

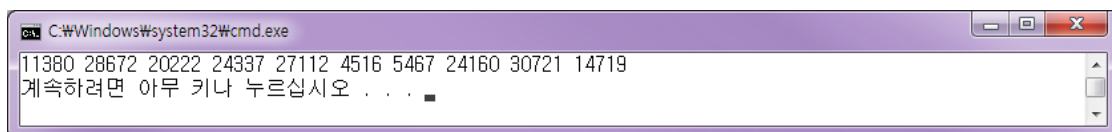
## 제8장 포인터와 동적객체 생성

1. 포인터의 개념을 이해한다.
2. 포인터와 관련된 연산을 이해한다.
3. 동적 메모리 할당을 사용할 수 있다.
4. 스마트 포인터를 사용할 수 있다.
5. 동적으로 객체를 생성하는 과정을 이해한다.

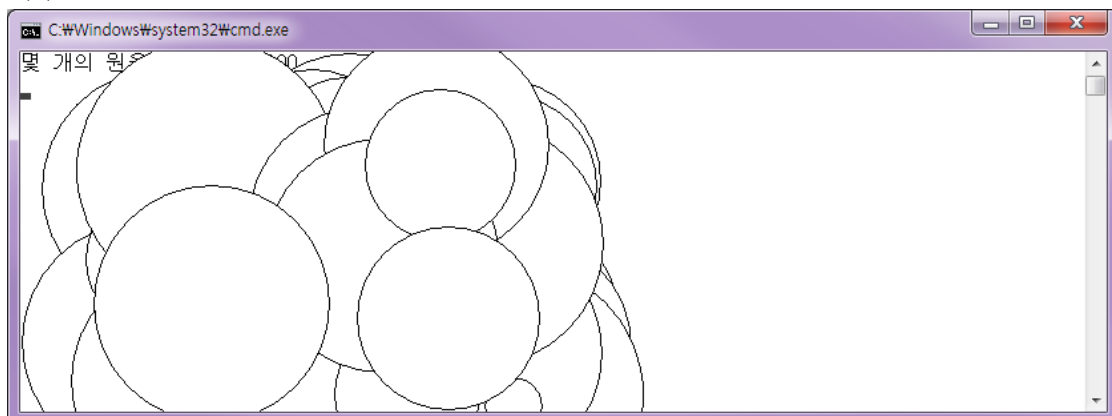
1

## 이번 장에서 만들어 볼 프로그램

(1) 정수 10개를 저장할 수 있는 동적 배열을 생성하고 난수를 저장한 후에 화면에 출력하는 프로그램을 작성해보자.



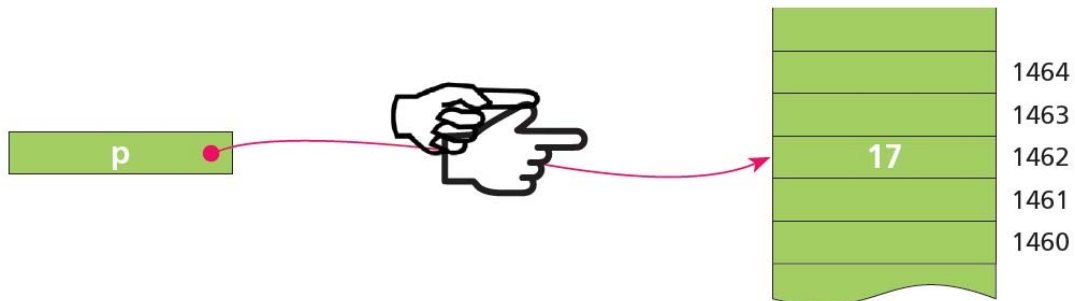
(2) 사용자가 입력한 개수만큼의 원을 동적으로 생성하여서 화면에 그리는 프로그램을 작성해보자.



2

# 포인터란?

- 주소를 가지고 있는 변수



3

## 포인터 선언

문법 8.1

포인터 선언

정수를 가리키는 포인터 정의

`int* p;`

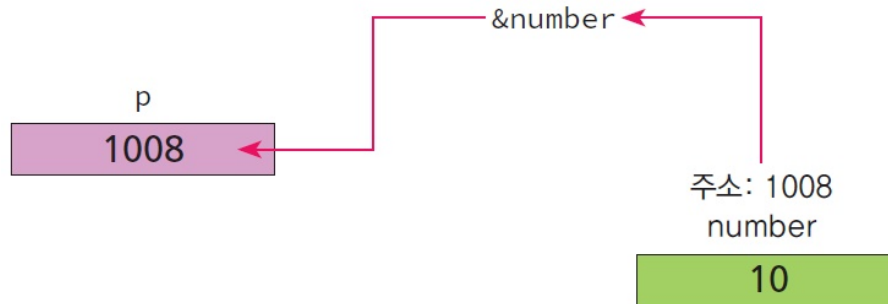
```
int *p1;
double *p2;
char *p3;
short *p4;
```

4

## 주소연산자 &

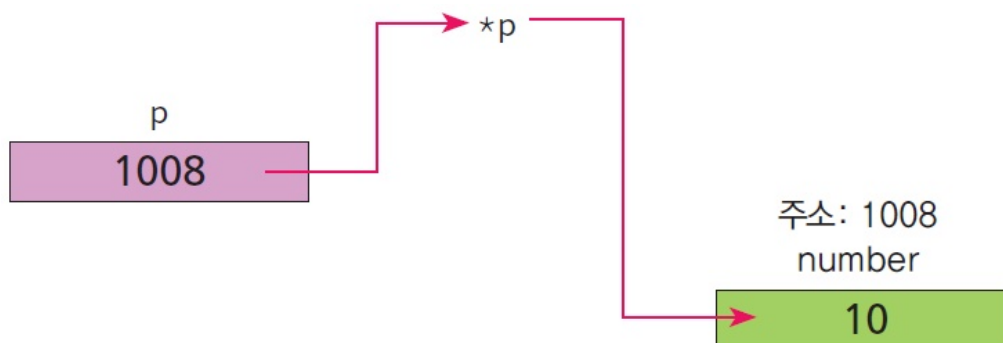
```
int number = 10;      // 변수 정의
int *p;               // 포인터 정의

p = &number;          // 변수 number의 주소를 포인터 p에 저장
```



5

## 간접참조연산자 \*



6

## 예제

```
#include <iostream>
using namespace std;
int main()
{
    int number = 10;
    // 변수 number의 주소를 계산하여 p에 저장한다.
    int *p = &number;

    // p가 가리키는 공간에 저장된 값을 출력한다.
    cout << *p << endl;
    return 0;
}
```

7

## nullptr

- ❑ `int *p = nullptr;(O)`
- ❑ `int *p = NULL;(X)`

8

## 동적 할당 메모리

- 동적 메모리 할당(dynamic memory allocation)이란 프로그램이 실행 도중에 동적으로 메모리를 할당받는 것을 말한다.



9

## 동적 메모리 사용절차

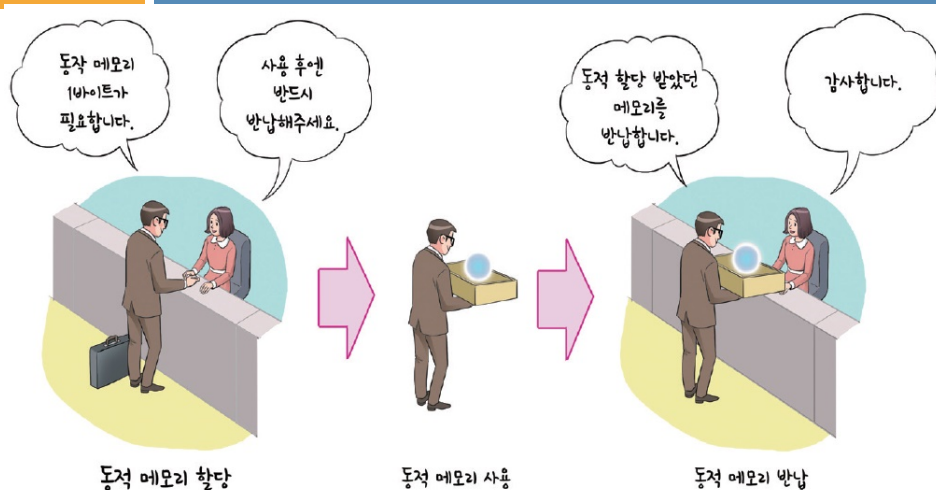


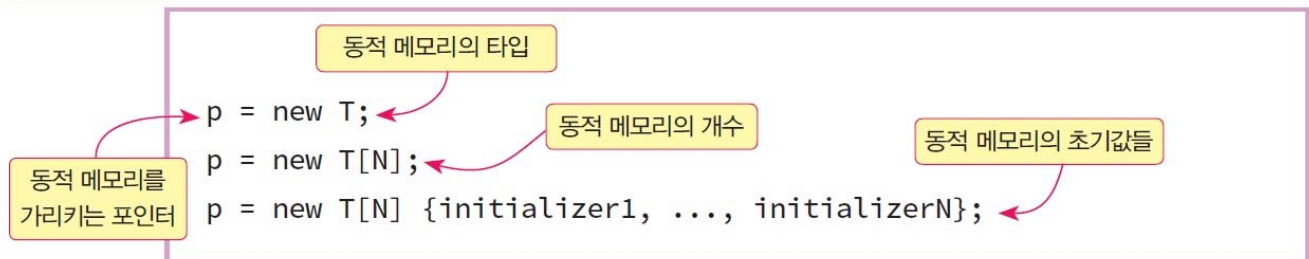
그림 8.1 동적 메모리 사용 단계

10

## 동적 메모리 할당

### 문법 8.3

#### new 연산자



11

## 동적 메모리 초기화

```
int *p;  
p = new int[5];
```

```
int *p = new int[5] { 0, 1, 2, 3, 4 };
```

12

# 동적 메모리 해제

## 문법 8.4

### delete 연산자

```
delete p;  
delete [] p;
```

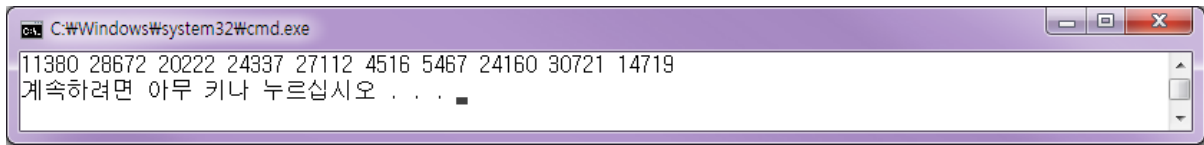
13

## 예제

```
#include <iostream>  
#include <time.h>  
using namespace std;  
int main()  
{  
    int *ptr;  
    srand(time(NULL)); // 난수 발생기 시드 설정  
    ptr = new int[10]; // ① 동적 메모리 할당  
    for (int i = 0; i < 10; i++)  
        ptr[i] = rand(); // ② 동적 메모리 사용  
    for (int i = 0; i < 10; i++)  
        cout << ptr[i] << " ";  
    delete[] ptr; // ③ 동적 메모리 반납  
    cout << endl;  
    return 0;  
}
```

14

## 실행결과



```
C:\Windows\system32\cmd.exe
11380 28672 20222 24337 27112 4516 5467 24160 30721 14719
계속하려면 아무 키나 누르십시오 . . .
```

15

## 스마트 포인터

- 동적 할당한 인스턴스를 자동으로 삭제해주는 포인터
- auto\_ptr
  - ▣ 동적 할당한 인스턴스를 자동으로 삭제
- shared\_ptr
  - ▣ 포인팅 횟수를 계수해서 0이 되면 대상을 삭제
- unique\_ptr
  - ▣ 한 대상을 오로지 한 포인터로만 포인팅
  - ▣ 하나의 소유자만 허용
- weak\_ptr
  - ▣ shared\_ptr에 접근
  - ▣ 참조는 못함: 참조하려면 lock() 메서드 사용

16



## auto\_ptr

---

- 배열을 지원하지 않음
  - ▣ AutoPtrSample.cpp
  - ▣ `auto_ptr<CMyData> ptrTest(new CMyData[3]);` -> error
- 대입 오류
  - ▣ AutoPtrSample2.cpp
  - ▣ `ptrNew = ptrTest;` // 원본 포인터가 null 됨

17

## shared\_ptr

---

- 포인팅 횟수를 계산해서 0이 되면 대상을 삭제
- SharePtrSample.cpp -> 출력 결과는?
- 배열로 객체를 삭제할 수 있는 방법을 제공
  - ▣ 배열로 대상을 삭제하는 함수를 등록
  - ▣ 실행 SharedPtrArray.cpp

```
void RemoveTest(CTest *pTest)
{
    delete [ ] pTest;
}

shared_ptr<CTest> ptr (new CTest[3], RemoveTest);
//ptr이 소멸할 때 자동으로 호출됨
```

18

## weak\_ptr

- 약하게 링크된 포인터를 래핑
  - ▣ 리소스를 가리키는 weak\_ptr 개체는 리소스의 참조 횟수에 영향을 주지 않는다
  - ▣ 해당 리소스를 관리하는 마지막 shared\_ptr 개체가 삭제되면 해당 리소스를 가리키는 weak\_ptr 개체가 있는 경우에도 리소스가 해제됨
  - ▣ 이는 데이터 구조에서 순환을 방지하는 데 필요
  - ▣ weak\_ptr& operator=(const weak\_ptr&);
  - ▣ void **swap**(weak\_ptr&);
  - ▣ void **reset**();
  - ▣ long **use\_count**() const;
  - ▣ bool **expired**() const;
  - ▣ shared\_ptr<Ty> **lock**() const;

19

## 스마트 포인터

- 스마트 포인터를 사용하면 프로그래머가 동적 메모리 할당 후에 잊어버려도 자동으로 동적 메모리가 삭제된다.
- 스마트 포인터는 자동으로 nullptr로 초기화된다.

```
#include <iostream>
#include <memory>
using namespace std;
int main()
{
    unique_ptr<int> p(new int);
    *p = 99;           // p를 사용한다.
}
```

20

## 스마트포인터로 배열 가리키기

```
#include <iostream>
#include <memory>
using namespace std;
int main()
{
    unique_ptr<int[]> buf(new int[10]);
    for (int i = 0; i<10; i++) {
        buf[i] = i;
    }
    for (int i = 0; i<10; i++) {
        cout << buf[i] << " ";
    }
    cout << endl;
    return 0;
}
```

21

## 8.5 객체의 동적 생성

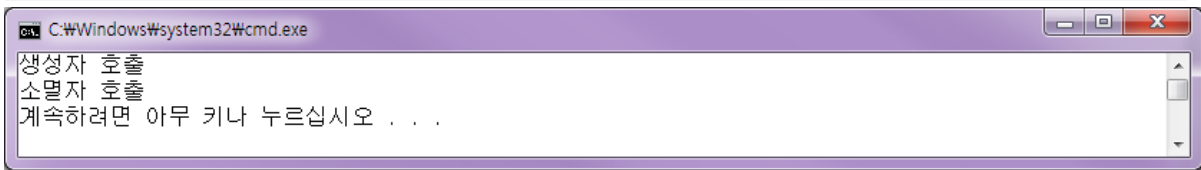
```
#include <iostream>
using namespace std;
class Dog {
private:
    string name;
    int age;
public:
    Dog() {
        cout << "생성자 호출\n";
        age = 1;
        name = "바둑이";
    }
    ~Dog() {
        cout << "소멸자 호출\n";
    }
};
```

22

## 객체의 동적 생성

```
int main()
{
    Dog * pDog = new Dog;
    delete pDog;

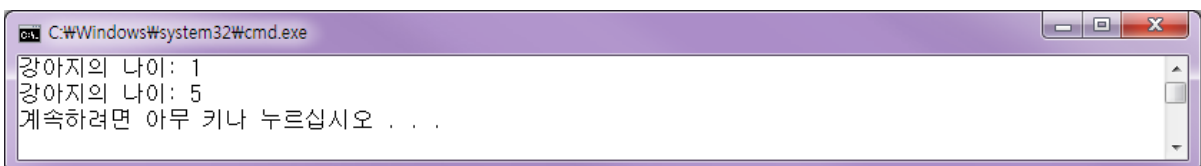
    return 0;
}
```



23

## 포인터를 통하여 멤버 접근하기

```
int main()
{
    Dog * pDog = new Dog;
    cout << "강아지의 나이: " << pDog->getAge() << endl;
    pDog->setAge(5);
    cout << "강아지의 나이: " << pDog->getAge() << endl;
    delete pDog;
    return 0;
}
```



24

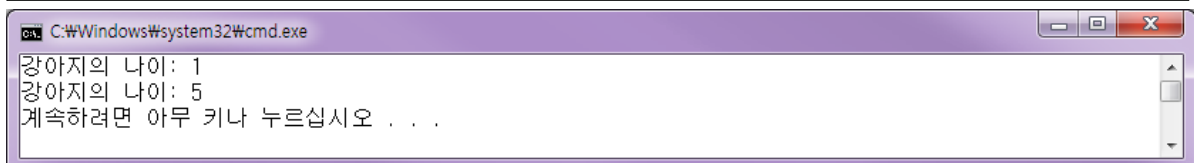
## 멤버 동적 생성

```
class Dog {
private:
    int *pWeight;
    int *pAge;
public:
    Dog() {
        pAge = new int(1);
        pWeight = new int(10);
    }
    ~Dog() {
        delete pAge;
        delete pWeight;
    }
    int getAge() { return *pAge; }
    void setAge(int age) { *pAge = age; }
    int getWeight() { return *pWeight; }
    void setWeight(int weight) { *pWeight = weight; }
};
```

25

## 멤버 동적 생성

```
int main()
{
    Dog * pDog = new Dog;
    cout << "강아지의 나이: " << pDog->getAge() << endl;
    pDog->setAge(5);
    cout << "강아지의 나이: " << pDog->getAge() << endl;
    delete pDog;
    return 0;
}
```



C:\Windows\system32\cmd.exe

```
강아지의 나이: 1
강아지의 나이: 5
계속하려면 아무 키나 누르십시오 . . .
```

26

Point라는 클래스가 다음과 같을 때 오류를 찾아 수정하라.  
아래 코드를 unique\_ptr을 사용하여 다시 작성하라.

```
#include <iostream>
using namespace std;
class Point {
    int x, y;
public:
    Point(int x, int y) : x(x), y(y) { }
    void setX(int x) { this->x = x; }
    void setY(int y) { this->y = y; }
    int  getX() { return x; }
    int  getY() { return y; }

int main()
{
    Point *p = new Point(100, 200);
    p.setX(30);
    p.setY(60);
    delete p;
    return 0;
}
```

27

## 실습

- 100개의 Point 객체를 저장할 수 있는 동적 객체 배열을 생성하고 Point 객체의 x, y 값을 난수로 채워라.

28

## this 포인터

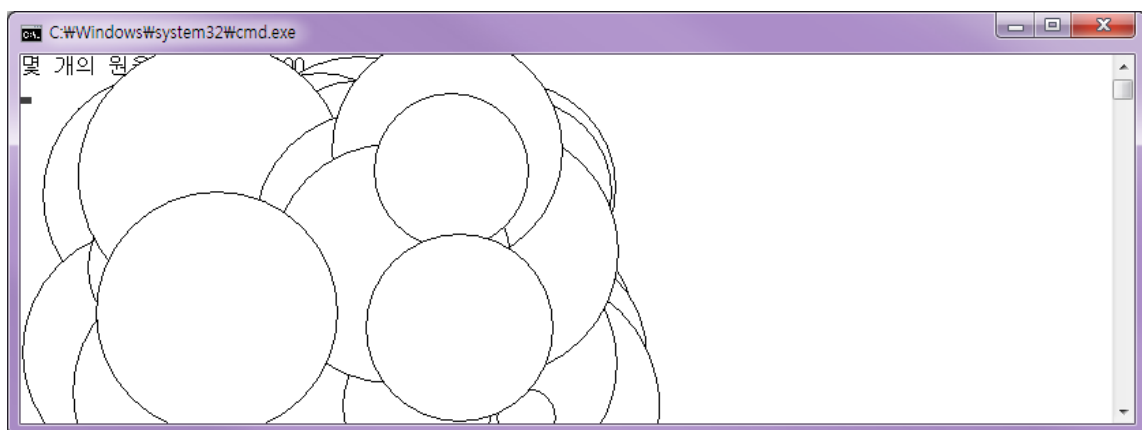
### □ 현재 객체를 참조하는 포인터

```
class Rectangle {  
private:  
    int length;  
    int width;  
public:  
    Rectangle() {  
        width = 30;  
        length = 40;  
    }  
    ~Rectangle() {}  
    void setLength(int length) { this->length = length; }  
    int getLength() { return this->length; }  
    void setWidth(int width) { this->width = width; }  
    int getWidth() { return width; }  
};
```

29

## Lab: 동적 원 생성

- 사용자가 입력한 개수만큼의 원을 동적으로 생성하여서 화면에 그리는 프로그램을 작성해 보자.



30

## this 포인터

```
#include <iostream>
#include <string>
#include <windows.h>
#include <conio.h>
using namespace std;
class Circle {
public:
    int x, y, radius;    // 원의 중심점과 반지름
    string color;        // 원의 색상
    void draw();
};
void Circle::draw()
{
    // 원을 화면에 그리는 함수
    HDC hdc = GetWindowDC(GetForegroundWindow());
    Ellipse(hdc, x - radius, y - radius, x + radius, y + radius);
}
```

31

## this 포인터

```
int main()
{
    int n;
    Circle *p;
    cout << "몇 개의 원을 만들까요: ";
    cin >> n;
    p = new Circle[n];
    for (int i = 0; i < n; i++) {
        p[i].x = 100 + rand() % 300;
        p[i].y = 100 + rand() % 200;
        p[i].radius = rand() % 100;
        p[i].draw();
    }
    delete[] p;
    getch();
    return 0;
}
```

32



- Circle 클래스를 이용하여 다음과 같은 동적 배열을 생성한다.
  - ▣ 동적 배열의 크기는 사용자가 입력한다.
  - ▣ 원의 반지름은 1부터 100 사이의 난수로 설정한다.
  - ▣ 동적 배열에 저장된 원의 면적이 100을 초과하는 원의 개수를 출력하는 프로그램을 작성하라.
  - ▣ 동적 배열은 사용이 끝나면 삭제하라.

## const 포인터

```
const int *p1;           // ①
int * const p2;          // ②
const int * const p3;    // ③
```

- ① p1은 변경되지 않는 정수를 가리키는 포인터이다. 이 포인터를 통하여 참조되는 값은 변경이 불가능하다.
- ② p2는 정수에 대한 상수 포인터이다. 정수는 변경될 수 있지만 p2는 다른 것을 가리킬 수 없다.
- ③ p3는 상수에 대한 상수 포인터이다. 포인터가 가리키는 값도 변경이 불가능하고 포인터 p3도 다른 것을 가리키게끔 변경될 수 없다.

## const 포인터와 const 멤버 함수

- 멤버 함수를 const로 정의하면 함수 안에서 멤버 변수를 변경하는 것이 금지된다. const 객체를 가리키는 포인터를 정의하면 이 포인터로 호출할 수 있는 함수는 const 함수뿐이다.

35

## 예제

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

class Circle
{
private:
    int radius;

public:
    Circle() :radius(10){}
    ~Circle() {}
    void setRadius(int radius) { this->radius = radius; }
    int getRadius() const { return radius; }
};
```

36

## 예제

```
int main()
{
    Circle* p = new Circle();
    const Circle *pConstObj = new Circle();
    Circle *const pConstPtr = new Circle();

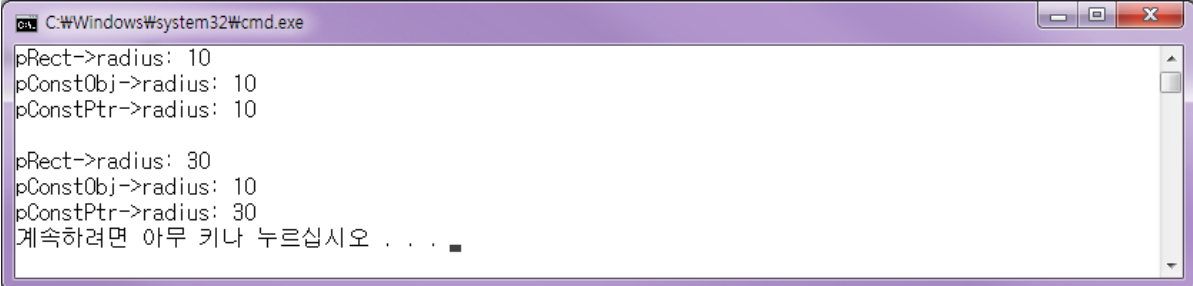
    cout << "pRect->radius: " << p->getRadius() << endl;
    cout << "pConstObj->radius: " << pConstObj->getRadius() << endl;
    cout << "pConstPtr->radius: " << pConstPtr->getRadius() << endl<<endl;

    p->setRadius(30);
    // pConstObj->setRadius(30);
    pConstPtr->setRadius(30);

    cout << "pRect->radius: " << p->getRadius() << endl;
    cout << "pConstObj->radius: " << pConstObj->getRadius() << endl;
    cout << "pConstPtr->radius: " << pConstPtr->getRadius() << endl;
    return 0;
}
```

37

## 실행결과



```
C:\Windows\system32\cmd.exe
pRect->radius: 10
pConstObj->radius: 10
pConstPtr->radius: 10

pRect->radius: 30
pConstObj->radius: 10
pConstPtr->radius: 30
계속하려면 아무 키나 누르십시오 . . .
```

38

# Q & A

---

