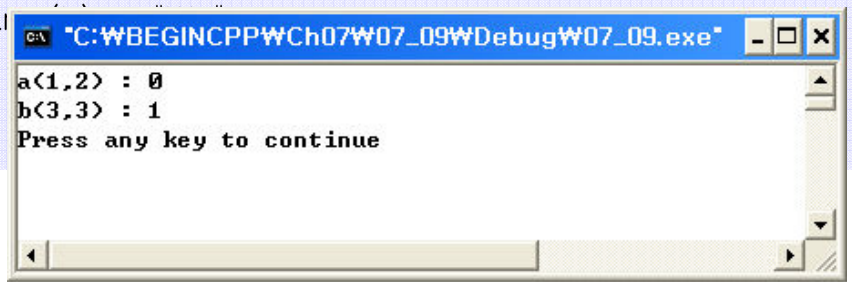
[기본예제 7-9] friend 함수를 이용한 비공개 멤버로의 접근

```
01  #include <iostream>
     using namespace std;
02  class Unit {
03          int x,y;
04  public:
05          Unit(int i, int j) { x=i; y=j; }
06          friend bool isSame(Unit a);
07  };
08
09  bool isSame(Unit a)
10  {
11          if (a.x == a.y ) return true;
12          else return false;
13  }
14
15  int main(int argc, char* argv[])
16  {
17                  Unit a(1,2), b(3,3);
18
19                  cout << "a(1,2) : " << isSame(a) << "Wn";
20                  cout << "b(3,3) : " << isSa
21          return 0;
22  }
```

Friend 함수를 정의한다.

Friend 함수의 몸체를 구현한다.

Friend 함수를 호출한다.



예제모음_12의 요구사항 및 실행결과

요구사항

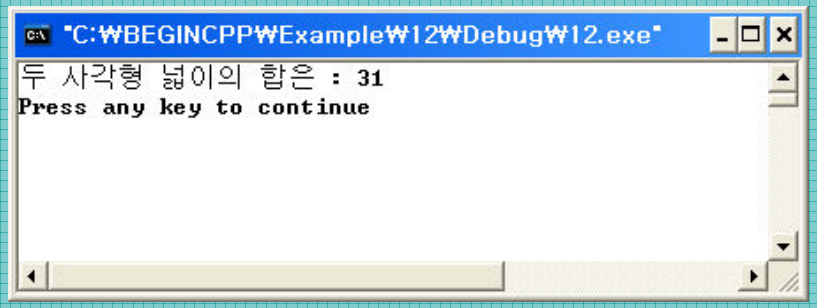
① 다음과 같은 사각형 클래스가 있다.

```
class Square{
public:
   int height,width;
   void addSquare();
};
```

사각형 클래스 객체를 입력으로 받아서, 자신과 입력 받은 사각형의 넓이의 합을 화면에 출력하도록 함수 addSquare를 수정하라.

② addSquare를 수정한 후, 적당한 생성자를 만들어서 5x3, 4x4인 두 객체에 대해 함수가 정상적으로 작동하는지 화면에 출력해본다.

실행결과

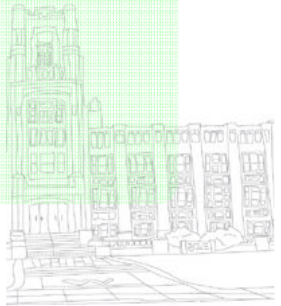


예제모음_12 소스

```
01  #include <iostream>
02  using namespace std;
03  class Square {
04  public:
05      int height,width;
06      Square(int x, int y)
07      {
08          height = x;
09          width = y;
10      }
11      void addSquare(Square A)
12      {
13          int area_A = A.height * A.width;
14          int area = height * width;
15          cout << "두 사각형 넓이의 합은 : " << area_A + area << "₩n";
16      }
17  };
18
19
20  int main(int argc, char* argv[])
21  {
22      Square A(5,3), B(4,4);
23      A.addSquare(B);
24      return 0;
25  }
```

생성자를 구현한다.

클래스를 입력인자로 갖는 멤버함수를 정의하여 넓이를 계산한다.



예제모음_13의 요구사항 및 실행결과

요구사항

① 다음과 같은 사각형 클래스가 있다.

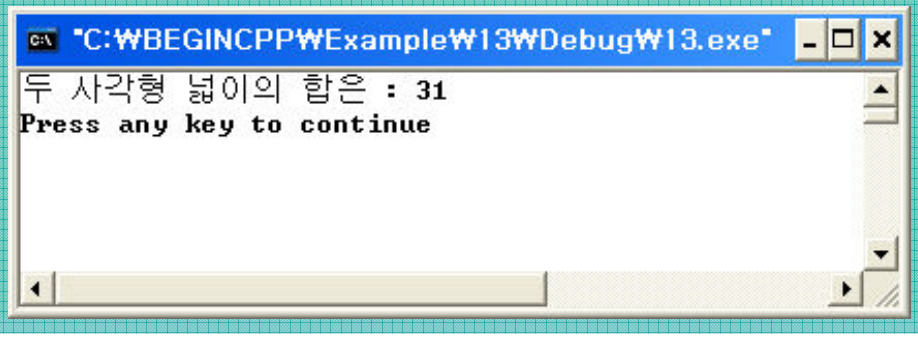
```
class Square{
   int height,width;
public:
   void addSquare();
};
```

사각형 클래스 객체를 입력으로 받아서, 자신과 입력 받은 사각형의 넓이의 합을 화면에 출력하도록 함수 addSquare를 수정하라.

② 클래스의 멤버변수인 height, width는 private로 고정하고, 프렌드 함수를 사용하도록 한다.

③ addSquare를 수정한 후, 적당한 생성자를 만들어서 5x3, 4x4인 두 객체에 대해 함수가 정상적으로 작동하는지 화면에 출력해 보도록 한다.

실행결과



예제모음_13 소스

```
01  #include <iostream>
02  using namespace std;
03  class Square {
04      int height,width;
05  public:
06      Square(int x, int y)
07      {
08          height = x;
09          width = y;
10      }
11      friend void addSquare(Square A, Square B)
12      {
13          int area_A = A.height * A.width;
14          int area_B = B.height * B.width;
15          cout << "두 사각형 넓이의 합은 : " << area_A + area_B << "Wn";
16      }
17  };
18
19
20  int main(int argc, char* argv[])
21  {
22      Square A(5,3), B(4,4);
23      addSquare(A, B);
24      return 0;
25  }
```

Friend함수를 정의하여 입력 받은 두 객체의 넓이의 합을 구한다.

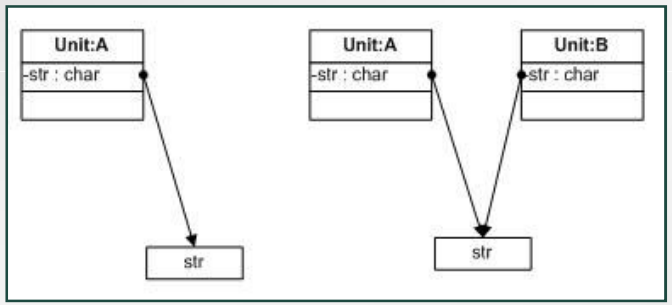
요약

클래스 객체의 치환

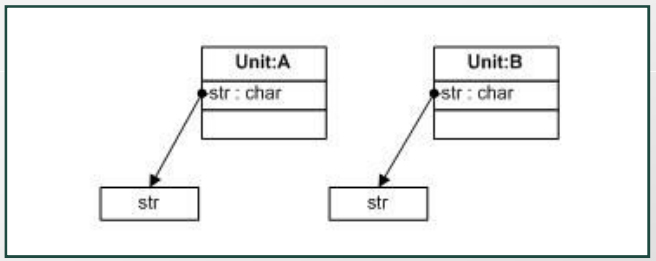
클래스 객체도 일반 변수와 동일하게 치환이 가능하다. 그러나
클래스 안에서 포인터를 다룰 경우에는 조심스럽게 치환해야 한다.

얕은 복사와 깊은 복사

메모리의 주소만을 참조하여 주소만 복사하는 것을 '얕은 복사(shallow copy)'라고 하며,
원본값까지 복사하는 것을 '깊은 복사(deep copy)'라고 한다.
얕은 복사로 인해 동작에 문제가 일어나는 경우에는 깊은 복사를 사용해야 한다.



〔얕은 복사의 예〕



〔깊은 복사의 예〕

함수와 클래스의 상관관계

함수의 입력인자로써 다른 기본 변수형들을 함수에 전달하는 것과 같은 방법으로 클래스 객체를 함수에 전달할 수 있으며,
함수의 출력인자로써도 동일한 방법으로 클래스 객체를 반환할 수 있다.

Friend 함수

Friend 함수란 클래스의 멤버가 아니면서 해당 클래스의 비공개 원소에 접근할 수 있는 함수를 뜻한다. 주로 이미 작성된 클래스의 멤버들을 크게 수정하지 않고 접근해야 할 필요성이 있을 때 사용된다.

Thank You !

IT CookBook for Beginner, C++ 기초 7장 끝