제 6장. 생성자와 소멸자

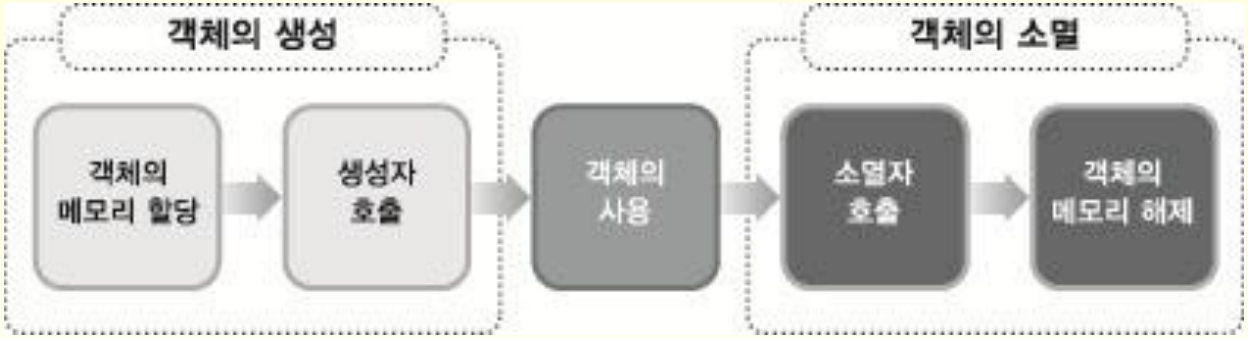
목포해양대 해양전자통신공학부

학습 목표

- 클래스의 객체를 초기화하기 위한 방법을 알아본다.
- 생성자와 기본 개념을 이해하고 생성자를 정의하는 방법을 알아본다.
- 소멸자의 기본 개념을 이해하고 알고 소멸자를 정의하는 방법을 알아본다
- 복사 생성자에 대하여 알아본다.

객체의 생성 및 소멸

- 객체가 생성될 때 생성자가 자동으로 호출된다.
  - 생성자는 객체가 만들어질 때 객체를 초기화한다.
- 객체가 소멸될 때 소멸자가 자동으로 호출된다.
  - 소멸자는 객체가 소멸될 때 정리 작업을 수행한다.

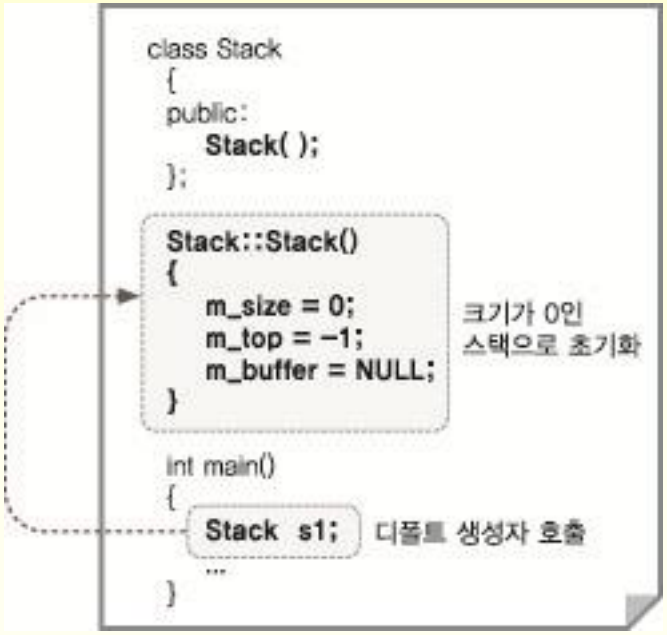


생성자와 소멸자

- 생성자와 소멸자는 자동으로 호출되는 함수이다.
- 생성자는 객체를 초기화하고, 소멸자는 객체를 정리한다.
- 생성자와 소멸자는 클래스 이름과 같은 이름의 함수이다. 생성자는 “클래스이름()” 형식의 함수이고 소멸자는 “~클래스이름()” 형식의 함수이다.
- 생성자와 소멸자는 리턴 값이 없으며 리턴 형으로 void를 사용하지도 않는다.
- 생성자는 인자를 가질 수 있고, 소멸자는 인자를 가질 수 없다.

디폴트 생성자

- 인자 없는 생성자를 말한다.
- 객체를 생성할 때 별도로 지정하지 않으면 항상 디폴트 생성자로 초기화된다.



인자 있는 생성자

- 객체를 특정 값으로 초기화하려면 생성자에 인자를 전달해서 처리한다.

```
void Stack::Initialize(int size)
{
    m_size = size;
    m_top = -1;
    m_buffer = new int[m_size];
    memset(m_buffer, 0, sizeof(int) * m_size);
}
```

변경

```
Stack::Stack(int size)
{
    m_size = size;
    m_top = -1;
    m_buffer = new int[m_size];
    memset(m_buffer, 0, sizeof(int) * m_size);
}
```

- 생성자는 오버로딩 가능하다.
- 생성자의 인자 전달 방법

변환 생성자

- 인자가 하나인 생성자는 형 변환에 이용될 수 있다.

```
class String {
protected:
char* m_str;          // 문자열을 저장하는 동적 메모리에 대한 포인터
public:
   String(const char* s);    // 변환 생성자
   …
};

int main()
{
   String s1 = "abc";              // String s1 = String("abc");로 처리된다.
   …
}
```

explicit

- 생성자에 explicit 키워드를 지정하면 해당 생성자가 명시적인 객체의 초기화에만 사용될 뿐 암시적인 형 변환에서는 사용될 수 없다.

```
class String {
public:
    explicit String(const char* s);
    ...
};

int main()
{
    String s1 = "abc";    // 암시적인 형 변환이 불가능하므로 컴파일 에러
    Test("Good-bye");    // 암시적인 형 변환이 불가능하므로 컴파일 에러
    String s1 = String("abc");          // 명시적 형 변환은 OK
    Test( String("Good-bye");          // 명시적 형 변환은 OK
}
```

초기화 리스트 (1)

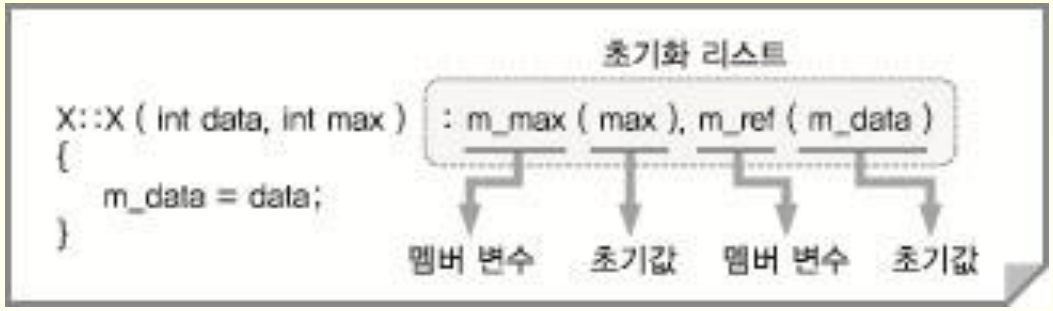
- const 변수나 레퍼런스 변수는 생성될 때 초기화하는 가능하지만 생성된 변수에 값을 대입하는 것은 불가능하다.
- 멤버 변수가 const 변수이거나 레퍼런스 변수일 때는 특별한 방법으로 초기화해야 한다. ➔ **초기화 리스트**

```
class X
{
protected:
   int m_data;
   const int m_max;        → 아직 메모리가 할당되지 않았으므로 초기화할 수 없다.
   int m_ref;
public:
   X(int data, int max);
};

X::X(int data, int max)
{
   m_data = data;
   m_max = max;            → 대입 연산으로 초기화할 수 없다.
   m_ref = m_data;
}

int main()
{
   X object(10, 100);
   ...
}
```

초기화 리스트 (2)

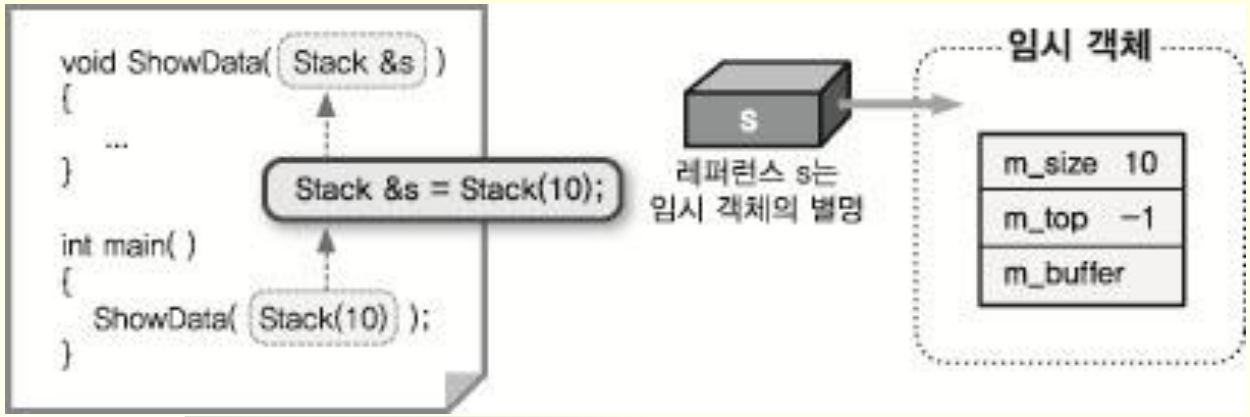
- 초기화 리스트

```
클래스이름::클래스이름( 인자리스트 )
: 멤버변수이름(초기값), 멤버변수이름(초기값), ...
{
}
```



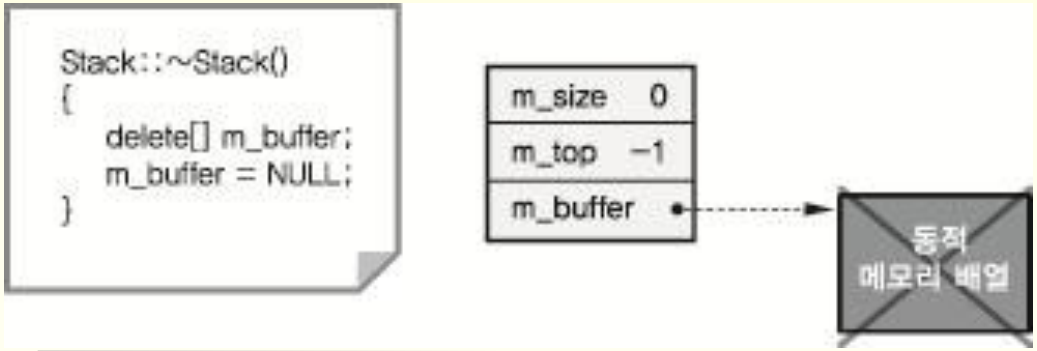
임시 객체

- 객체 없이 생성자를 호출하면 임시 객체가 생성된다.
  - 다음 문장으로 넘어갈 때 임시 객체는 자동으로 소멸된다.
- 함수에 인자를 전달하거나 형 변환을 수행할 때 유용하게 사용된다.



소멸자

- 소멸자는 객체가 소멸될 때 자동으로 호출되는 함수이다.
  - 소멸자는 객체가 소멸되기 전에 반드시 수행해야 할 정리 작업을 처리하는 데 사용된다.
- 소멸자는 "~클래스 이름()" 형식의 함수이다.
- 소멸자는 인자를 가질 수 없기 때문에 오버로딩할 수 없다.



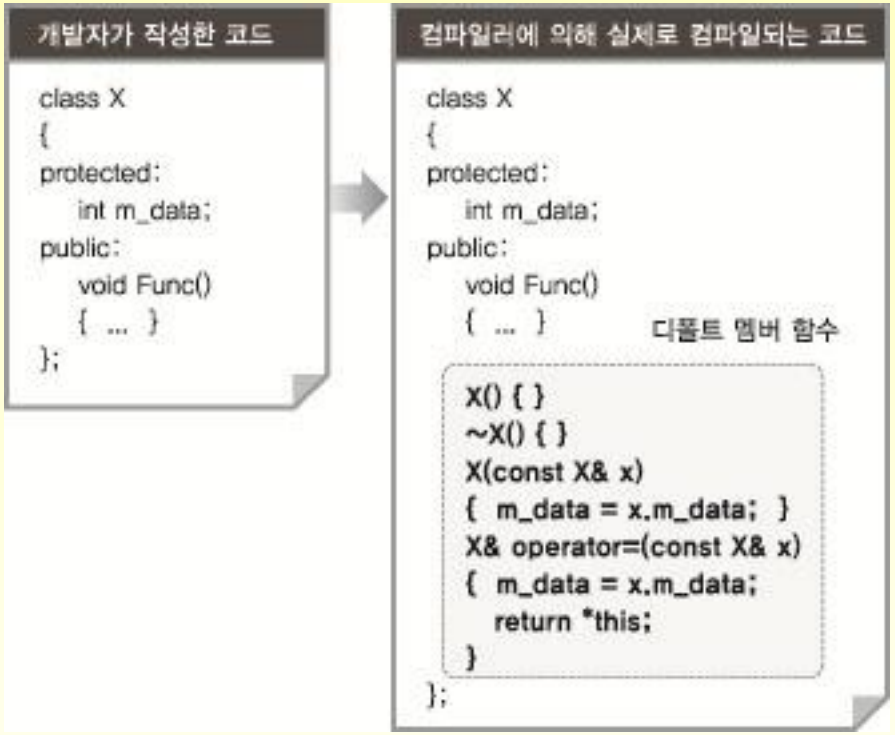
객체 간의 초기화와 대입

- 클래스의 객체는 같은 클래스의 객체 간에 서로 초기화나 대입이 가능하다.
- 객체 간의 초기화는 같은 클래스의 다른 객체와 같은 값을 갖도록 초기화하는 것이다.
  - 클래스의 멤버 변수를 1대1로 초기화한다.
- 객체 간의 대입은 같은 클래스의 다른 객체의 값을 대입하는 것이다.
  - 클래스의 멤버 변수를 1대1로 대입한다.

```
Stack s1(5);
Stack s2 = s1;        // 객체 간의 초기화
Stack s3;
s3 = s1;              // 객체 간의 대입
```

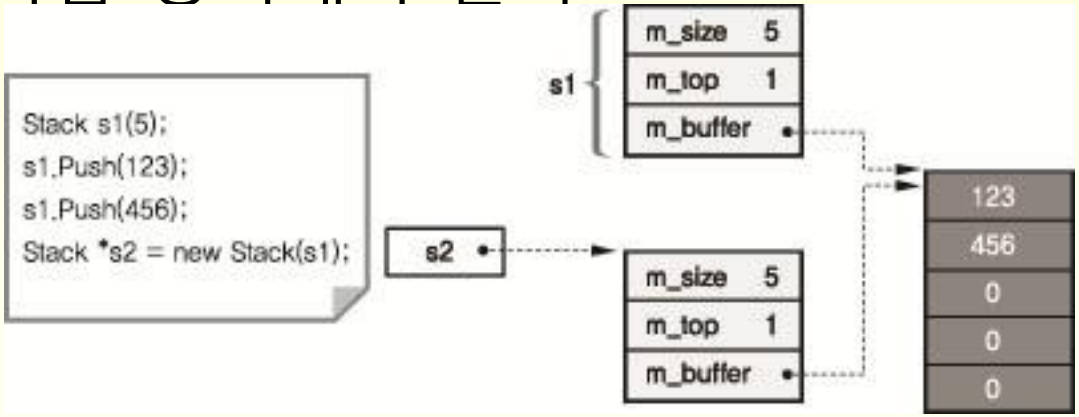
디폴트 멤버 함수

- 개발자가 따로 정의하지 않으면 컴파일러에 의해서 제공되는 함수
  - 디폴트 생성자
  - 디폴트 소멸자
  - 디폴트 복사 생성자
  - 디폴트 대입 연산자



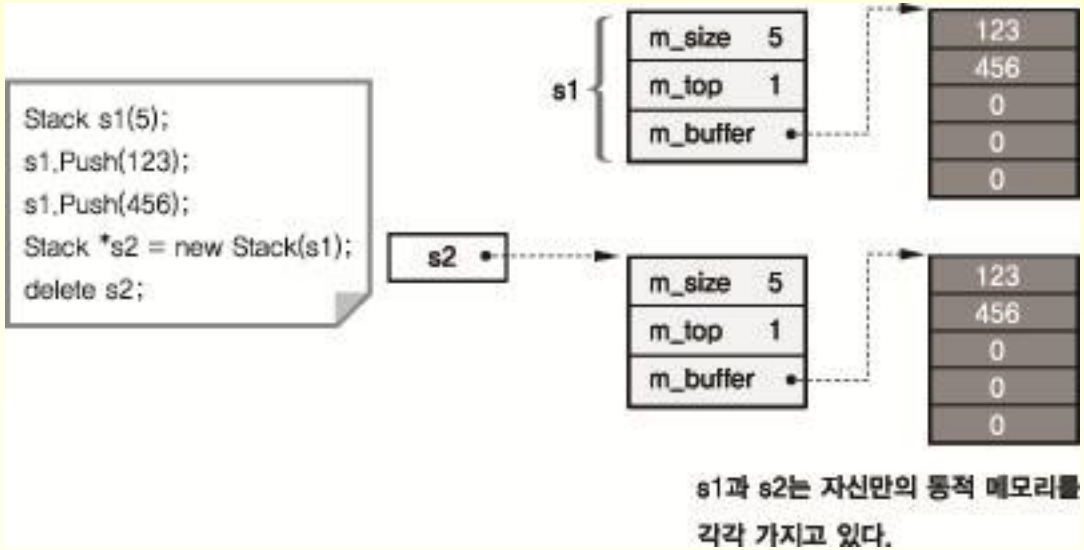
복사 생성자 (1)

- 멤버 변수 중에 포인터 변수가 있을 때 포인터 변수가 가리키는 데이터를 복사하는 대신 주소만 복사하는 것을 얕은 복사(shallow copy)라고 한다.
- 클래스 안의 멤버 변수 중 포인터 변수가 있고, 포인터 변수가 동적 메모리를 가리킬 때 올바른 처리를 위해서 복사 생성자, 대입 연산자, 소멸자 등을 직접 정의해야 한다.



복사 생성자 (2)

- 포인터의 주소를 복사하는 대신 포인터가 가리키는 데이터를 복사하는 것을 깊은 복사(deep copy)라고 한다.



복사 생성자 (3)

```
class Stack
{
    ...

    Stack(const Stack& s);   // 복사 생성자
};

Stack::Stack(const Stack& s)
{
    m_size = s.m_size;
    m_top = s.m_top;
    m_buffer = new int[m_size];
    memcpy(m_buffer, s.m_buffer, sizeof(int) * (m_top+1));
}
```

정리

- 클래스의 객체가 생성될 때 생성자가 자동으로 호출되고, 클래스의 객체가 소멸될 때 소멸자가 자동으로 호출된다.
- 생성자는 객체를 사용할 수 있도록 '준비'하는 함수이고, 소멸자는 객체의 '정리'를 수행하는 함수이다.
- 생성자는 "클래스 이름()" 형식의 함수이고 소멸자는 "~클래스 이름()" 형식의 함수이다.
- 생성자는 오버로딩 가능하고, 소멸자는 오버로딩할 수 없다.
- 생성자 중 인자가 하나인 생성자를 변환 생성자라고 한다.
- 멤버 변수 중에 반드시 초기화해야하는 멤버 변수, 즉 const 멤버 변수나 레퍼런스 멤버 변수는 초기화 리스트를 이용해서 초기화한다.
- 같은 클래스의 객체 간에 초기화나 대입이 가능하다.
- 같은 클래스의 객체를 이용해서 초기화하는 생성자를 복사 생성자라고 한다.
- 복사 생성자를 따로 정의하지 않으면 컴파일러가 디폴트 복사 생성자를 제공한다.

실습과제

- 실습과제 1, 2 [500점]
- 중간고사 문제 개선 [500점]
  - 6장 개념 도입(타자, 투수 생성자 구현)
  - 선수 10명 이상으로 구성된 야구팀 클래스 생성
  - 타자 문제(학번 : 홀수)
    - 팀내 타율, 안타 수 등이 제일 좋은 선수의 정보를 출력하는 멤버 함수 작성
  - 투수 문제(학번 : 짝수)
    - 팀내 승률, 방어율 등이 제일 좋은 선수의 정보를 출력하는 멤버 함수 작성