- 생성자
- 소멸자
- 객체배열
- 객체포인터
- 객체배열 동적생성/반환

클래스의 완성 실습

박 종 혁 교수

UCS Lab

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

예제1) 매개 변수를 가지는 생성자

```
#include <iostream>
#include <string>
using namespace std;
class Car {
private:        int speed;      // 속도
        int gear;               // 기어
        string color;           // 색상
public: Car(int s, int g, string c)
        {
                speed = s;
                gear = g;
                color = c;
        }
        void print()
        {       cout << "============== " << endl;
                cout << "속도: " << speed << endl;
                cout << "기어: " << gear << endl;
                cout << "색상: " << color << endl;
                cout << "============== " << endl;
        }
};
```

```
int main()
{
        Car c1(0, 1, "red");            // 생성자 호출
        Car c2(0, 1, "blue");  // 생성자 호출
        c1.print();
        c2.print();
        return 0;
}
```

```
==============
속도: 0
기어: 1
색상: red
==============
==============
속도: 0
기어: 1
색상: blue
==============
```

c1

speed	0
gear	1
color	"red"

c2

speed	0
gear	1
color	"blue"

예제 2) Rectangle 클래스 만들기

다음 main() 함수가 잘 작동하도록 Rectangle 클래스를 작성하고 프로그램을 완성하라. Rectangle 클래스는 width와 height의 두 멤버 변수와 3 개의 생성자, 그리고 isSquare() 함수를 가진다.

```
int main() {
  Rectangle rect1;
  Rectangle rect2(3, 5);
  Rectangle rect3(3);
  if(rect1.isSquare()) cout << "rect1은 정사각형이다." << endl;
  if(rect2.isSquare()) cout << "rect2는 정사각형이다." << endl;
  if(rect3.isSquare()) cout << "rect3는 정사각형이다." << endl;
}
```

```
rect1은 정사각형이다.
rect3는 정사각형이다.
```

예제2-프로그래밍)

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

class Rectangle {
public:
  int width, height;

  Rectangle();
  Rectangle(int w, int h);
  Rectangle(int length);
  bool isSquare();
};

Rectangle::Rectangle() {
  width = height = 1;
}

Rectangle::Rectangle(int w, int h) {
  width = w; height = h;
}
Rectangle::Rectangle(int length) {
  width = height = length;
}

// 정사각형이면 true를 리턴하는 멤버 함수
bool Rectangle::isSquare() {
  if(width == height) return true;
  else return false;
}
```

```
int main() {
  Rectangle rect1;
  Rectangle rect2(3, 5);
  Rectangle rect3(3);

  if(rect1.isSquare()) cout << "rect1은 정사각형이다."
<< endl ;
  if(rect2.isSquare()) cout << "rect2는 정사각형이다."
<< endl;
  if(rect3.isSquare()) cout << "rect3는 정사각형이다."
<< endl;
}
```

3 개의 생성자가 필요함

```
rect1은 정사각형이다.
rect3는 정사각형이다.
```

예제 3) Circle 클래스에 소멸자 작성 및 실행

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

class Circle {
public:
  int radius;

  Circle();
  Circle(int r);
  ~Circle(); // 소멸자
  double getArea();
};


Circle::Circle() {
  radius = 1;
  cout << "반지름 " << radius << " 원 생성" << endl;
}

Circle::Circle(int r) {
  radius = r;
  cout << "반지름 " << radius << " 원 생성" << endl;
}

Circle::~Circle() {
  cout << "반지름 " << radius << " 원 소멸" << endl;
}
```

```
double Circle::getArea()
{
  return 3.14*radius*radius;
}


int main() {
  Circle donut;
  Circle pizza(30);

  return 0;
}
```

main() 함수가 종료하면 main() 함수의 스택에 생성된 pizza, donut 객체가 소멸된다.

```
반지름 1 원 생성
반지름 30 원 생성
반지름 30 원 소멸
반지름 1 원 소멸
```

객체는 생성의 반대 순으로 소멸된다.

예제 3) 지역 객체와 전역 객체의 생성 및 소멸 순서

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

class Circle {
public:
  int radius;
  Circle();
  Circle(int r);
  ~Circle();
  double getArea();
};

Circle::Circle() {
  radius = 1;
  cout << "반지름 " << radius << " 원 생성" << endl;
}

Circle::Circle(int r) {
  radius = r;
  cout << "반지름 " << radius << " 원 생성" << endl;
}

Circle::~Circle() {
  cout << "반지름 " << radius << " 원 소멸" << endl;
}

double Circle::getArea() {
  return 3.14*radius*radius;
}
```

다음 프로그램의 실행 결과는 무엇인가?

```
Circle globalDonut(1000);
Circle globalPizza(2000);

void f() {
  Circle fDonut(100);
  Circle fPizza(200);
}

int main() {
  Circle mainDonut;
  Circle mainPizza(30);
  f();
}
```

전역 객체 생성

지역 객체 생성

지역 객체 생성

```
반지름 1000 원 생성
반지름 2000 원 생성
반지름 1 원 생성
반지름 30 원 생성
반지름 100 원 생성
반지름 200 원 생성
반지름 200 원 소멸
반지름 100 원 소멸
반지름 30 원 소멸
반지름 1 원 소멸
반지름 2000 원 소멸
반지름 1000 원 소멸
```

예제4) 객체배열

Circle 클래스의 2차원 배열 선언 및 활용

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

class Circle {
  int radius;
public:
  Circle() { radius = 1; }
  Circle(int r)  { radius = r; }
  void setRadius(int r) { radius = r; }
  double getArea();
};

double Circle::getArea() {
  return 3.14*radius*radius;
}
```

```
Circle circles[2][3] =
     { { Circle(1), Circle(2), Circle(3) },
       { Circle(4), Circle(5), Circle() } };
```

```
int main() {
  Circle circles[2][3];

  circles[0][0].setRadius(1);
  circles[0][1].setRadius(2);
  circles[0][2].setRadius(3);
  circles[1][0].setRadius(4);
  circles[1][1].setRadius(5);
  circles[1][2].setRadius(6);

  for(int i=0; i<2; i++)
        // 배열의 각 원소 객체의 멤버 접근
    for(int j=0; j<3; j++) {
      cout << "Circle [" <<i<< "," <<j<< "]의 면적은 ";
      cout << circles[i][j].getArea() <<endl;
    }
}
```

```
Circle [0,0]의 면적은 3.14
Circle [0,1]의 면적은 12.56
Circle [0,2]의 면적은 28.26
Circle [1,0]의 면적은 50.24
Circle [1,1]의 면적은 78.5
Circle [1,2]의 면적은 113.04
```

예제 5) 객체 포인터 선언 및 활용

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

class Circle {
  int radius;
public:
  Circle() { radius = 1; }
  Circle(int r)  { radius = r; }
  double getArea();
};

double Circle::getArea()
{
  return 3.14*radius*radius;
}
```

```
int main() {
  Circle donut;
  Circle pizza(30);

  // 객체 이름으로 멤버 접근
  cout << donut.getArea() << endl;

  // 객체 포인터로 멤버 접근
  Circle *p;
  p = &donut;
  cout << p->getArea() << endl; // donut의 getArea() 호출
  cout << (*p).getArea() <<endl; // donut의 getArea() 호출

  p = &pizza;
  cout << p->getArea() << endl; // pizza의 getArea() 호출
  cout << (*p).getArea() << endl; // pizza의 getArea() 호출
}
```

```
3.14
3.14
3.14
2826
2826
```

예제6) 객체 배열의 동적 생성과 반환 응용

원을 개수를 입력 받고 Circle 배열을 동적 생성하라.

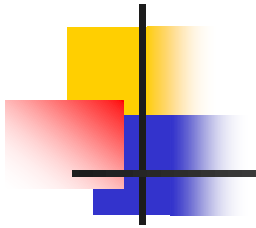
반지름 값을 입력 받아 Circle 배열에 저장하고, 면적이 100에서 200 사이인 원의 개수를 출력하라.

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

class Circle {
  int radius;
public:
  Circle();
  ~Circle() { }
  void setRadius(int r) { radius = r; }
  double getArea() { return 3.14*radius*radius; }
};

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

계속)

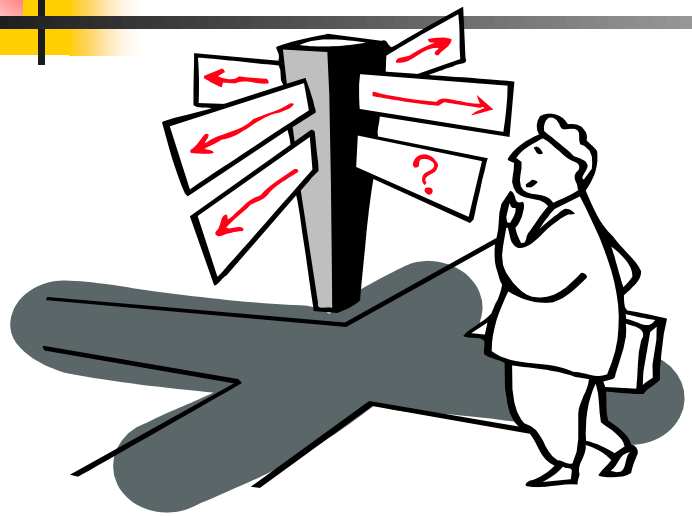
```
int main() {
  cout << "생성하고자 하는 원의 개수?";
  int n, radius;
  cin >> n; // 원의 개수 입력

  Circle *pArray = new Circle [n]; // n 개의 Circle 배열 생성
  for(int i=0; i<n; i++) {
    cout << "원" << i+1 << ": "; // 프롬프트 출력
    cin >> radius; // 반지름 입력
    pArray[i].setRadius(radius); // 각 Circle 객체를 반지름으로 초기화
  }

  int count =0; // 카운트 변수
  Circle* p = pArray;
  for(int i=0; i<n; i++) {
    cout << p->getArea() << ' '; // 원의 면적 출력
    if(p->getArea() >= 100 && p->getArea() <= 200)
      count++;
    p++;
  }
  cout << endl << "면적이 100에서 200 사이인 원의 개수는 "
      << count << endl;

  delete [] pArray; // 객체 배열 소멸
}
```

```
생성하고자 하는 원의 개수?4
원1: 5
원2: 6
원3: 7
원4: 8
78.5 113.04 153.86 200.96
면적이 100에서 200 사이인 원의 개수는 2
```



참고문헌

- 뇌를 자극하는 C++, 이현창 저, 한빛미디어
- C++ ESPRESSO, 천인국 저, 인피니티북스, 2011
- 명품 C++ Programming, 황기태 , 생능출판사, 2013



Q&A