04

객체 포인터와 객체 배열, 객체의 동적 생성

학습 목표

1. 객체에 대한 포인터를 선언하고 활용할 수 있다.
2. 객체의 배열을 선언하고 활용할 수 있다.
3. new를 이용하여 동적으로 메모리나 배열을 할당 받고 delete를 이용하여 반환할 수 있다.
4. new를 이용하여 동적으로 객체나 객체 배열을 할당 받고 delete를 이용하여 반환할 수 있다.
5. this 포인터의 개념을 이해하고, 활용할 수 있다.
6. string 클래스를 이용하여 문자열을 다룰 수 있다.

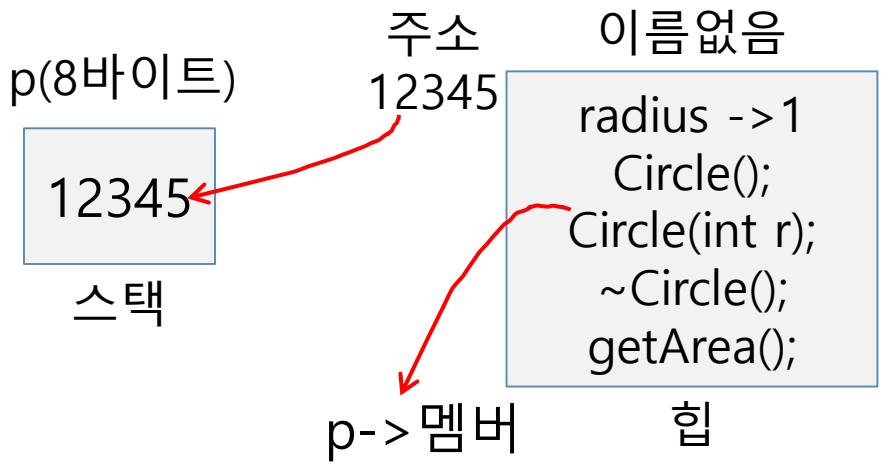
객체의 동적 생성 및 반환

- 객체의 동적 생성 및 반환 -> 기본자료형과 동일

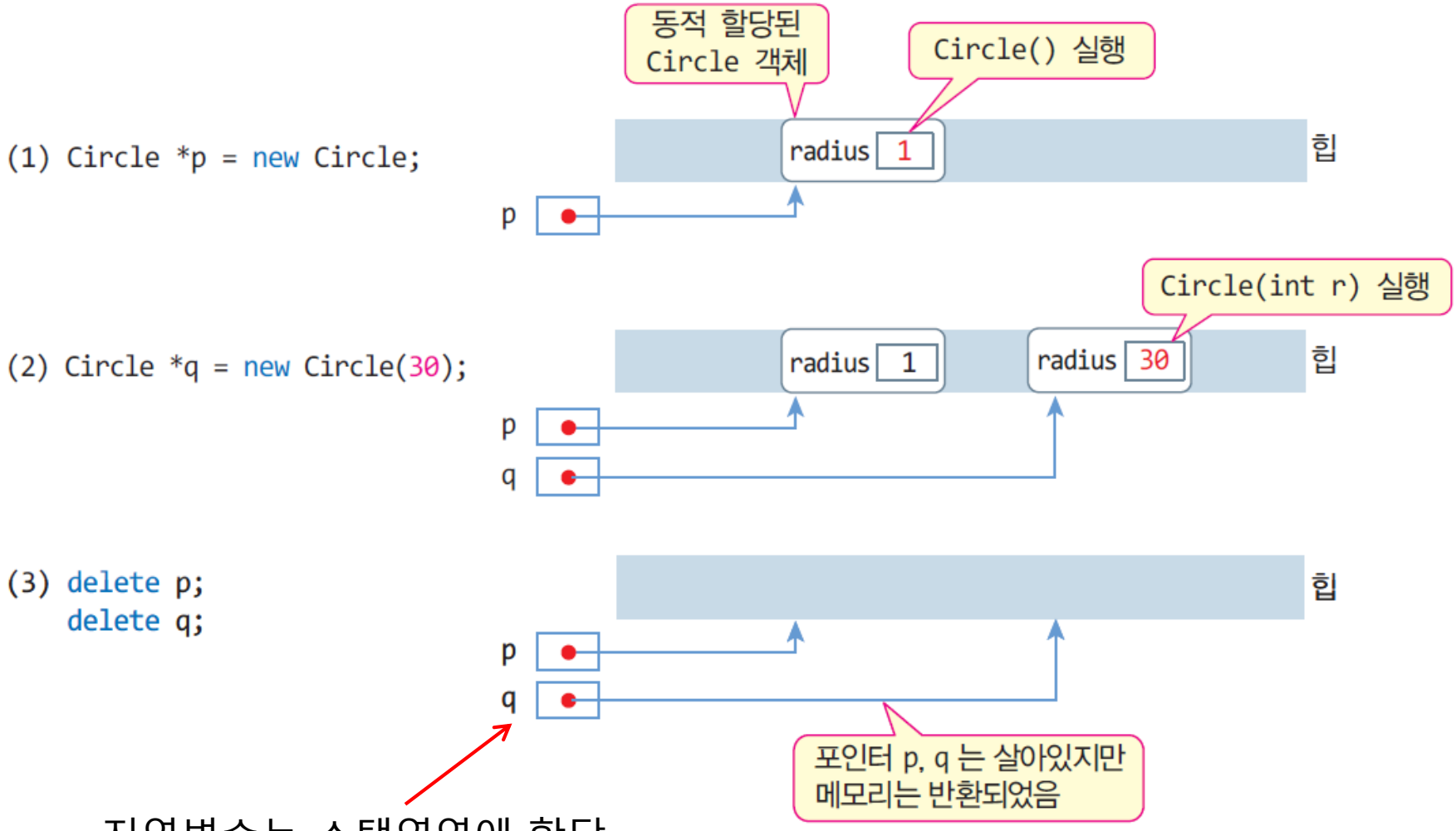
```
클래스명* 객체포인터 = new 클래스명;
클래스명* 객체포인터 = new 클래스명(생성자의 인자);
delete 객체포인터;
```

  - new 연산자 실행시 객체를 힙영역에 동적 할당 후 생성자 호출
  - 생성자에게 전달할 인자는 클래스명 뒤에 소괄호안에 작성
  - delete 연산자 실행시 소멸자 호출 후 메모리 해제(반환)

```
Circle* p = new Circle;
Circle* q = new Circle(30);
p->getArea();
q->getArea();
....
delete p;
delete q;
```



객체의 동적 생성 및 반환


동적 할당된
Circle 객체
Circle() 실행
(1) Circle *p = new Circle;
radius 1
힙
p
Circle(int r) 실행
(2) Circle *q = new Circle(30);
radius 1
radius 30
힙
p
q
(3) delete p;
delete q;
힙
p
q
포인터 p, q 는 살아있지만
메모리는 반환되었음
지역변수는 스택영역에 할당
객체가 반환직전에 소멸자가 호출

예제 4-7 Circle 객체의 동적 생성 및 반환


```
#include <iostream>
using namespace std;
class Circle {
	int radius;
public:
	Circle();
	Circle(int r);
	~Circle();
	void setRadius(int r) { radius = r; }
	double getArea() { return 3.14 * radius * radius; }
};
Circle::Circle() {
	radius = 1;
	cout << "생성자 실행 radius = " << radius << endl;
}
Circle::Circle(int r) {
	radius = r;
	cout << "생성자 실행 radius = " << radius << endl;
}
```

예제 4-7 Circle 객체의 동적 생성 및 반환


```
Circle::~Circle() {
	cout << "소멸자 실행 radius = " << radius
	<< endl;
}
int main() {
	Circle* p = new Circle;
	Circle* q = new Circle(30);
	cout << p->getArea() << endl;
	cout << q->getArea() << endl;

	delete p;
	delete q;
}
```

```
생성자 실행 radius = 1
생성자 실행 radius = 30
3.14
2826
소멸자 실행 radius = 1
소멸자 실행 radius = 30
```

예제 4–8 Circle 객체의 동적 생성과 반환 응용

- 정수 반지름을 입력 받고 Circle 객체를 동적 생성하여 면적을 출력하라. 음수가 입력되면 프로그램은 종료한다.

```
#include <iostream>
using namespace std;
class Circle {
	int radius;
public:
	Circle();
	Circle(int r);
	~Circle();
	void setRadius(int r) { radius = r; }
	double getArea() { return 3.14 * radius * radius; }
};
Circle::Circle() {
	radius = 1; cout << "생성자 실행 radius = " << radius << endl;
}
```

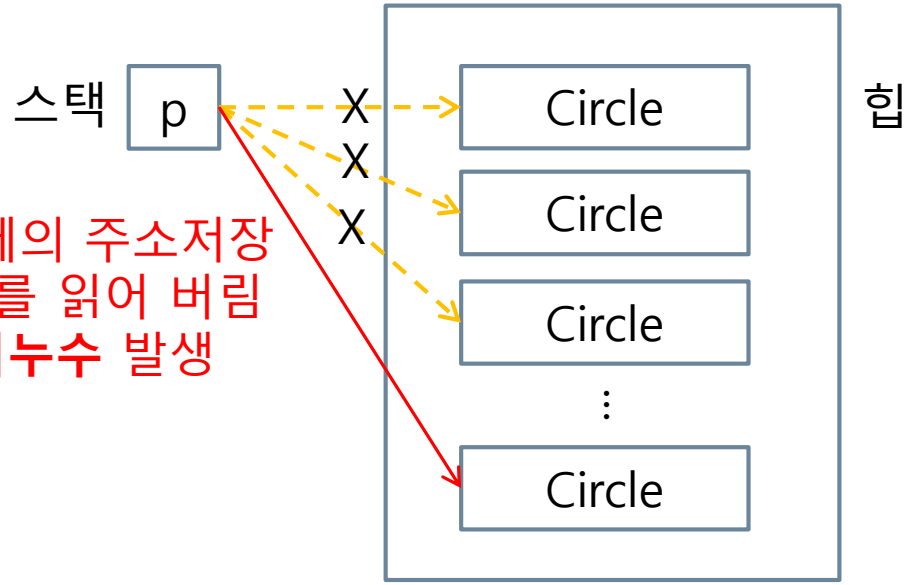
```
정수 반지름 입력(음수이면 종료)>> 5<엔터>
생성자 실행 radius = 5
원의 면적은 78.5
소멸자 실행 radius = 5
정수 반지름 입력(음수이면 종료)>> 9<엔터>
생성자 실행 radius = 9
원의 면적은 254.34
소멸자 실행 radius = 9
정수 반지름 입력(음수이면 종료)>> -1<엔터>
```

예제 4–8 Circle 객체의 동적 생성과 반환 응용


```
Circle::Circle(int r) {
	radius = r; cout << "생성자 실행 radius = " << radius << endl;
}
Circle::~Circle() {
	cout << "소멸자 실행 radius = " << radius << endl;
}
int main() {
	int radius;
	while (true) {
		cout << "정수 반지름 입력(음수이면 종료)>> ";
		cin >> radius;
		if (radius < 0) break;                    // 음수가 입력되면 종료.
		Circle* p = new Circle(radius);           // 동적 객체 생성
		cout << "원의 면적은 " << p->getArea() << endl;
		delete p;   // 객체반환, delete 문이 없다면 메모리 누수발생
	}
}
```

예제 4–8 Circle 객체의 동적 생성과 반환 응용


```
while (true) {
	cout << "정수 반지름 입력(음수이면 종료)>> ";
	cin >> radius;
	if (radius < 0) break;
	Circle* p = new Circle(radius);
	cout << "원의 면적은 " << p->getArea() << endl;
}
```



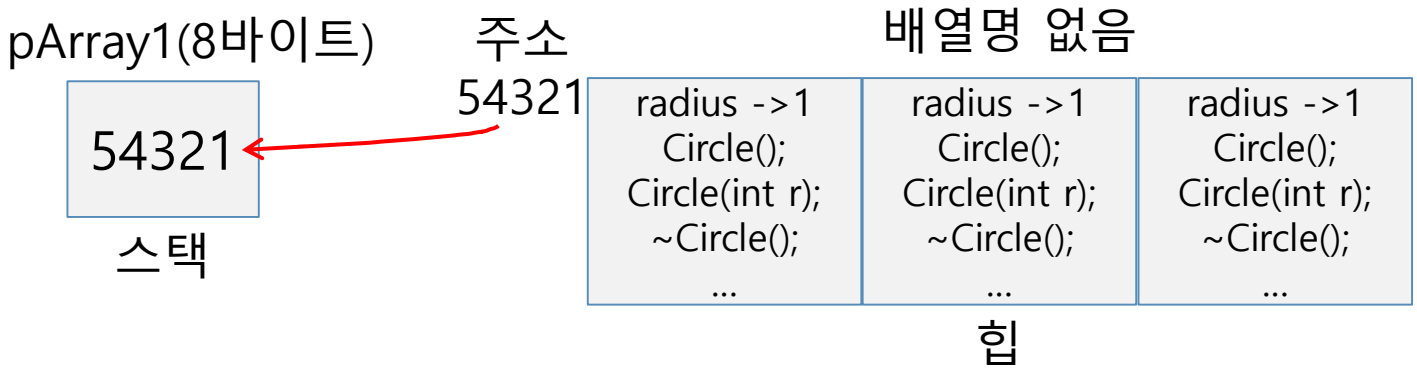
p는 마지막으로 생성된 객체의 주소저장
이전에 생성한 객체의 주소를 잃어 버림
-> delete 불가능 -> **메모리누수** 발생

객체 배열의 동적 생성 및 반환

- 객체 배열의 동적 생성 및 반환 -> 기본자료형 배열과 동일

```
클래스명* 객체포인터 = new 클래스명[배열크기];
클래스명* 객체포인터 = new 클래스명[배열크기]{생성자리스트};
delete [ ] 포인터;
```

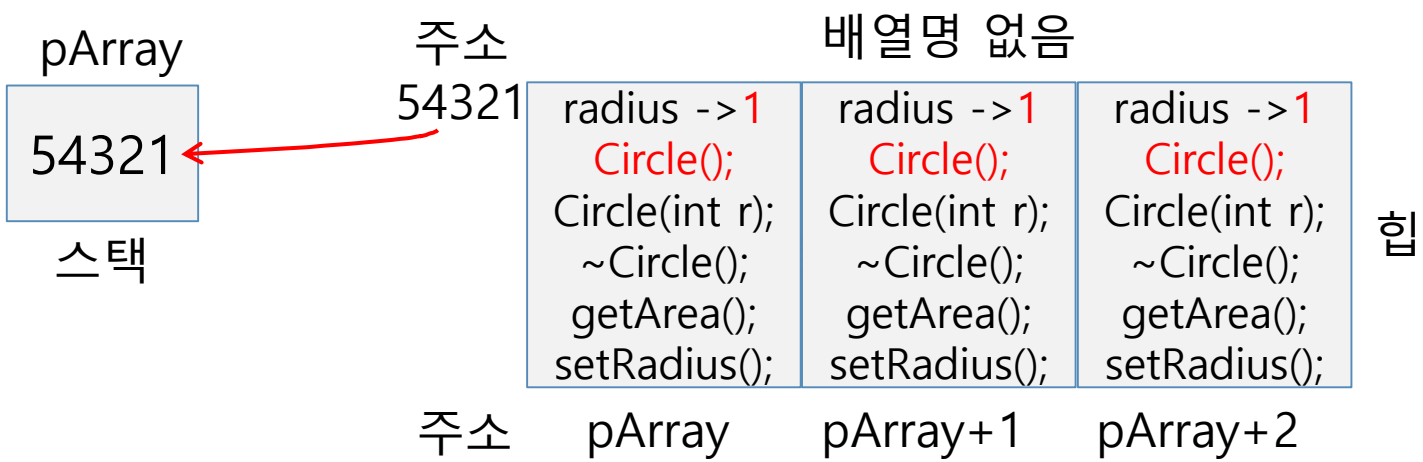
```
Circle* pArray1 = new Circle[3];
Circle* pArray2 = new Circle[3]{ Circle(1), Circle(2), Circle(3) };
delete [ ] pArray1;
delete [ ] pArray2;
```



객체 배열의 사용, 배열의 반환과 소멸자

- 객체배열 동적생성 후 매개변수 없는 생성자 호출

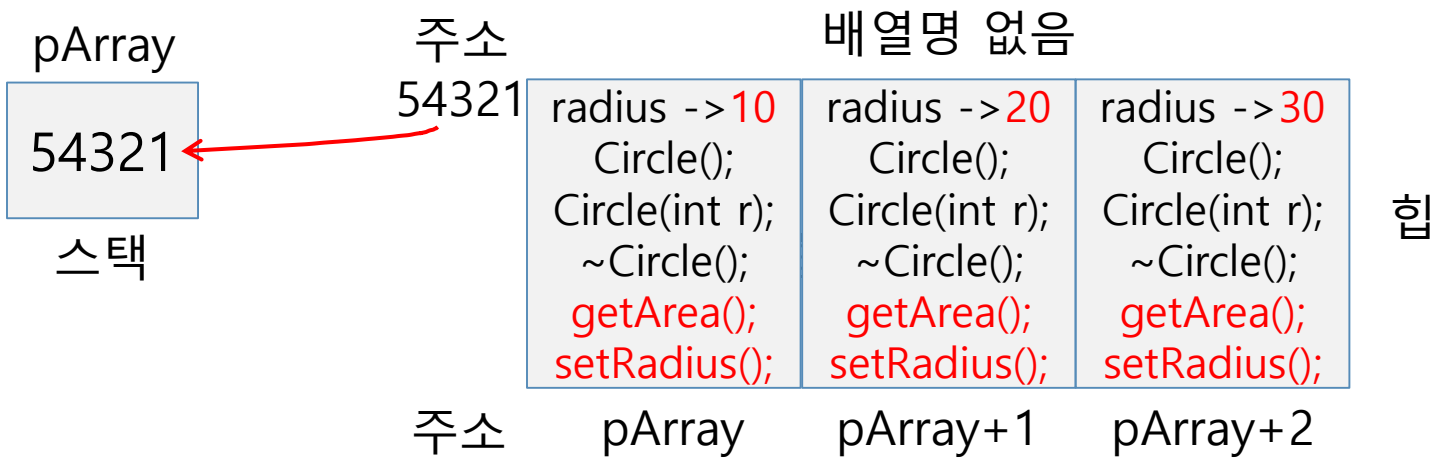
```
Circle* pArray = new Circle[3];
//pArray->Circle() -> (pArray+1)->Circle() -> (pArray+2)->Circle()
```



객체 배열의 사용, 배열의 반환과 소멸자

- 객체 포인터를 이용하여 멤버함수 호출

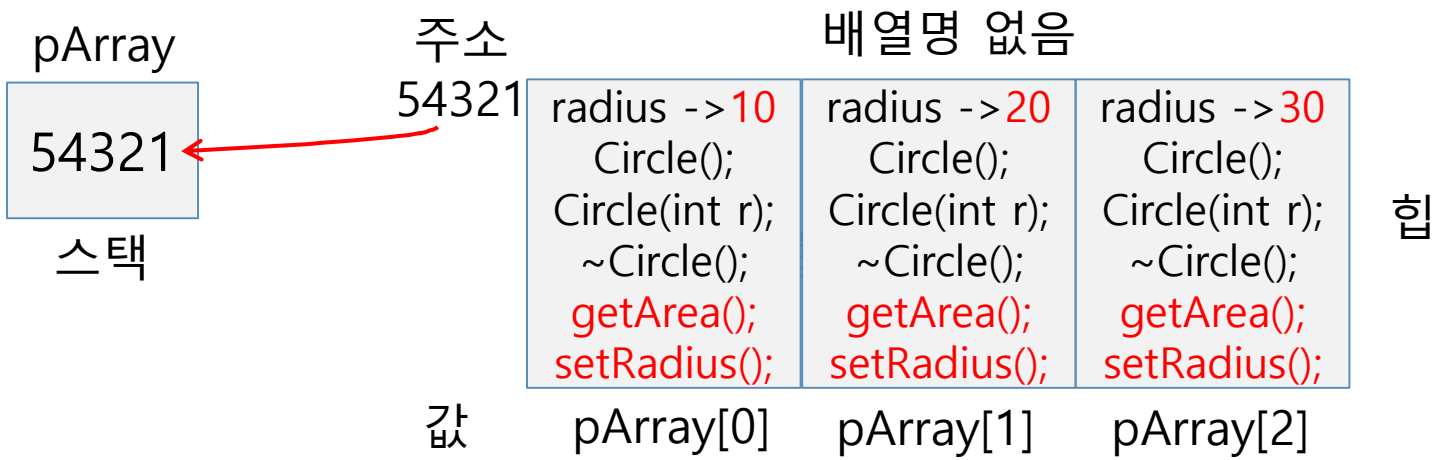
```
Circle* pArray = new Circle[3];
pArray->setRadius(10);
(pArray + 1)->setRadius(20);
(pArray + 2)->setRadius(30);
for (int i = 0; i < 3; i++) {
      cout << (pArray + i)->getArea();
}
```



객체 배열의 사용, 배열의 반환과 소멸자

- 배열 표현을 이용하여 멤버함수 호출

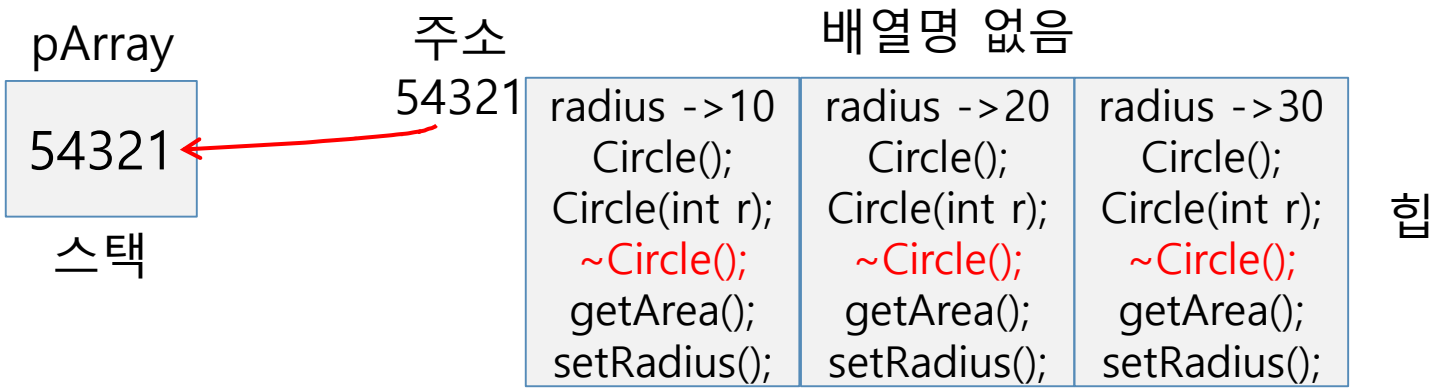
```
Circle* pArray = new Circle[3];
pArray[0].setRadius(10);
pArray[1].setRadius(20);
pArray[2].setRadius(30);
for (int i = 0; i < 3; i++) {
      cout << pArray[i].getArea();
}
```



객체 배열의 사용, 배열의 반환과 소멸자

- delete연산 수행시 소멸자 호출 후 반환됨

```
delete [ ] pArray;
//pArray[2].~Circle() -> pArray[1].~Circle() -> pArray[0].~Circle()
```



예제 4-9 Circle 배열의 동적 생성 및 반환

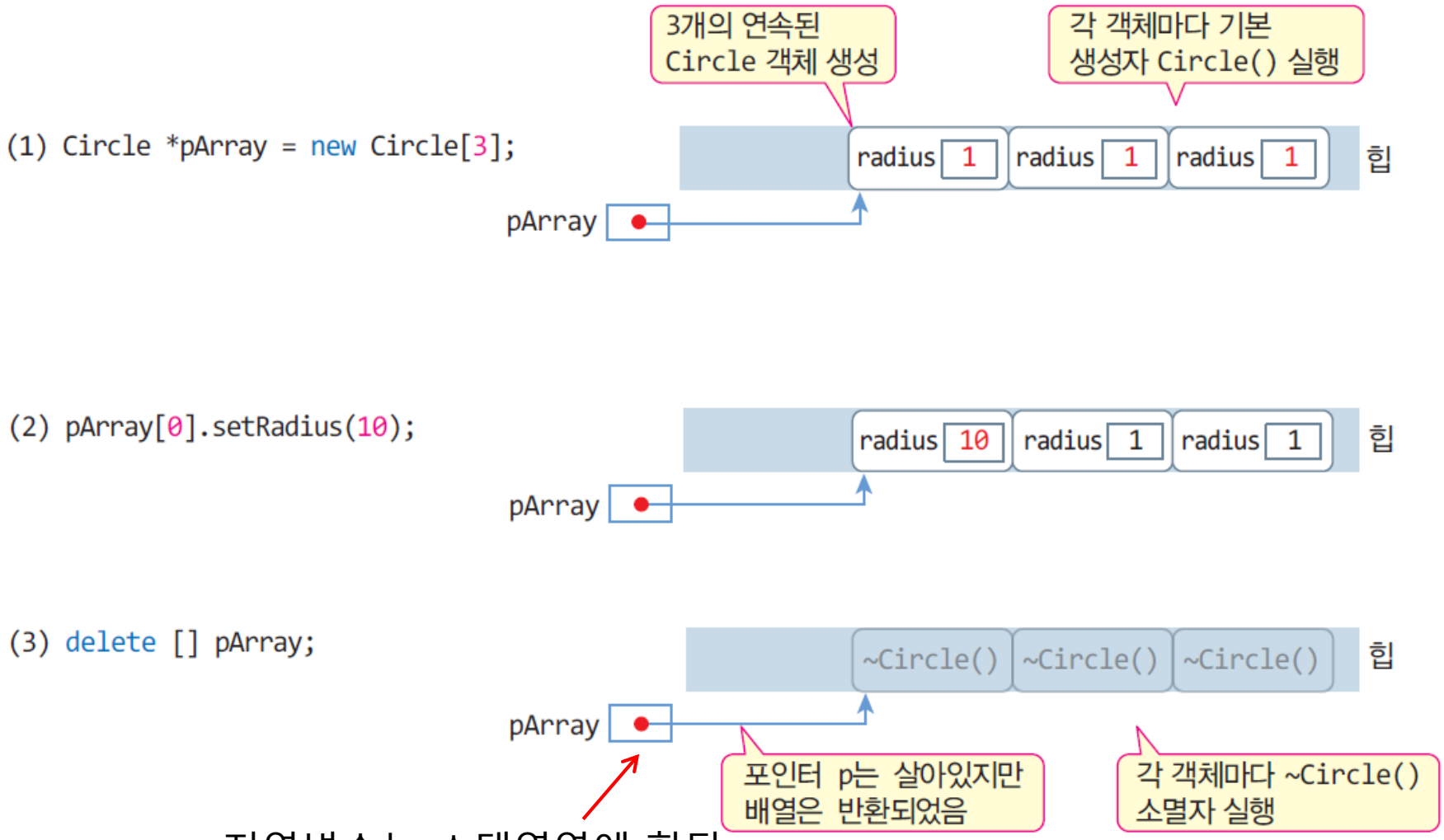

```
#include <iostream>
using namespace std;
class Circle {
	int radius;
public:
	Circle();
	Circle(int r);
	~Circle();
	void setRadius(int r) { radius = r; }
	double getArea() { return 3.14 * radius * radius; }
};
Circle::Circle() {
	radius = 1; cout << "생성자 실행 radius = " << radius << endl;
}
Circle::Circle(int r) {
	radius = r; cout << "생성자 실행 radius = " << radius << endl;
}
Circle::~Circle() {
	cout << "소멸자 실행 radius = " << radius << endl;
}
```

예제 4-9 Circle 배열의 동적 생성 및 반환


```
int main() {
	Circle* pArray = new Circle[3];
	pArray[0].setRadius(10);
	pArray[1].setRadius(20);
	pArray[2].setRadius(30);

	for (int i = 0; i < 3; i++)
		cout << pArray[i].getArea() << '₩n';

	Circle* p = pArray;
	for (int i = 0; i < 3; i++) {
		cout << p->getArea() << '₩n';
		p++;
	}

	delete [] pArray;
}
```

```
생성자 실행 radius = 1
생성자 실행 radius = 1
생성자 실행 radius = 1
314
1256
2826
314
1256
2826
소멸자 실행 radius = 30
소멸자 실행 radius = 20
소멸자 실행 radius = 10
```

객체 배열의 동적 생성 및 반환



예제 4-10 객체 배열의 동적 생성과 반환 응용

- 원을 개수를 입력 받고 Circle 배열을 동적 생성하고 반지름 값을 입력 받아 면적을 계산하고 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;
}
```

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

예제 4-10 객체 배열의 동적 생성과 반환 응용


```
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 << ": "; // 도움말 출력
		cin >> radius; // 반지름 입력
		pArray[i].setRadius(radius);
	}
```

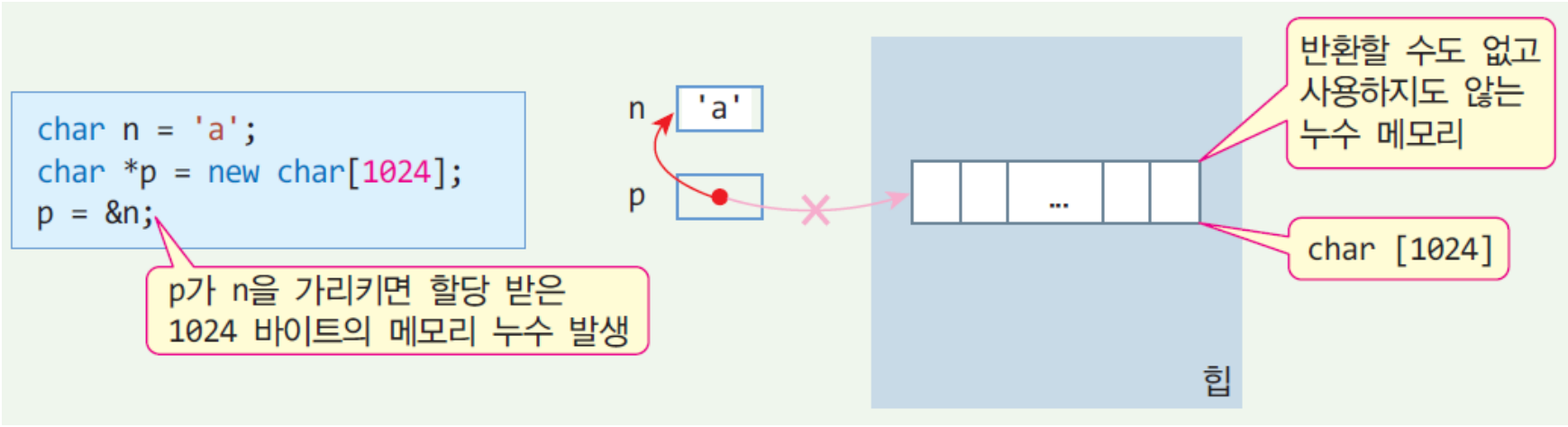
예제 4-10 객체 배열의 동적 생성과 반환 응용

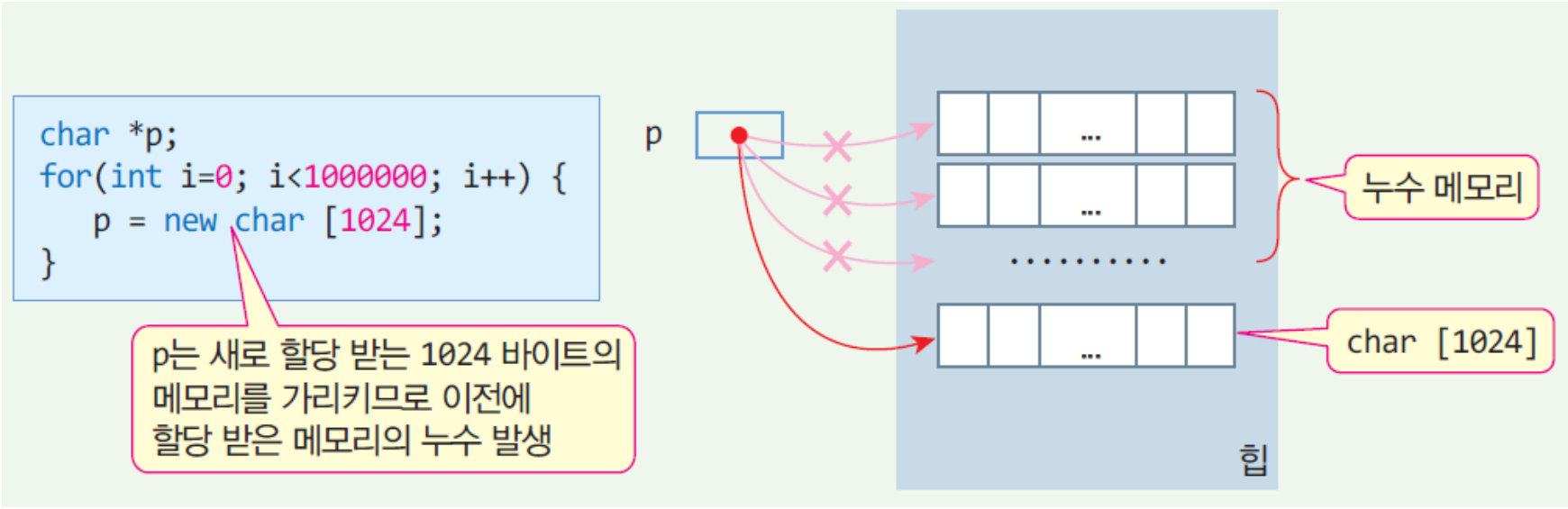

```
    int count = 0;
    Circle* p = pArray;
    for (int i = 0; i < n; i++) {
        cout << p->getArea() << ' ';
        if (p->getArea() >= 100 && p->getArea() <= 200)
            count++;
        p++;
    }
    cout << endl << "면적이 100에서 200 사이인 원의 개수는 "
    << count << endl;

    delete [] pArray;
}
```

동적 메모리 할당과 메모리 누수

- 힙은 응용프로그램들이 실행 중에 할당 받아 사용하는 공유 메모리
- 하나의 프로그램이 많은 메모리를 할당 받으면 힙 메모리가 부족하여 다른 프로그램이 할당 못 받는 경우 발생
- 동적 할당 받은 메모리가 필요 없다면 반드시 반환해야 하고 반환하지(delete) 않으면 다른 프로그램이 사용하지 못함
- 프로그램이 종료될 때 모든 동적할당 메모리는 자동으로 반환됨
- 동적으로 할당 받은 메모리의 주소를 잃어버려 힙에 반환 할 수 없게 되면 **메모리 누수(memory leak)**가 발생

동적 메모리 할당과 메모리 누수





실습과제1

- 메모리 누수를 설명하라.
- 메모리 누수가 왜 심각한 문제를 발생시키는지 설명하라.
- 메모리 누수를 방지하는 방법을 설명하라.

실습과제2

- 아래 코드의 문제점을 설명하고 해결방법을 설명하시오.

```
// 예제4-8의 Circle 클래스 정의 추가
int main() {
	int radius;
	while (true) {
		cout << "반지름 입력(음수이면 종료)>> ";
		cin >> radius;
		if (radius < 0) break;                    .
		Circle* p = new Circle(radius);
		cout << "원의 면적:" << p->getArea() << endl;
	}
}
```

실습과제3

- 아래 코드의 문제점을 설명하고 해결방법을 설명하시오.

```
// 예제4-9의 Circle 클래스 정의 추가
int main() {
	Circle* pArray = new Circle[3];
	for (int i = 0; i < 3; i++) {
		cout << pArray ->getArea() << '₩n';
		pArray++;
	}

	delete [ ] pArray;
}
```

실습과제4

- 지난과제에서 사용한 삼각형 클래스 Triangle을 정의하고 객체 배열 3개를 동적으로 생성하여 아래 결과처럼 동작하는 프로그램을 작성하시오. 객체 배열을 동적 할당 시 초기화 해야 한다.(예제 4-9과 유사함)
- 생성자, 소멸자, getArea 멤버함수가 필요함

```
밑변 1 높이 1 인 삼각형 생성
밑변 2 높이 2 인 삼각형 생성
밑변 4 높이 4 인 삼각형 생성
삼각형의 면적은 0.5
삼각형의 면적은 2
삼각형의 면적은 8
밑변 4 높이 4 인 삼각형 소멸
밑변 2 높이 2 인 삼각형 소멸
밑변 1 높이 1 인 삼각형 소멸
```

실습과제5

- 지난과제에서 사용한 구 클래스 Sphere을 정의하고 먼저 생성할 구의 개수를 입력 받은 후 객체 배열을 동적으로 생성하여 아래 결과처럼 동작하는 프로그램을 작성하시오. (예제 4-10과 유사함)
- setRadius, getVolume 멤버함수가 필요함

```
생성하고자 하는 구의 개수: 3<엔터>
구1의 반지름: 1 <엔터>
구2의 반지름: 2 <엔터>
구3의 반지름: 3 <엔터>
구1의 부피 4.18
구2의 부피 34.49
구3의 부피 113.04
```

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ************************************************
//    제   목 :  정수 4개의 평균을 구하는 프로그램
//    날   짜 :  2023년 9월10일
//    작성자 :  15010101 홍길동
// ************************************************

// 소스코드 작성
```