창의적 소프트웨어 설계

3주차 실습 – 클래스 기초

노인우, inwoo13@hanyang.ac.kr
한중수, wndtnsla@hanyang.ac.kr

Overview

목표

- C/C++ Struct
  - initialize
  - user-defined type

- C++ Class
  - information hiding - what for?
  - member functions
  - this – **pointer** to the instance
  - class vs instance
  - const / const member function
  - reference (&)
  - const reference
  - constructor & destructor
  - scope example
  - static members

C/C++ Struct

- C의 struct와 C++의 struct는 다르다

- 다음 두 질문은 다른 것:
  - “C의 구조체와 C++의 클래스 차이가 뭔가요?”
  - “C++에서 구조체와 클래스의 차이가 뭔가요?”

- C++에서, 구조체와 클래스는 ‘기본 접근 제어가 public인가, private인가’ 이외에는 같다

Information Hiding

- Information Hiding (Access Control)

- Member Functions & Variables
  - public
  - private
  - protected

Information Hiding – Example (1)

```
#include <iostream>

class TestClass{
   private:
   int m_index;

   public:
   int* getIndexPointer(){
      return &m_index;
   }

   int getIndexValue(){
      return m_index;
   }
};
```

Information Hiding – Example (2)

```
int main(){
    TestClass*  tc = new TestClass;
    int*  pindex = tc->getIndexPointer();

    std::cout << "Address of Index: " << pindex << std::endl;

    // private 멤버변수에 포인터를 이용한 외부 접근
    std::cout << "Value of Index: " << *pindex << std::endl;

    return 0;
}
```

```
Address of Index: 0xfa1c20
Value of Index: 0
```

this pointer

```
#include <iostream>

class TestClass{
   private:
   int m_index;

   public:
   void showIndexValue(){
      int m_index = 10;
      std::cout << "[TestClass][showIndexValue]  member: " << this->m_index  << std::endl;
      std::cout << "[TestClass][showIndexValue]  local: " << m_index << std::endl;
   }
};

int main(){
   TestClass*  tc = new TestClass;
   tc->showIndexValue();

   return 0;
}
```

this pointer

◆ 출력 결과:

- this 포인터 없이 이름만 명시할 경우, 멤버 변수 보다 지역 변수를 우선 참조한다

```
[TestClass][showIndexValue] member: 0
[TestClass][showIndexValue] local: 10
```

C/C++ const

- const variable
  - 인자로 넘겨 받았을 때 값을 변경 할 수 없다

- const member function
  - 멤버 변수의 값을 변경 할 수 없다
  - 지역 변수의 값은 변경 가능

C/C++ const member variable

```
#include <iostream>

class TestClass{
   private:
   const int m_height = 173;
};

int main(){
   TestClass*  tc = new TestClass;
   tc->m_height = 190;

   return 0;
}
```

C/C++ const member variable

◆ 출력 결과:

```
imtutor@imtutor-desktop:~/class_materials/0919$ g++ example.cpp
example.cpp:5:26: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11
    const int m_height = 173;
                         ^
example.cpp: In function ‘int main()’:
example.cpp:5:26: error: ‘const int TestClass::m_height’ is private
example.cpp:10:9: error: within this context
    tc->m_height = 190;
      ^
example.cpp:10:18: error: assignment of read-only member ‘TestClass::m_height’
    tc->m_height = 190;
```

C/C++ const member function

```
#include <iostream>

class TestClass{
   private:
   int m_weight;

   public:
   int setWeight(int new_weight){
       this->m_weight = new_weight;
       return this->m_weight;
   }

   int getWeight() const{
      m_weight += 1;
      return m_weight;
   }
};
```

```
int main(){
   TestClass*  tc = new TestClass;
   tc->setWeight(75);
   int weight = tc->getWeight();

   std::cout << "weight: " << weight <<
std::endl;

   return 0;
}
```

C/C++ const member function

◆ 출력 결과:

```
example.cpp: In member function ‘int TestClass::getWeight()  const’:
example.cpp:12:18: error: assignment of member ‘TestClass::m_weight’  in read-only
object
        m_weight += 1;
```

C++ Reference

- Pointer
  - 포인터 변수는 독자적인 주소 공간을 갖는다

- Reference
  - 레퍼런스는 대상에 대한 참조자일 뿐,
  - 독자적인 주소공간을 갖지 않는다

C++ Reference

◆ Pointer vs Reference

	포인터	레퍼런스
선언 방법	int *pData;	int &rData;

NULL 값	NULL 허용
	int *pData = NULL; // 실행가능
	- 참조 대상을 아무때나 참조 가능
		NULL 불가
		int &rData = NULL; // 컴파일 에러
		int &rData = num;
		- 이런식으로 선언과 동시에 초기화를 해주어야 한다.

초기화 시	string name("Hyeven");
	string *pData = &name;
	- name의 주소값을 대입합니다.
		string name("Hyeven");
		string &rData = name;
		- name의 실제 값을 대입합니다.

대상 변경	가리키는 대상을 변경할 수 있음
	string name("Hyeven");
	string sname("wonjayk");
	string *pData = &name;
	*pData = &sname;
	- pData가 name을 가리키지 않고 sname을 가리키게 됨
	- 값이 변경되는 것이 아님
		가리키는 대상을 변경할 수 없음, 값을 변경
		string name("Hyeven");
		string sname("wonjayk");
		string &rData = name;
		rData = sname;
		- name의 값을 sname의 값으로 변경하는 결과

C++ Reference

예제

- 레퍼런스에 NULL을 할당 불가하다는 건 무슨 의미인가?
  - 레퍼런스는 가리킬 대상을 가져야 한다 (True)
  - 레퍼런스가 가리키는 대상이 NULL을 가질 수 없다 (False)

```
#include <iostream>

int main(){
    int* a = NULL;
    int* &ra = a;

    std::cout << "result: " << ra << std::endl;

    return 0;
}
```

```
imtutor@imtutor-desktop:~/class_materials/0919$  ./a.out
result: 0
```

C++ Const Reference

- 예제:

```
#include "stdio.h"

struct Triplet { int a, b, c; };

void TestConstReference(const  Triplet ct, const Triplet* cpt, const Triplet& crt) {
   ct.a = 10, cpt->b = 20, crt.c = 30; // All are errors.
   printf("%d, %d, %d\n", ct.a, cpt->b, crt.c);
}

int main() {
   Triplet triplet;
   triplet.a = 10, triplet.b = 20, triplet.c = 30;
   TestConstReference(triplet,  NULL, triplet); // Causes SEGFAULT.
   return 0;
}
```

Class const reference

◆ 결과:

```
imtutor@imtutor-desktop:~/class_materials/0919$ g++ example.cpp
example.cpp: In function ‘void TestConstReference(Triplet, const Triplet*, const
Triplet&)’:
example.cpp:5:10: error: assignment of member ‘Triplet::a’ in read-only object
    ct.a = 10, cpt->b = 20, crt.c = 30;// All are errors.
         ^
example.cpp:5:23: error: assignment of member ‘Triplet::b’ in read-only object
    ct.a = 10, cpt->b = 20, crt.c = 30;// All are errors.
                       ^
example.cpp:5:35: error: assignment of member ‘Triplet::c’ in read-only object
    ct.a = 10, cpt->b = 20, crt.c = 30;// All are errors.
```

Class Constructor / Destructor

```
#include <iostream>

class MacBook{
private:
   int m_price;
public:
   MacBook(int const _price){
      this->m_price = _price;
      std::cout << "MacBook $" << this->m_price << std::endl;
   }

   ~MacBook(){
      std::cout << "Destroying MacBook" << std::endl;
   }
};
```

Class Constructor / Destructor

```
int main(){
    MacBook* mbook = new MacBook(3000);

    delete mbook;

    return 0;
}
```

```
imtutor@imtutor-desktop:~/class_materials/0919$ g++ example.cpp
imtutor@imtutor-desktop:~/class_materials/0919$ ./a.out
MacBook $3000
Destroying MacBook
```

Class Static Members

- Static Member Variables
- Static Member Functions

Class Static Members

```
#include <iostream>

class MacBook{
private:
   static int version;
   int m_price;
public:
   MacBook(int const _price){
      this->m_price = _price;
      std::cout << "MacBook $" << this->m_price << std::endl;
   }

   ~MacBook(){
      std::cout << "Destroying MacBook Ver: " << version << std::endl;
   }

   static int setVersion(int _ver){
      this->version = _ver;
   }
};
```

```
int main(){
    MacBook* mbook = new MacBook(3000);
    MacBook* mbook_pro = new MacBook(5000);

    MacBook::setVersion(10);

    delete mbook;
    delete mbook_pro;
    return 0;
}
```

```
example.cpp: In static member function ‘static int MacBook::setVersion(int)’:
example.cpp:18:9: error: ‘this’ is unavailable for static member functions
        this->version = _ver;
        ^
```

참고자료

1. 포인터와 레퍼런스, http://wonjayk.tistory.com/253


Q&A

Appendix #1. 주석 버그

- 주석의 내용과 실제 코드가 다른 버그
  - 예: 코드를 수정하고 옛날 주석을 그대로 남겨두는 경우
- 동작에는 이상이 없으나 협업 & 관리 측면에서 버그
- 개발자 중에는 주석 버그 가능성 때문에 주석을 상세히 남기는 코딩 컨벤션을 기피하는 경우도 있음
- 그렇다면 대안은 무엇일까?