고급 C++ 프로그래밍 (성신여대)

원리를 알면 IT가 맛있다
IT COOKBOOK

1학기: c++프로그래밍(1-8장, 17)

1장 c++입문
2장 기본프로그래밍
3장 선택문
4장 수학함수, 문자, 문자열
5장 반복문
6장 함수
7장 1차원 배열
8장 다차원 배열
17장 재귀호출

2학기: 고급 c++프로그래밍(9-16장)

9장 객체와 클래스
10장 객체지향 개념
11장 포인터와 동적메모리 관리
12장 템플릿, 벡터, 스택
13장 파일 입력과 출력
14장 연산자 오버로딩
15장 상속과 다형성
16장 예외처리

평가방법 (비대면 전환시)

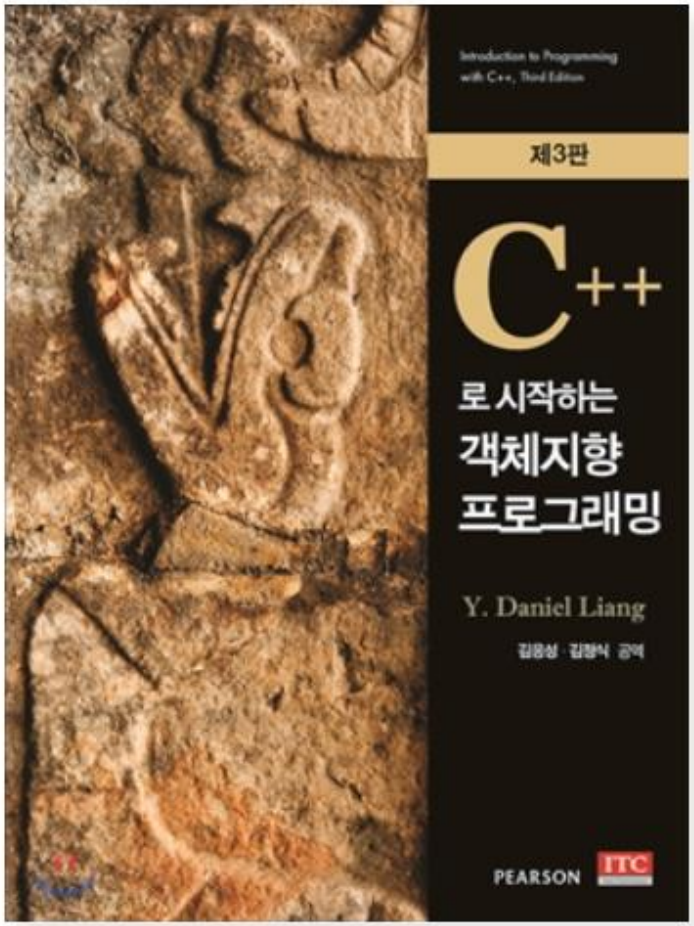
원리를 알면 IT가 맛있다
IT COOKBOOK

중간고사: 20 (지필)
기말고사: 30 (지필)
프로젝트 과제1: 15 (개인 프로젝트)
프로젝트 과제2: 15 (개인 프로젝트)
출석: 5
실습과제: 15(매주 실습과제를 각1점 혹은 2점으로 배점)

총점: 100

NOTE: 이번 학기는학교측의 방침에 따라 줌(ZOOM)을 통한 원격 실시간수업을 원칙으로 함

교재


Introduction to Programming
with C++, Third Edition
제3판
C++
로 시작하는
객체지향
프로그래밍
Y. Daniel Liang
PEARSON
ITC

구조체와 클래스

구조체

- 클래스는 c와 c++ 언어를 구분 짓는 가장 큰 특징이다.
- 클래스에 들어가기 전에 구조체(structure)의 정의를 어떻게 하는지 설명한다. 클래스는 구조체의 확장으로 이해할 수 있다.
- 구조체는 다양한 형태의 자료집합을 이용해 정의할 수 있다.

```
struct CDAccount
{
        double balance; //금액
        double interestRate; //이자율
        int term; //개월수
};

CDAccount account; //구조체 변수 선언
```

구조체 사용

- 구조체의 멤버변수에 접근하기 위해서는 . 연산자를 사용한다.

```
struct CDAccount
{
        double balance; //금액
        double interestRate; //이자율
        int term; //개월수
}; //구조체 선언

void main()
{
        CDAccount myAccount = { 1000000, 3.5, 12}; //선언과 동시초기화

        CDAccount yourAccount; //.을 이용한 멤버변수 접근
        yourAccount.balance = 2000000;
        yourAccount.interestRate = 3.8;
        yourAccount.term = 18;

        CDAccount hisAccount = yourAccount; //복사하기
}
```

클래스

- 클래스란 기본적으로 멤버 데이터뿐만 아니라 멤버 함수를 가지는 하나의 구조체이다.
- 클래스는 객체지향 프로그래밍의 기법에서 가장 중심 개념이다.
- 클래스형의 변수를 객체 또는 인스턴스라고 한다.

클래스


```
class CDAccount
{
    public:
        double balance; //금액
        double interestRate; //이자율
        int term; //개월수

        void print()
        {
            double result;
            result = balance + balance *(interestRate/100)* term/12;
            cout << fixed << setprecision(0) << result<< endl;
        }
}; //클래스 선언

void main()
{
        CDAccount myAccount; //객체 선언

        myAccount.balance = 2000000;
        myAccount.interestRate = 3.8;
        myAccount.term = 18;
        myAccount.print();
}
```

Chapter 9 객체와 클래스

객체를 위한 클래스 정의

- 객체 지향 프로그래밍(OOP, Object-Oriented Programming)에서는 프로그래밍에 객체(object)를 사용한다.
- 객체는 명확하게 구별되는 실제 세계에서의 개체(요소)를 나타낸다. 예를 들어, 학생, 책상, 원, 대여(물건을 빌리는 것)조차도 모두 객체로 볼 수 있다.
- 객체는 자신만의 유일한 특성과 상태(state), 행동(behavior)을 갖는다.
- 객체의 상태(state)는 데이터 필드 즉 변수로 표현된다.
- 객체의 행동(behavior)은 함수에 의해 정의 된다.

객체를 위한 클래스 정의


```
class Circle
{
public:
  // The radius of this circle
  double radius;

  // Construct a circle object
  Circle()
  {
    radius = 1;
  }

  // Construct a circle object
  Circle(double newRadius)
  {
    radius = newRadius;
  }

  // Return the area of this circle
  double getArea()
  {
    return radius * radius * 3.14159;
  }
};
```

클래스 정의와 객체생성의 예 (실습1)


```
#include <iostream>
using namespace std;

class Circle  //프로그래머 정의 클래스는 첫 문자를 대문자로 하는 것이 좋다
{
     public:
        double radius;  //반지름

        Circle() {
                radius = 1;
         }

        Circle(double newRadius){
                 radius = newRadius;
        }

        double getArea(){
                return radius * radius * 3.14159;
        }

}; // 세미콜론 반드시 있어야 함
```

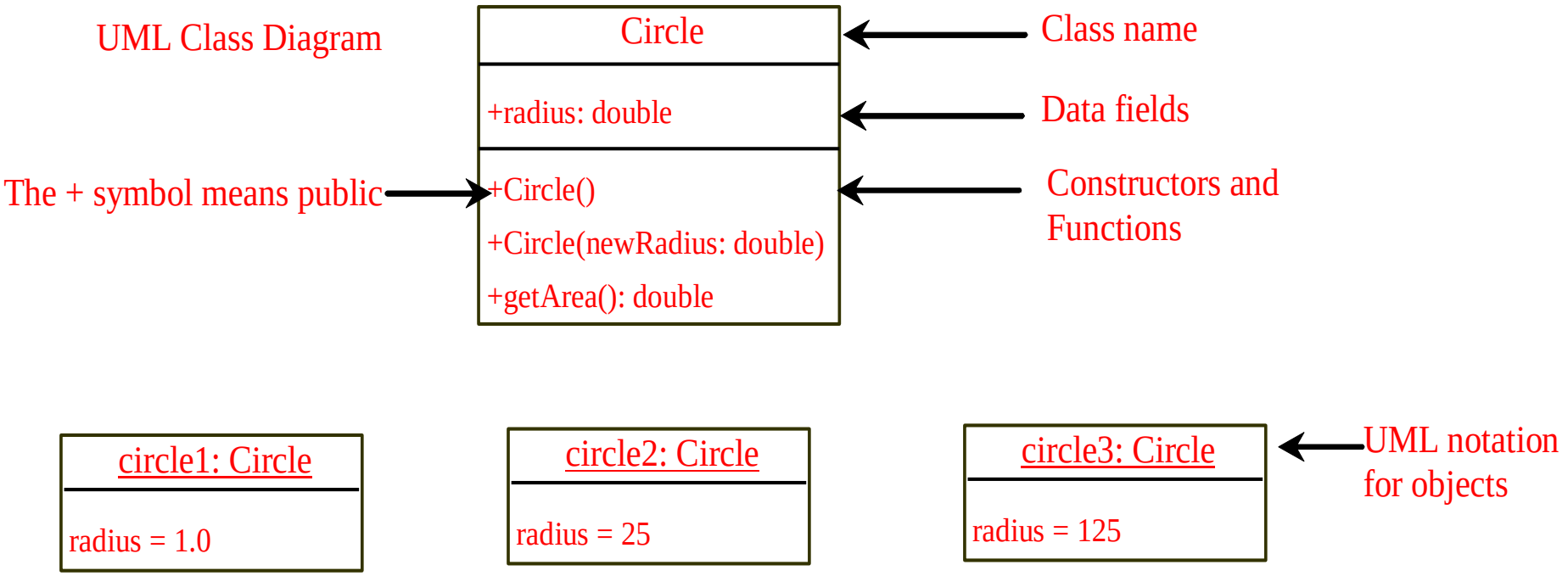
클래스 정의와 객체생성의 예(실습1)


```
int main()
 {
       Circle circle1;
       Circle circle2(25);
       Circle circle3(125);

       cout << "circle1 radius: " << circle1.radius <<
               " area: " <<   circle1.getArea() << endl;
       cout << "circle2 radius: " << circle2.radius <<
               " area: " << circle2.getArea() << endl;
       cout << "circle3 radius: " << circle3.radius <<
               " area: " << circle3.getArea() << endl;

       // Modify circle radius
       circle2.radius = 100;
       cout << "circle2 radius: " << circle2.radius <<
               " area: " << circle2.getArea() << endl;
       return 0;
  }
```

클래스 정의와 객체생성의 예



- 클래스(class)는 객체가 어떻게 생겨야 하는 지를 적어놓은 설계도와 같다.
- 객체(object)는 설계도를 통해 만들어진 인스턴스(instance)이다.
- 클래스에는 멤버변수를 초기화하기 위한 생성자가 있다.

생성자

- 생성자는 아주 특별한 기능을 가진 멤버함수이다.
- 클래스도 일반변수처럼 객체를 생성할 때 초기값을 줄 수 있어야 하는데, 이를 가능하게 하는 것이 생성자다.
- 생성자는 다음과 같은 특징이 있다
  - 생성자는 클래스 자신과 같은 이름을 가진다.
  - 생성자는 반환유형을 가질 수 없다
  - 생성자는 객체가 생성될 때 자동호출 된다.
  - 생성자도 오버로딩을 적용할 수 있다.
  - 접근 지정자가 public: 으로 설정되어야 한다.

생성자

- 프로그래머가 생성자를 만들지 않으면 C++ 컴파일러는 매개변수가 없고 내용도 없는 생성자를 자동으로 만들어 놓는다. 이 생성자를 '기본생성자' (default constructor)라고 하는데 아무런 일도 하지 않는다.

- 그런데 프로그래머가 매개변수를 갖는 생성자를 만들어주면 컴파일러는 더 이상 기본 생성자를 제공하지 않는다 즉, 매개변수가 있는 생성자를 만들어 사용하던 중 매개변수를 갖지 않는 생성자가 필요하다면 이를 프로그래머가 직접 정의하고 사용해야 한다.

객체 생성

- 인수 없는 생성자를 사용하여 객체를 생성하기 위한 구문은 다음과 같다.

```
Circle circle1;
```

- 인수 있는 생성자를 사용하여 객체를 생성하기 위한 구문은 다음과 같다.

```
Circle circle2(5.5);
```

생성자의 초기화목록

- 생성자에 초기화목록(initializer list)를 사용하여 초기화할 수도 있다.

```
Circle( )
{
      radius = 1;
}
```



```
Circle( ) : radius(1)
{
}
```

```
Circle(double newRadius) {
      radius = newRadius;
}
```

```
Circle(double newRadius )
     : radius(newRadius)
{
}
```

객체 구성과 사용

- 객체의 멤버란 데이터필드(ex) Circle 의 radius)와 함수(ex) Circle의 getArea())를 의미하며, 이를 각각 멤버변수와 멤버함수라 부른다.

- 객체가 생성된 후에는 점 연산자(.)를 사용하여 데이터에 접근하고 함수를 호출한다.

```
Circle circle1;
circle1.radius = 3;
circle1.getArea( );
```

- 객체의 멤버함수는 같은 클래스의 모든 객체에 공유되므로 단 하나의 복사본만 생성된다. 그러므로 객체의 실제크기는 크지 않다.

객체 구성과 사용

- C++에서 하나의 객체로부터 다른 객체로 내용을 복사하기 위해 대입 연산자(=)를 사용할 수 있다. 기본적으로 한 객체의 각각의 데이터 필드는 다른 객체의 대응 부분으로 복사된다.

```
circle2 = circle1;
```

- 복사 후에 circle1과 circle2는 여전히 두 개의 다른 객체이지만, radius의 값은 같은 값이 된다.