Lecture 9

Object-Oriented Programming V

Constructors, destructors, and memory leak

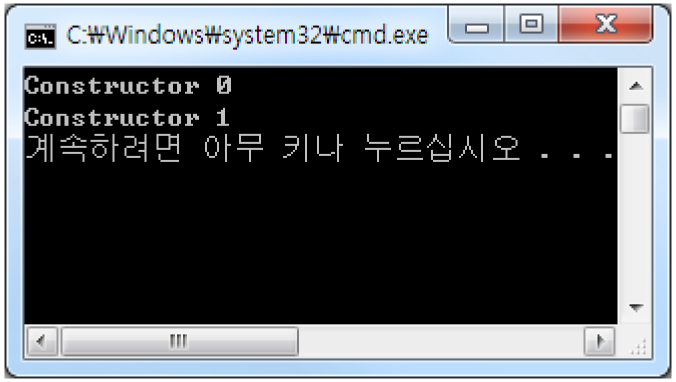
Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

Constructors

- Special member functions that are executed whenever we create a new class object.
  - Usually, the job of a constructor is to ensure that the datafields of an object start out with sensible initial values.

```cpp
#include <iostream>

class Box {
public:
   Box() {
      height = width = length = 0;
      std::cout << "Constructor 0" << std::endl;
   }
   Box(double h, double w, double l) {
      height = h; width = w; length = l;
      std::cout << "Constructor 1" << std::endl;
   }

   double height, width, length;
};

void main() {
   Box box1;
   Box box2(1,1,1);
}
```


C:₩Windows₩system32₩cmd.exe

```
Constructor 0
Constructor 1
계속하려면 아무 키나 누르십시오 . . .
```

Constructors

- We can initialize member variables with the constructor initializer.

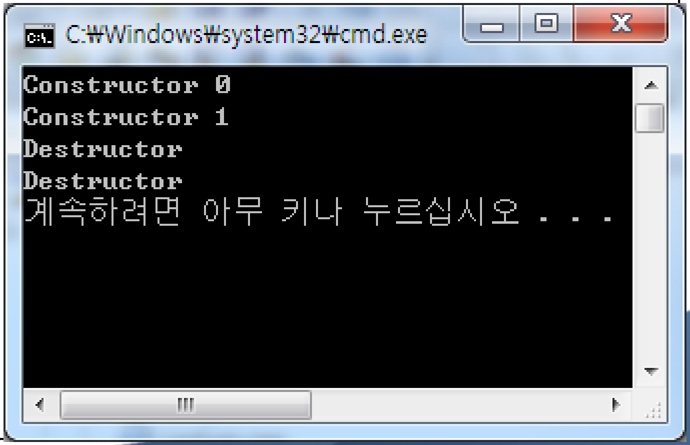
```
#include <iostream>

class Box {
public:
    Box() : height(0), width(0), length(0) {
        std::cout << "Constructor 0" << std::endl;
    }
    Box(double h, double w, double l) : height(h), width(w), length(l) {
        std::cout << "Constructor 1" << std::endl;
    }

    double height, width, length;
};

void main() {
    Box box1;
    Box box2(1,1,1);
}
```

Constructor Initializer

Destructor

- Complementary to the constructor.
  - Called automatically when an object goes out of scope or when a dynamically allocated object is deleted.

```cpp
#include <iostream>

class Box {
public:
     Box() {
       height = width = length = 0;
       std::cout << "Constructor 0" << std::endl;
    }
    Box(double h, double w, double l) : height(h), width(w), length(l) {
       std::cout << "Constructor 1" << std::endl;
    }
    ~Box() {
       std::cout << "Destructor" << std::endl;
    }

    double height, width, length;
};

void main() {
    Box box1;
    Box box2(1,1,1);
}
```


C:\Windows\system32\cmd.exe

```
Constructor 0
Constructor 1
Destructor
Destructor
계속하려면 아무 키나 누르십시오 . . .
```

Memory Leak

- Failure to delete dynamically allocated memory can lead to **“memory leak”**.

```
#include <iostream>
#include <string>

using namespace std;

void print() {
   string* str = new string("Programming Methodology is easy");
   cout << *str << endl;

}

void main() {
   while(1) print();
}
```

Causes
Memory Leak

How to Avoid Memory Leak?

- Objects created with the “new” operator needs an explicit delete.

```
#include <iostream>
#include <string>

using namespace std;

void print() {
   string* str = new string("Programming Methodology is easy");
   cout << *str << endl;

}

void main() {
   while(1) print();
}
```

Causes
Memory Leak

How to Avoid Memory Leak?

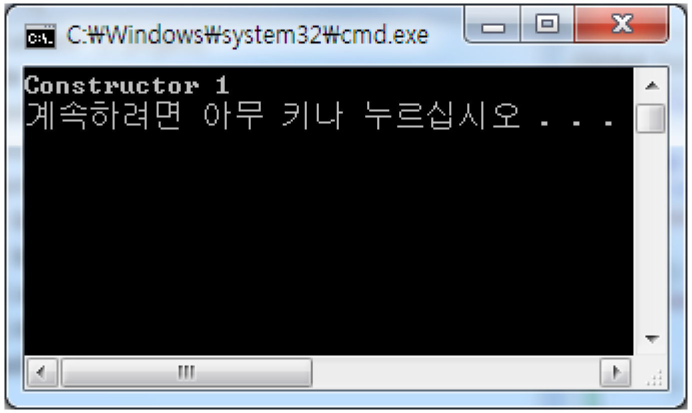
- Objects created with the “new” operator needs an explicit delete.

```
#include <iostream>
#include <string>

using namespace std;

void print() {
   string* str = new string("Programming Methodology is easy");
   cout << *str << endl;
   delete str;
}

void main() {
   while(1) print();
}
```

Memory Leak in a User Defined Class

```
#include <iostream>

class Array {
public :
   Array() : ptr(NULL) {
      std::cout << "Constructor 0" << std::endl;
   }
   Array(std::size_t num) {
      ptr = new int[num];
      std::cout << "Constructor 1" << std::endl;
   }

   int * ptr;
   int a, *b;
};

void f() {
   Array var(10);
}

void main() {
   f();    // There is memory leak in f().
           // Memory pointed by ptr is not deleted.
}
```

Causes Memory Leak

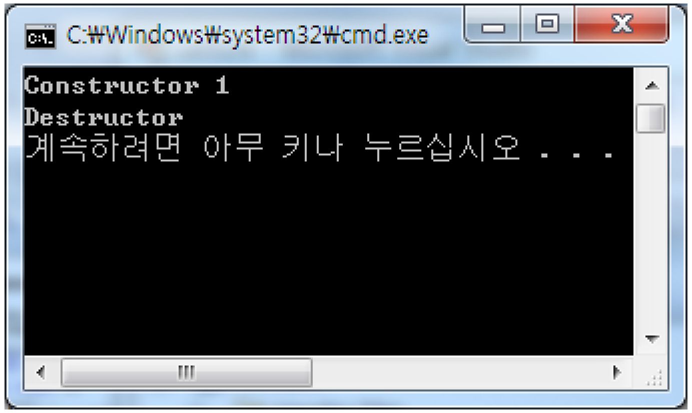


```
Constructor 1
계속하려면 아무 키나 누르십시오 . . .
```

How to Prevent Memory Leak in a User Defined Class?

- You can prevent memory leak by defining the destructor properly.

```cpp
#include <iostream>

class Array {
public :
   Array() : ptr(NULL) {
      std::cout << "Constructor 0" << std::endl;
   }
   Array(std::size_t num) {
      ptr = new int[num];
      std::cout << "Constructor 1" << std::endl;
   }
   ~Array() {
      if(ptr != NULL) delete [] ptr;
      std::cout << "Destructor" << std::endl;
   }

   int * ptr;
   int a, *b;
};

void f() {
   Array var(10);
}

void main() {
   f();    // There is no memory leak in f()
}
```



```
Constructor 1
Destructor
계속하려면 아무 키나 누르십시오 . . .
```