5.복사 생성자

- 초기화 스타일
- 복사 생성자 소개
- 깊은 복사(deep copy) 생성자
- 복사 생성자 호출 시기

Jong Hyuk Park

초기화 스타일

Jong Hyuk Park

C & C++ 스타일 초기화

❖ 두 가지 형태의 초기화

```
int main(void)
{
    int val1=20;
    AAA a1=10;
    . . . . .
```

```
int main(void)
{
    int val1(20);
    AAA a1(10);
    . . . . .
```

초기화 스타일 예

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

class AAA
{
	int val;
public:
	AAA(int n){
		val=n;
	}
	void ShowData(){
		cout<<val<<endl;
	}
};

int main(void)
{
	AAA a1(10);   // C++ 스타일 초기화
	a1.ShowData();
	AAA a2=20;    // C 스타일 초기화
	a2.ShowData();
	return 0;
}
```

복사 생성자 소개

Jong Hyuk Park

복사 생성자란?

❖ 복사 생성자

- 선언되는 객체와 같은 자료형의 객체를 인수로 전달하는 생성자
- 인수는 참조자(& 선언)로 전달
  - 참조자가 아니면 무한루프에 빠짐
- 전달되는 인수는 대개 const 선언

❖ 디폴트 복사 생성자

- 복사 생성자 정의 생략 시 자동으로 삽입되는 복사 생성자
- 인수로 전달되는 객체의 멤버변수를 선언되는 객체의 멤버변수로 복사

복사 생성자 예

```
class AAA
{
public:
    AAA(){
        cout<<"AAA() 호출"<<endl;
    }

    AAA(int i){
        cout<<"AAA(int i) 호출"<<endl;
    }

    AAA(const AAA& a){
        cout<<"AAA(const AAA& a) 호출"<<endl;
    }
};
```

```
int main(void)
{
    AAA obj1;
    AAA obj2(10);
    AAA obj3(obj2);
    return 0;
}
```

디폴트 복사 생성자 예

```
#include<iostream>
using std::cout;
using std::endl;
class Point
{
	int x, y;
public:
	Point(int _x, int _y){
		x=_x;
		y=_y;
	}

	void ShowData(){
		cout<<x<<' '<<y<<endl;
	}
};

int main(void)
{
	Point p1(10, 20);
	Point p2(p1);	// 디폴트 복사 생성자 호출
	p1.ShowData();
	p2.ShowData();
	return 0;
}
```

```
// 디폴트 복사 생성자와 같은 형태

Point(const Point& p){
	x=p.x;
	y=p.y;
}
```

깊은 복사(deep copy) 생성자 소개

Jong Hyuk Park

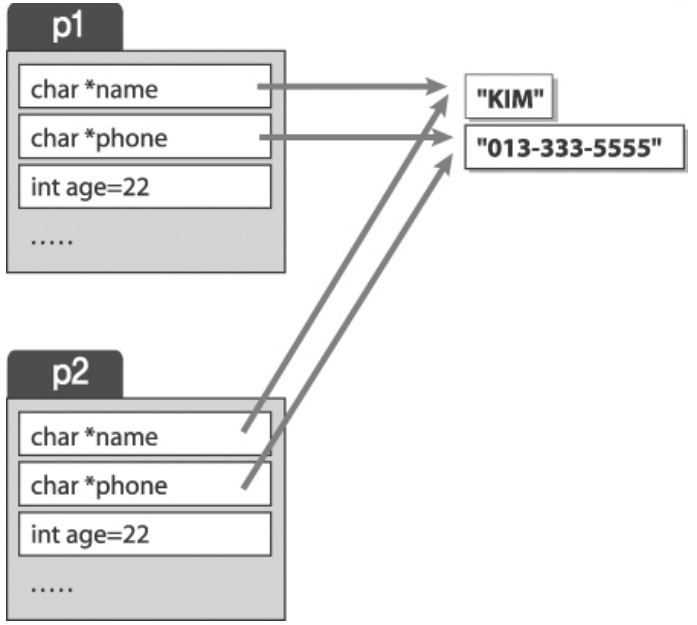
디폴트 복사 생성자 문제점 예 (1)

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

class Person
{
        char *name;
        char *phone;
        int age;
public:
        Person(char* _name, char* _phone, int _age);
        ~Person();
        void ShowData();
};

Person::Person(char* _name, char* _phone, int _age)
{
        name=new char[strlen(_name)+1];
        strcpy(name, _name);

        phone=new char[strlen(_phone)+1];
        strcpy(phone, _phone);

        age=_age;
}
```

```
Person::~Person()
{
        delete []name;
        delete []phone;
}
void Person::ShowData()
{
        cout<<"name: "<<name<<endl;
        cout<<"phone: "<<phone<<endl;
        cout<<"age: "<<age<<endl;
}

int main()
{
        Person p1("KIM", "013-333-5555", 22);
        Person p2=p1;   // Person p2(p1)

        return 0;
}
```

디폴트 복사 생성자 문제점 예 (2)

- 실행 시 얕은 복사(shallow copy)에 의한 메모리 참조 에러 발생
  - 객체 p1 선언 시 생성자에서 name, phone 메모리 할당
  - 객체 p2 선언 시 디폴트 복사 생성자에서 메모리 할당 없이 객체 p1의 name, phone의 포인터만 복사(얕은 복사)
  - p2 소멸자 호출하여 name, phone 포인터가 가리키는 메모리 해제
    - 소멸자 호출 순서는 생성자와 반대 순서
  - p1 소멸자 호출하여 name, phone 포인터가 가리키는 메모리 해제 시 실행 에러
    - 이미 p2 소멸자에 의해 해제 !!!
- 메모리를 할당하는 **깊은 복사**를 하는 생성자를 작성하여 해결

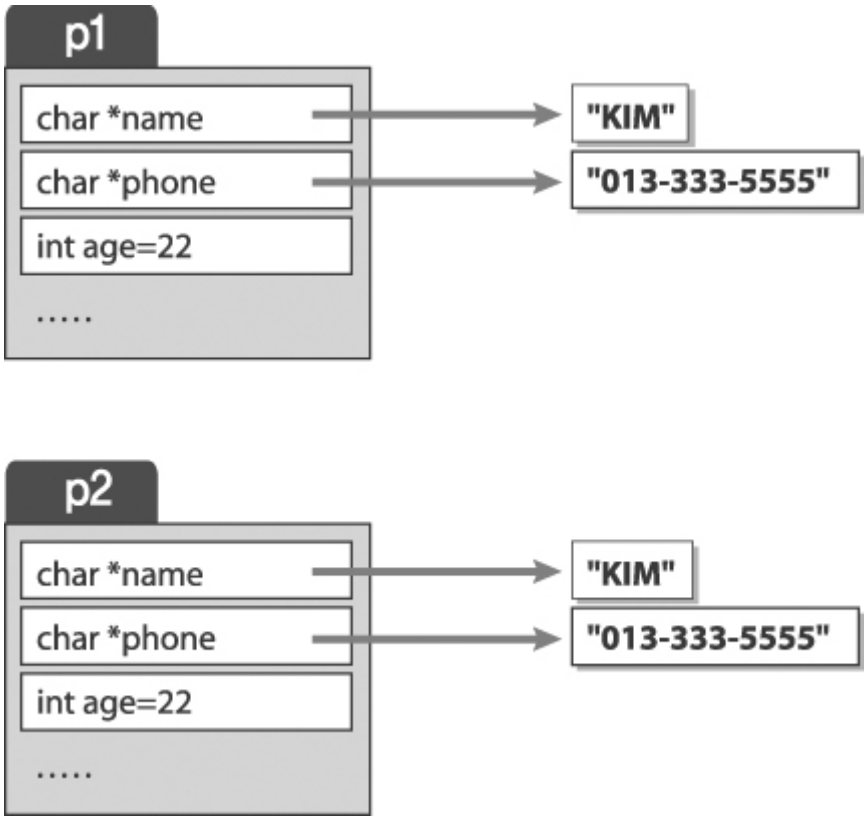


깊은 복사 생성자 예 (1)

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

class Person
{
        char *name;
        char *phone;
        int age;
public:
        Person(char* _name, char* _phone, int _age);
        Person(const Person& p);
        ~Person();
        void ShowData();
};

Person::Person(const Person& p)
{
        name=new char[strlen(p.name)+1];
        strcpy(name, p.name);

        phone=new char[strlen(p.phone)+1];
        strcpy(phone, p.phone);

        age=p.age;
}
```

```
Person::Person(char* _name, char* _phone, int _age)
{
        name=new char[strlen(_name)+1];
        strcpy(name, _name);
        phone=new char[strlen(_phone)+1];
        strcpy(phone, _phone);
        age=_age;
}
Person::~Person()
{
        delete []name;
        delete []phone;
}
void Person::ShowData()
{
        cout<<"name: "<<name<<endl;
        cout<<"phone: "<<phone<<endl;
        cout<<"age: "<<age<<endl;
}
int main()
{
        Person p1("KIM", "013-333-5555", 22);
        Person p2=p1;   // Person p2(p1)
        p1.ShowData();
        p2.ShowData();
        return 0;
}
```

깊은 복사 생성자 예 (2)


p1
char *name
char *phone
int age=22
.....
"KIM"
"013-333-5555"
p2
char *name
char *phone
int age=22
.....
"KIM"
"013-333-5555"

복사 생성자 호출 시기

Jong Hyuk Park

복사 생성자의 호출 시기

❖ 복사 생성자 호출 형태 3가지

- 기존에 생성된 객체로 새로운 객체 초기화

- 함수 호출 시 객체를 값에 의해 전달
  - 실인수를 전달받는 가인수를 위한 메모리 공간 할당
  - 전달되는 인수 값 복사

- 함수 내에서 객체를 값에 의해 리턴
  - 리턴되는 객체가 함술르 호출한 영역으로 복사되어 전달

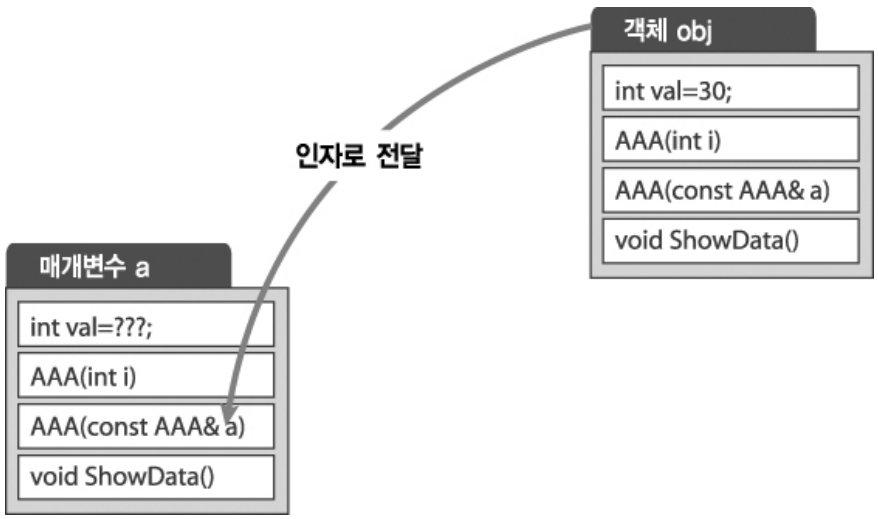
기존에 생성된 객체로 새로운 객체 초기화

```
class AAA
{
    int val;
public:
    AAA(int i){
        val=i;
    }
    AAA(const AAA& a){
        cout<<"AAA(const A& a) 호출"<<endl;
        val=a.val;
    }
    void ShowData(){
        cout<<"val: "<<val<<endl;
    }
};
```

```
int main()
{
    AAA obj1(10);
    AAA obj2=obj1;
    return 0;
}
```

함수 호출 시 객체를 값에 의해 전달 예

```
#include<iostream>
using std::cout;
using std::endl;
class AAA
{
        int val;
public:
        AAA(int i){
                val=i;
        }
        AAA(const AAA& a){
                cout<<"AAA(const A& a) 호출"<<endl;
                val=a.val;
        }
        void ShowData(){
                cout<<"val: "<<val<<endl;
        }
};
void function(AAA a)
{
        a.ShowData();
}
int main()
{
        AAA obj(30);
        function(obj);
        return 0;
}
```



객체의 인수 전달

❖ 객체가 인수로 전달되는 경우에는 실인수가 가인수에 복사(call by value)되어 전달

- 객체가 복사될 때에는 생성자를 호출하지 않지만 함수 종료로 복사된 가인수의 객체가 소멸할 때에는 소멸자를 호출

- 객체를 인수로 전달하는 경우에는 주의해서 사용
  - 동적으로 할당된 메모리를 해제하기 위해서 소멸자를 사용한 경우에는 함수의 종료 시에 소멸자를 호출하고 메모리를 해제
  - 이후 프로그램에서 이 동적 메모리를 사용하면 에러가 발생
  - 이를 피하기 위해 객체를 복사하지 않고 객체의 참조자로 인수를 전달하면 함수 종료시 소멸자가 호출되지 않음

객체의 인수 전달 예

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

class String {
        char *str;
public:
        String(char *);          // 생성자
        ~String();               // 소멸자
        char *get()          { return str; }
};
String::String(char *p)      // 생성자 외부 정의
{
   cout << " ! constructor₩n";
   str = new char[strlen(p)+1];
   strcpy(str,p);
}
String::~String()          // 소멸자 외부 정의
{
   cout << " ! destructor₩n";
   delete str;
}
```

```
void show1(String &s)
{   cout << "show1 : " << s.get(); }
void show2(String s)
{   cout << "show2 : " << s.get(); }
int main()
{
   String ss("C++ Language");
   show1(ss); cout << endl;
   show2(ss); cout << endl;
}
```

```
! constructor
show1 : C++ Language
show2 : C++ Language ! destructor

! destructor
```

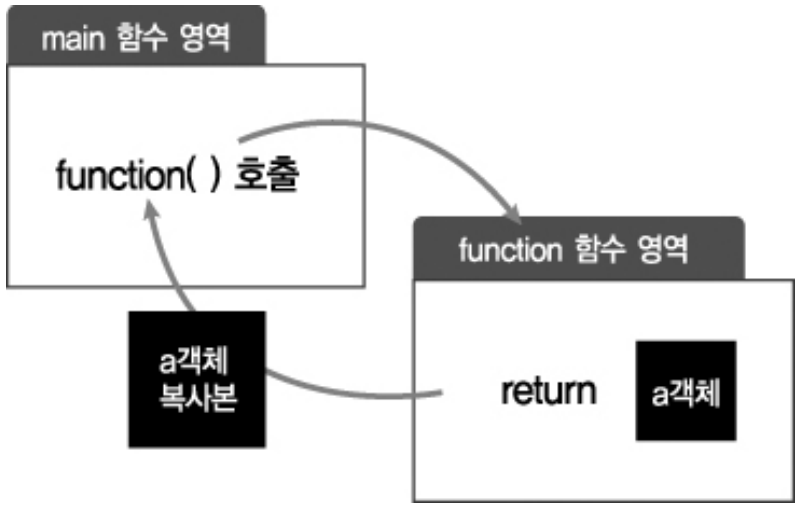
<= 에러

show1() 함수는 참조자를 인수로 전달하므로 소멸자를 호출하지 않는다. show2() 함수는 객체를 인수로 전달하므로 s에 객체가 복사되고 함수의 종료 시에 소멸자를 호출하여 str이 가리키는 메모리를 해제한다. 따라서 main() 프로그램의 종료 시에 객체 ss를 위한 소멸자를 호출 하게 되나 이미 메모리가 해제되었으므로 할당된 상태에 따라 에러 메시지를 출력한다.

함수 내에서 객체를 값에 의해 리턴 예

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

class AAA
{
	int val;
public:
	AAA(int i){
		val=i;
	}
	AAA(const AAA& a){
		cout<<"AAA(const A& a) 호출"<<endl;
		val=a.val;
	}
	void ShowData(){
	cout<<"val: "<<val<<endl;
	}
};
AAA function(void)
{
	AAA a(10);
	return a;
}
```

```
int main()
{
	function(); //function().ShowData();

	return 0;
}
```



과제

1. 연습문제 5-1: NameCard 클래스 프로그램

2. 4장의 은행계좌 관리 과제 프로그램에 다음을 추가하여라
   - 5-7절과 같은 복사 생성자 정의
   - *4장의 과제*
     - *고객의 전화번호 정보를 추가하여라. 전화번호는 다음과 같이 private으로 선언하라.*
       *char *phone;*
     - *(이름,전화번호)와 계좌번호로 잔액조회를 하는 Inquire() 오버로딩 함수를 작성하여라.*
       - *void Inquire(char *name, char *phone);*
       - *void Inquire(int id);*

Q & A

Jong Hyuk Park