



4장. 객체 포인터와 객체 배열, 객체의 동적 생성

다룰 내용

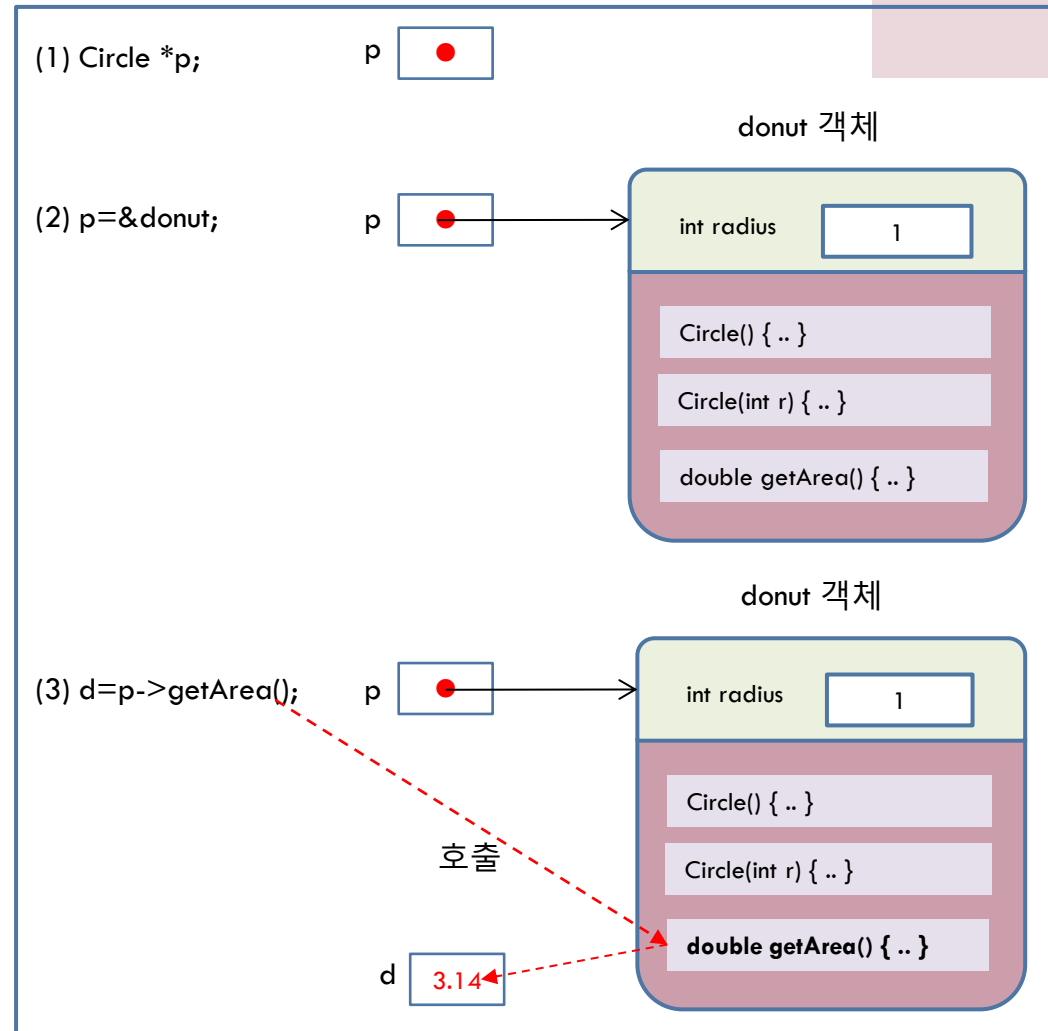
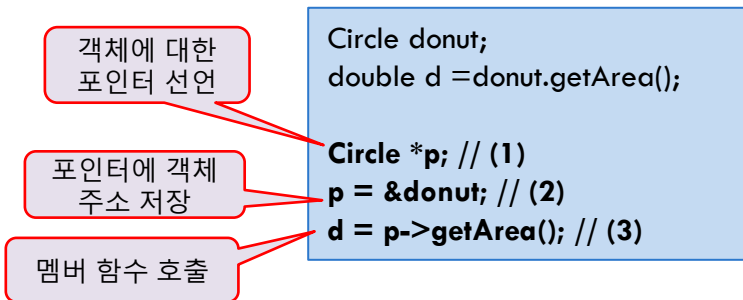
- 1) 객체포인터
- 2) 객체배열
- 3) 동적메모리 할당 및 반환
- 4) 객체와 객체배열의 동적 생성 및 반환
- 5) This 포인터
- 6) String 클래스

객체 포인터를 사용해야 되는 경우

- 동적 메모리 할당
 - ▶ 동적으로 할당된 객체에 접근할 때
- 함수에서 객체 배열을 인수로 사용
 - ▶ 함수의 인수로 객체배열을 넘길 때

객체 포인터

- 객체에 대한 포인터
 - ▶ C 언어의 포인터와 동일
 - ▶ 객체의 주소 값을 가지는 변수
- 포인터로 멤버를 접근할 때
 - ▶ 객체포인터->멤버



예제 4-1 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
```



객체 배열

객체 배열, 생성 및 소멸

- 객체 배열 선언 가능
 - ▶ 기본 자료형 배열 선언과 형식 동일
 - ▶ `int n[3];` // 정수형 배열 선언
 - ▶ `Circle c[3];` // Circle 타입의 배열 선언
- 객체 배열 선언
 1. 객체 배열을 위한 공간 할당
 2. 배열의 각 원소 객체마다 생성자 실행
 - ▶ `c[0]`의 생성자, `c[1]`의 생성자, `c[2]`의 생성자 실행
 - ▶ **매개 변수 없는 생성자 호출**
 - ▶ 매개 변수 있는 생성자를 호출할 수 없음
 - ▶ `Circle circleArray[3](5);` // 오류
- 배열 소멸
 - ▶ 배열의 각 객체마다 소멸자 호출. 생성의 반대순으로 소멸
 - ▶ `c[2]`의 소멸자, `c[1]`의 소멸자, `c[0]`의 소멸자 실행

예제 4- 2 Circle 클래스의 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;
}
```

```
int main() {
    Circle circleArray[3]; // (1) Circle 객체 배열 생성

    // 배열의 각 원소 객체의 멤버 접근
    circleArray[0].setRadius(10); // (2)
    circleArray[1].setRadius(20);
    circleArray[2].setRadius(30);

    for(int i=0; i<3; i++) // 배열의 각 원소 객체의 멤버 접근
        cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;

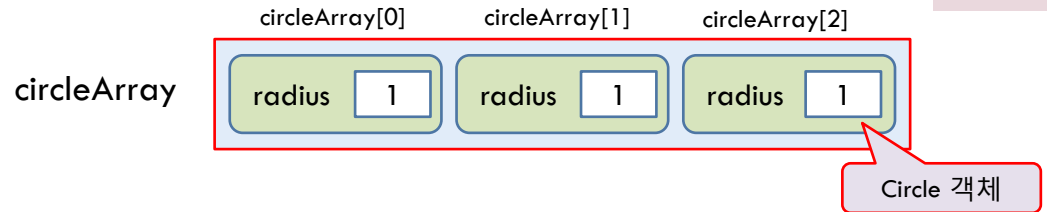
    Circle *p; // (3)
    p = circleArray; // (4)
    for(int i=0; i<3; i++) { // 객체 포인터로 배열 접근
        cout << "Circle " << i << "의 면적은 " << p->getArea() << endl;
        p++; // (5)
    }
}
```

```
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826
```

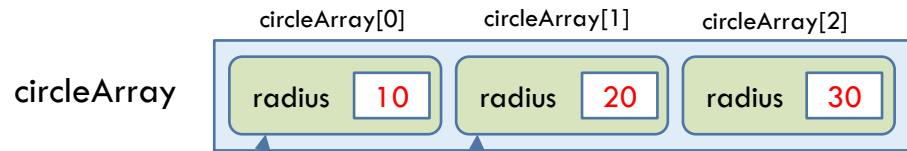

배열 생성과 활용(예제 4-2의 실행 과정)

기본 생성자
Circle() 호출

(1) `Circle circleArray[3];`



(2)
`circleArray[0].setRadius(10);`
`circleArray[1].setRadius(20);`
`circleArray[2].setRadius(30);`



(3) `Circle *p;`



(4) `p = circleArray;`



(5) `p++;`



객체 배열 생성시 기본 생성자를 호출함

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

class Circle {
    int radius;
public:
    double getArea() {
        return 3.14*radius*radius;
    }
};

int main() {
    Circle circleArray[3];
}
```

컴파일러가 자동으로 기본 생성자
Circle() {} 삽입.
컴파일 오류가 발생하지 않음

기본 생성자 Circle() 호출

(a) 생성자가 선언되어
있지 않은 Circle 클래스

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

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

int main() {
    Circle waffle(15);
    Circle circleArray[3];
}
```

Circle(int r)
호출

기본 생성자 Circle() 호출.
기본 생성자가 없으므로 컴
파일 오류

error.cpp(15): error C2512: 'Circle': 사용할 수
있는 적절한 기본 생성자가 없습니다

(b) 기본 생성자가 없으므로 컴파일 오류
C++ 프로그래밍1

객체 배열 초기화

- 객체 배열 초기화 방법
 - ▶ 배열의 각 원소 객체당 생성자 지정하는 방법

```
Circle circleArray[3] = { Circle(10), Circle(20), Circle() };
```

- ▶ circleArray[0] 객체가 생성될 때, 생성자 Circle(10) 호출
- ▶ circleArray[1] 객체가 생성될 때, 생성자 Circle(20) 호출
- ▶ circleArray[2] 객체가 생성될 때, 생성자 Circle() 호출

예제 4-3 객체 배열 초기화

```
#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;
}

int main() {
    Circle circleArray[3] = { Circle(10), Circle(20), Circle() }; // Circle 배열 초기화

    for(int i=0; i<3; i++)
        cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;
}
```

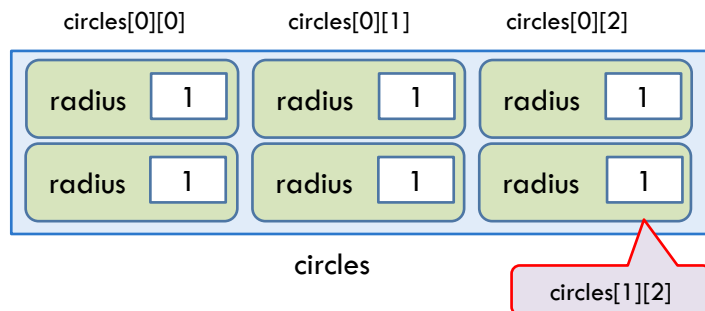
circleArray[0] 객체가 생성될 때, 생성자 Circle(10),
circleArray[1] 객체가 생성될 때, 생성자 Circle(20),
circleArray[2] 객체가 생성될 때, 기본 생성자 Circle()이
호출된다.

Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 3.14

2차원 배열

Circle circles[2][3];

Circle() 호출

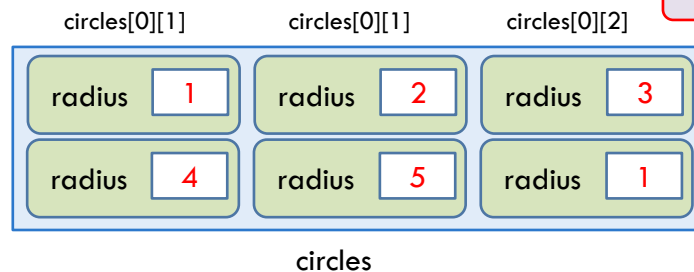


(a) 2차원 배열 선언

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

Circle(int r) 호출

Circle() 호출



(b) 2차원 배열 선언과 초기화

```
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);
```

2차원 배열을 초기화하는 다른 방식

예제 4-4 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;
}
```

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

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



동적 메모리 할당

메모리 정적 할당과 동적 할당

- 정적 할당
 - ▶ 변수 선언을 통해 필요한 메모리 할당
 - ▶ 많은 양의 메모리는 배열 선언을 통해 할당
- 동적 할당
 - ▶ 필요한 양이 예측되지 않는 경우. 프로그램 작성시 할당 받을 수 없음
 - ▶ 실행 중에 운영체제로부터 할당 받음
 - ▶ 힙(heap)으로부터 할당
 - ▶ 힙은 운영체제가 소유하고 관리하는 메모리로 모든 프로세스가 공유할 수 있는 메모리

동적 메모리 할당 및 반환

- C 언어의 동적 메모리 할당 : malloc()/free() 라이브러리 함수 사용
- C++의 동적 메모리 할당/반환
 - ▶ new 연산자
 - ▶ 기본 타입 메모리 할당, 배열 할당, 객체 할당, 객체 배열 할당
 - ▶ 객체의 동적 생성 - 힙 메모리로부터 객체를 위한 메모리 할당 요청
 - ▶ 객체 할당 시 생성자 호출
 - ▶ delete 연산자
 - ▶ new로 할당 받은 메모리 반환
 - ▶ 객체의 동적 소멸 - 소멸자 호출 뒤 객체를 힙에 반환

new와 delete 연산자

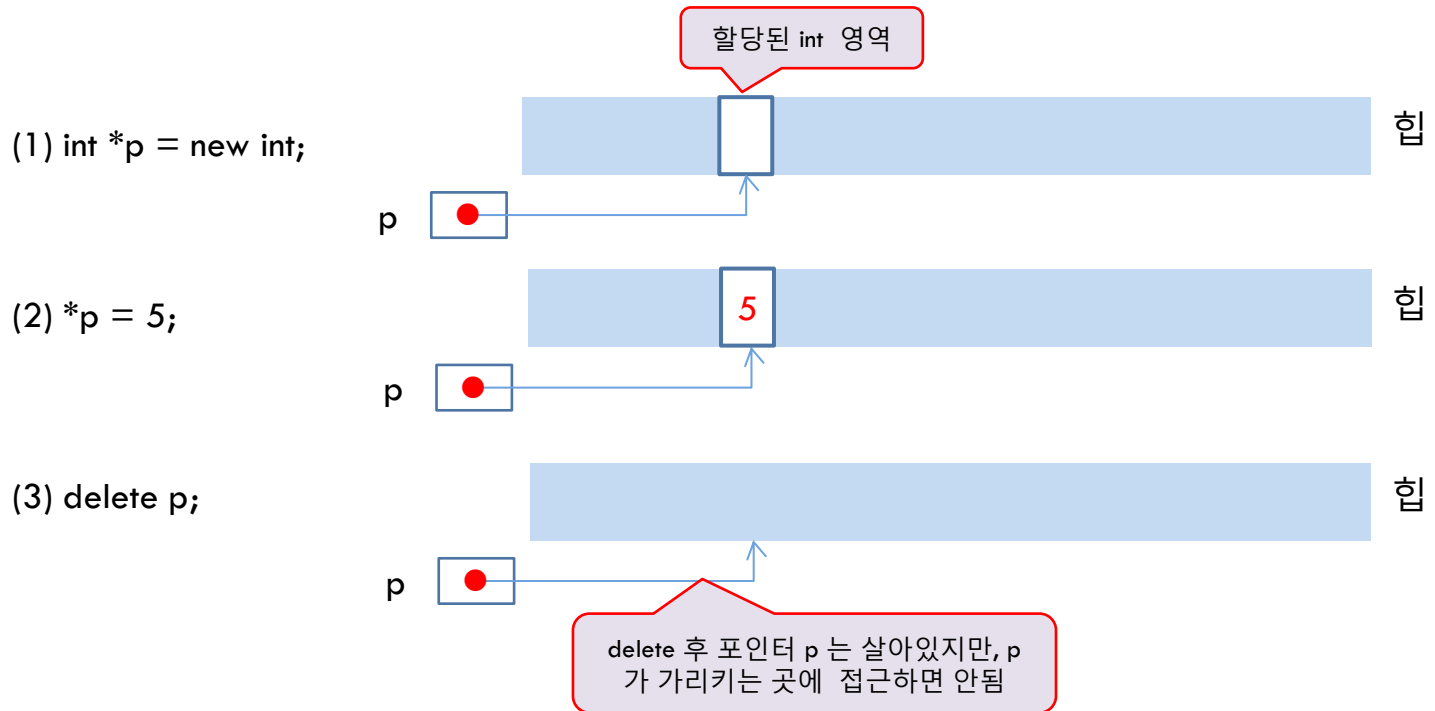
- C++의 기본 연산자
- new/delete 연산자의 사용 형식

```
데이터타입 *포인터변수 = new 데이터타입 ;  
delete 포인터변수;
```

- new/delete의 사용

```
int *pInt = new int; // int 타입의 메모리 동적 할당  
char *pChar = new char; // char 타입의 메모리 동적 할당  
Circle *pCircle = new Circle(); // Circle 클래스 타입의 메모리 동적 할당  
  
delete pInt; // 할당 받은 정수 공간 반환  
delete pChar; // 할당 받은 문자 공간 반환  
delete pCircle; // 할당 받은 객체 공간 반환
```

기본 타입의 메모리 동적 할당 및 반환



예제 4-5 정수형 공간의 동적 할당 및 반환 예

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

int main() {
    int *p;

    p = new int;
    if(!p) {
        cout << "메모리를 할당할 수 없습니다.";
        return 0;
    }

    *p = 5; // 할당 받은 정수 공간에 5 삽입
    int n = *p;
    cout << "*p = " << *p << '\n';
    cout << "n = " << n << '\n';

    delete p;
}
```

int 타입 1개 할당

p 가 NULL이면, 메모리 할당 실패

할당 받은 메모리 반환

```
*p = 5
n = 5
```

delete 사용 시 주의 사항

- 적절치 못한 포인터로 delete하면 실행 시간 오류 발생
 - ▶ 동적으로 할당 받지 않는 메모리 반환 - 오류

```
int n;  
int *p = &n;  
delete p; // 실행 시간 오류  
// 포인터 p가 가리키는 메모리는 동적으로 할당 받은 것이 아님
```

- ▶ 동일한 메모리 두 번 반환 - 오류

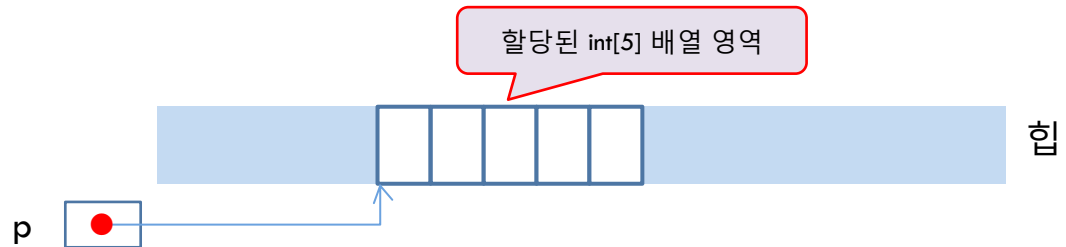
```
int *p = new int;  
delete p; // 정상적인 메모리 반환  
delete p; // 실행 시간 오류. 이미 반환한 메모리를 중복 반환할 수 없음
```

배열의 동적 할당 및 반환

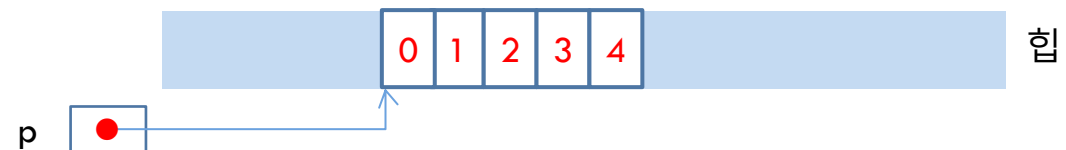
- new/delete 연산자의 사용 형식

데이터타입 *포인터변수 = **new** 데이터타입 [배열의 크기]; // 동적 배열 할당
delete [] 포인터변수; // 배열 반환

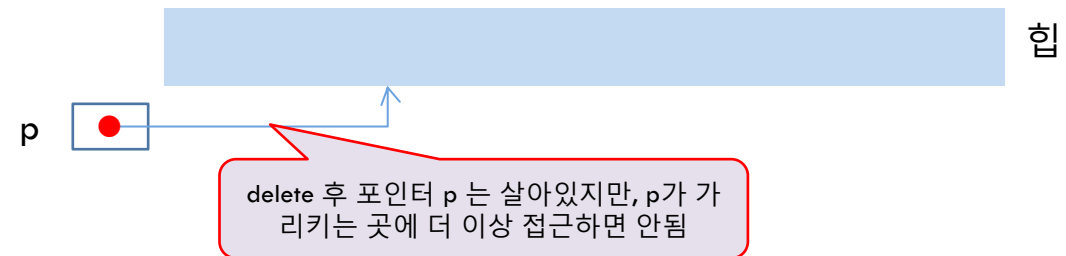
(1) int *p = new int [5];



(2) for(int i=0; i<5; i++)
p[i] = i;



(3) delete [] p;



예제 4-6 정수형 배열의 동적 할당 및 반환

사용자로부터 입력할 정수의 개수를
입력 받아 배열을 동적 할당 받고,
하나씩 정수를 입력 받은 후
합을 출력하는 프로그램을 작성하라.

```
입력할 정수의 개수는?4
1번째 정수: 4
2번째 정수: 20
3번째 정수: -5
4번째 정수: 9
평균 = 7
```

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

int main() {
    cout << "입력할 정수의 개수는?";
    int n;
    cin >> n; // 정수의 개수 입력
    if(n <= 0) return 0;
    int *p = new int[n]; // n 개의 정수 배열 동적 할당
    if(!p) {
        cout << "메모리를 할당할 수 없습니다.";
        return 0;
    }

    for(int i=0; i<n; i++) {
        cout << i+1 << "번째 정수: "; // 프롬프트 출력
        cin >> p[i]; // 키보드로부터 정수 입력
    }

    int sum = 0;
    for(int i=0; i<n; i++)
        sum += p[i];
    cout << "평균 = " << sum/n << endl;

    delete [] p; // 배열 메모리 반환
}
```

동적 할당 메모리 초기화 및 delete 시 유의 사항

- 동적 할당 메모리 초기화

- ▶ 동적 할당 시 초기화

```
데이터타입 *포인터변수 = new 데이터타입(초깃값);
```

```
int *pInt = new int(20); // 20으로 초기화된 int 타입 할당  
char *pChar = new char('a'); // 'a'로 초기화된 char 타입 할당
```

- ▶ 배열은 동적 할당 시 초기화 불가능

```
int *pArray = new int [10](20); // 구문 오류. 컴파일 오류 발생  
int *pArray = new int(20)[10]; // 구문 오류. 컴파일 오류 발생
```

- delete시 [] 생략

- ▶ 컴파일 오류는 아니지만 비정상적인 반환

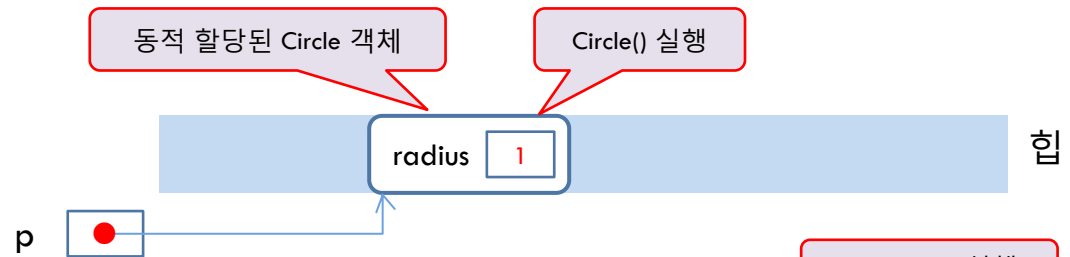
```
int *p = new int [10];  
delete p; // 비정상 반환. delete [] p;로 하여야 함.
```

```
int *q = new int;  
delete [] q; // 비정상 반환. delete q;로 하여야 함.
```

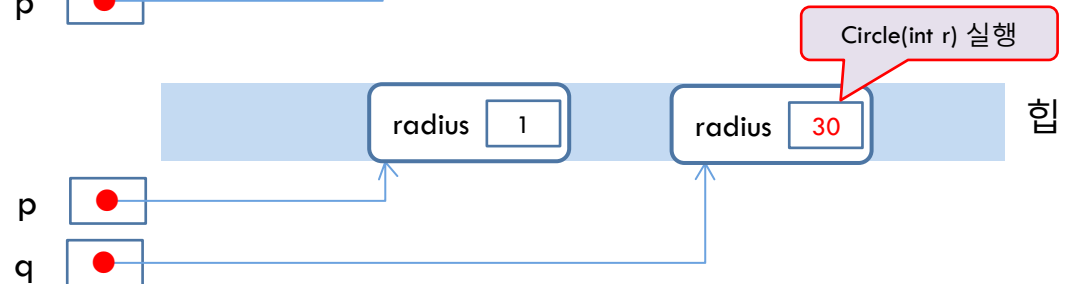

객체의 동적 생성 및 반환

```
클래스이름 *포인터변수 = new 클래스이름;  
클래스이름 *포인터변수 = new 클래스이름(생성자매개변수리스트);  
delete 포인터변수;
```

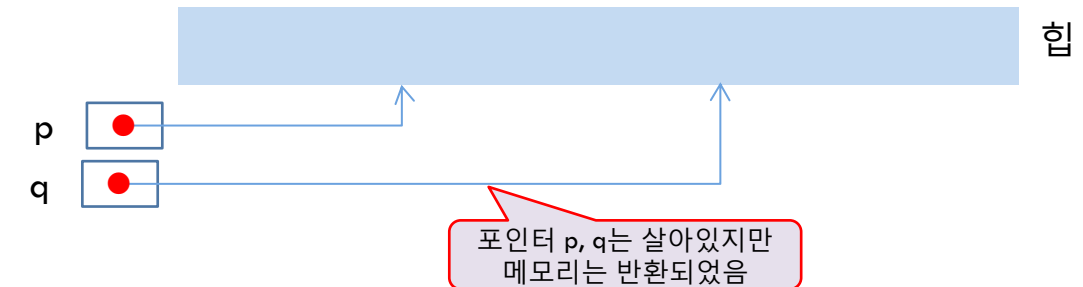
(1) Circle *p = new Circle;



(2) Circle *q = new Circle(30);



(3) delete p;
delete q;



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;
}

Circle::~~Circle() {
    cout << "소멸자 실행 radius = " << radius << endl;
}
```

```
int main() {
    Circle *p, *q;
    p = new Circle;
    q = new Circle(30);
    cout << p->getArea() << endl << q->getArea() << endl;
    delete p;
    delete q;
}
```

생성한 순서에 관계 없이 원하는
순서대로 delete 할 수 있음

```
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;
}

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 입력되어 종료한다.
        Circle *p = new Circle(radius); // 동적 객체 생성
        cout << "원의 면적은 " << p->getArea() << endl;
        delete p; // 객체 반환
    }
}
```

delete 문이 없다면
메모리 누수 발생

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

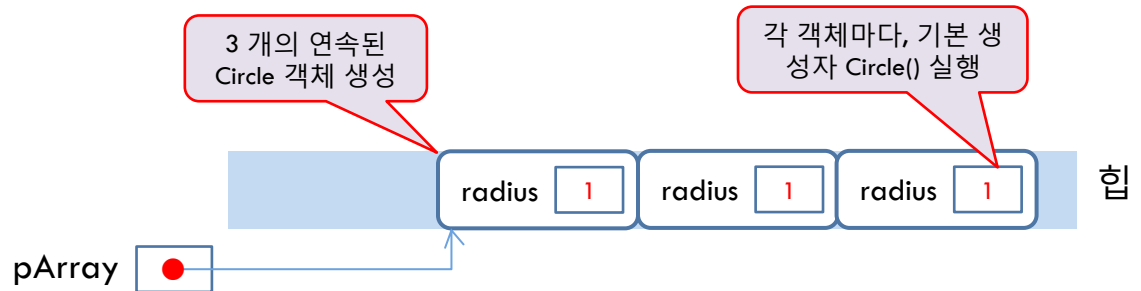
음수가 입력되면 종료

객체 배열의 동적 생성 및 반환

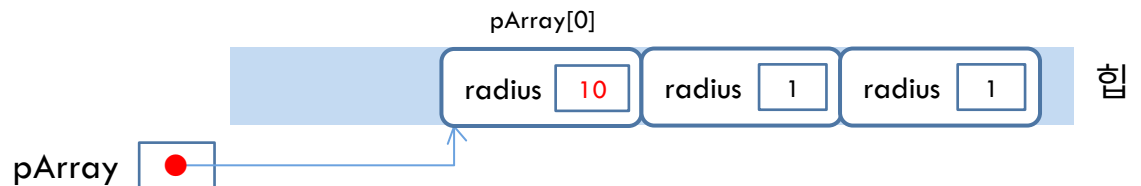
클래스이름 *포인터변수 = **new** 클래스이름 [배열 크기];

delete [] 포인터변수; // 포인터변수가 가리키는 객체 배열을 반환

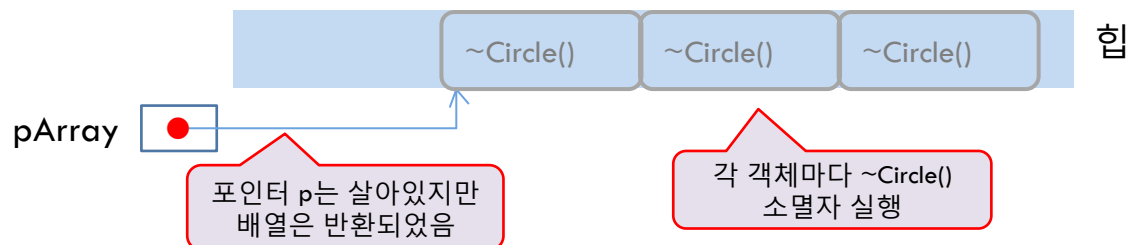
(1) Circle *pArray = new Circle[3];



(2) `pArray[0].setRadius(10);`



(3) `delete [] pArray;`



객체 배열의 사용, 배열의 반환과 소멸자

- 동적으로 생성된 배열도 보통 배열처럼 사용

```
Circle *pArray = new Circle[3]; // 3개의 Circle 객체 배열의 동적 생성

pArray[0].setRadius(10); // 배열의 첫 번째 객체의 setRadius() 멤버 함수 호출
pArray[1].setRadius(20); // 배열의 두 번째 객체의 setRadius() 멤버 함수 호출
pArray[2].setRadius(30); // 배열의 세 번째 객체의 setRadius() 멤버 함수 호출

for(int i=0; i<3; i++) {
    cout << pArray[i].getArea(); // 배열의 i 번째 객체의 getArea() 멤버 함수 호출
}
```

- 포인터로 배열 접근

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

- 배열 소멸

```
delete [] pArray;
```

pArray[2] 객체의 소멸자 실행(1)
pArray[1] 객체의 소멸자 실행(2)
pArray[0] 객체의 소멸자 실행(3)

각 원소 객체의 소멸자 별도 실행. 생성의 반대순

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;
}
```

```
int main() {
    Circle *pArray = new Circle [3]; // 객체 배열 생성

    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; // 포인터 p에 배열의 주소값으로 설정
    for(int i=0; i<3; i++) {
        cout << p->getArea() << '\n';
        p++; // 다음 원소의 주소로 증가
    }

    delete [] pArray; // 객체 배열 소멸
}
```

각 원소 객체의
기본 생성자 Circle() 실행

각 배열 원소 객체의
소멸자 ~Circle() 실행

```
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
```

소멸자는 생성의
반대 순으로 실행

예제 4-10 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;
}
```

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

```
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
    }

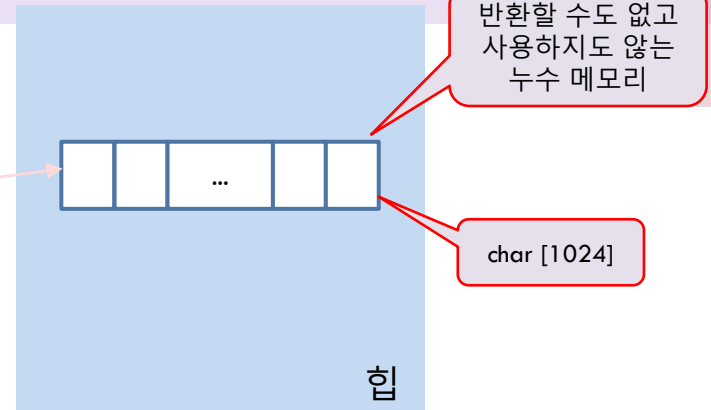
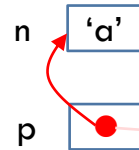
    int count =0; // 카운트 변수
    Circle* p = pArray;
    for(int i=0; i<n; i++) {
        cout << p->getArea() << ' '; // 원의 면적 출력
        if(p->getArea() >= 100 && p->getArea() <= 200)
            count++;
        p++;
    }
    cout << endl << "면적이 100에서 200 사이인 원의 개수는 "
        << count << endl;

    delete [] pArray; // 객체 배열 소멸
}
```

동적 메모리 할당과 메모리 누수

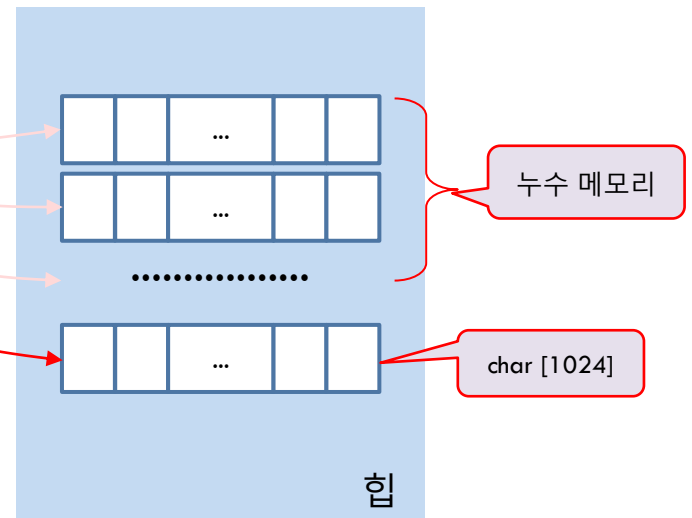
```
char n = 'a';  
char *p = new char[1024];  
p = &n;
```

p가 n을 가리키면 할당 받은 1024 바이트의 메모리 누수 발생



```
char *p;  
for(int i=0; i<1000000; i++) {  
    p = new char [1024];  
}
```

p는 새로 할당 받는 1024 바이트의 메모리를 가리키므로 이전에 할당 받은 메모리의 누수 발생



* 프로그램이 종료되면, 운영체제는 누수 메모리를 모두 힙에 반환함



this 포인터

this 포인터

- this

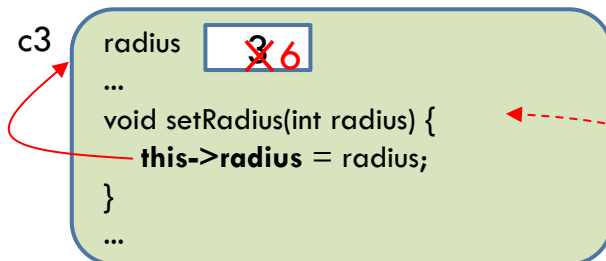
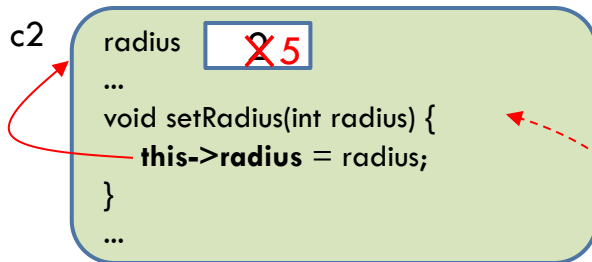
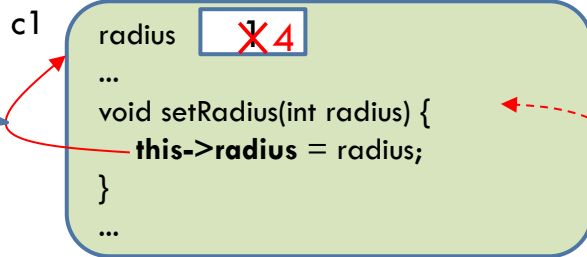
- ▶ 포인터, 객체 자신 포인터
- ▶ 클래스의 멤버 함수 내에서만 사용
- ▶ 개발자가 선언하는 변수가 아니고, 컴파일러가 선언한 변수
 - ▶ 멤버 함수에 컴파일러에 의해 묵시적으로 삽입 선언되는 매개 변수

```
class Circle {  
    int radius;  
public:  
    Circle() { this->radius=1; }  
    Circle(int radius) { this->radius = radius; } // radius = radius로 하면 오류 발생  
    void setRadius(int radius) { this->radius = radius; }  
    ....  
};
```

this와 객체

* 각 객체 속의 this는 다른 객체의 this와 다름

this는 객체 자신
에 대한 포인터

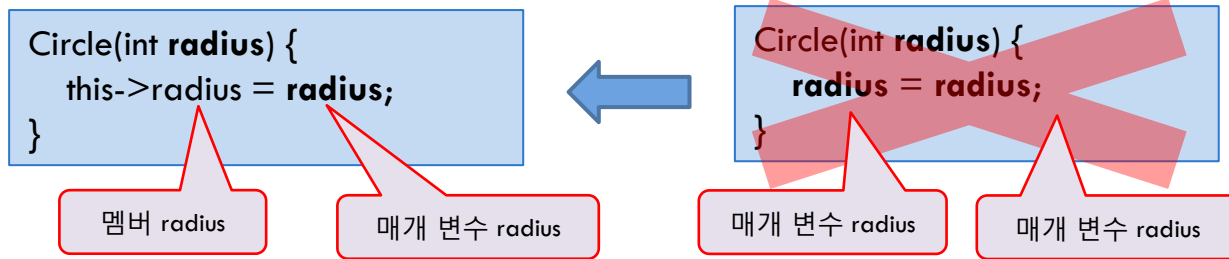


```
class Circle {  
    int radius;  
public:  
    Circle() {  
        this->radius=1;  
    }  
    Circle(int radius) {  
        this->radius = radius;  
    }  
    void setRadius(int radius) {  
        this->radius = radius;  
    }  
};
```

```
int main() {  
    Circle c1;  
    Circle c2(2);  
    Circle c3(3);  
  
    c1.setRadius(4);  
    c2.setRadius(5);  
    c3.setRadius(6);  
}
```

this가 필요한 경우

- 매개변수의 이름과 멤버 변수의 이름이 같은 경우



- 멤버 함수가 객체 자신의 주소를 리턴 할 때
 - ▶ 연산자 중복 시에 매우 필요

```
class Sample {  
public:  
    Sample* f() {  
        ....  
        return this;  
    }  
};
```

this의 제약 사항

- 멤버 함수가 아닌 함수에서 `this` 사용 불가
 - ▶ 객체와의 관련성이 없기 때문
- `static` 멤버 함수에서 `this` 사용 불가
 - ▶ 객체가 생기기 전에 `static` 함수 호출이 있을 수 있기 때문에

this 포인터의 실체 - 컴파일러에서 처리

```
class Sample {  
    int a;  
public:  
    void setA(int x) {  
        this->a = x;  
    }  
};
```

(a) 개발자가 작성한 클래스

컴파일러에 의해
변환

```
class Sample {  
    ...  
public:  
    void setA(Sample* this, int x) {  
        this->a = x;  
    }  
};
```

this는 컴파일러에 의해 묵시적으로 삽입된 매개 변수

(b) 컴파일러에 의해 변환된 클래스

```
Sample ob;
```

```
ob.setA(5);
```

컴파일러에 의해 변환

```
ob.setA(&ob, 5);
```

ob의 주소가 this 매개변수에 전달됨

(c) 객체의 멤버 함수를 호출하는 코드의 변환



string 클래스

string 클래스를 이용한 문자열

- C++ 문자열
 - ▶ C-스트링
 - ▶ C++ string 클래스의 객체
- string 클래스
 - ▶ C++ 표준 라이브러리, <string> 헤더 파일에 선언

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

- ▶ 가변 크기의 문자열

```
string str = "I love "; // str은 'I', ' ', 'l', 'o', 'v', 'e', ' '의 7개 문자로 구성
str.append("C++."); // str은 "I love C++."이 된다. 11개의 문자
```

- ▶ 다양한 문자열 연산을 실행하는 연산자와 멤버 함수 포함
 - ▶ 문자열 복사, 문자열 비교, 문자열 길이 등
- ▶ 문자열, 스트링, 문자열 객체, string 객체 등으로 혼용

string 객체 생성 및 입출력

- ▶ 문자열 생성 => 생성자 종류는 교재 185페이지 참조

```
string str; // 빈 문자열을 가진 스트링 객체
string address("서울시 성북구 삼선동 389"); // 문자열 리터럴로 초기화
string copyAddress(address); // address를 복사한 copyAddress 생성

// C-스트링(char [] 배열)으로부터 스트링 객체 생성
char text[] = {'L', 'o', 'v', 'e', ' ', 'C', '+', '+', '\0'};
string title(text); // "Love C++" 문자열을 가진 title 생성
```

- ▶ 문자열 출력

- ▶ cout과 << 연산자

```
cout << address << endl; // "서울시 성북구 삼선동 389" 출력
cout << title << endl; // "Love C++" 출력
```

- ▶ 문자열 입력

- ▶ cin과 >> 연산자

```
string name;
cin >> name; // 공백이 입력되면 하나의 문자열로 입력
```

string 객체의 동적 생성

- new/delete를 이용하여 문자열을 동적 생성/반환 가능

```
string *p = new string("C++"); // 스트링 객체 동적 생성
```

```
cout << *p; // "C++" 출력
```

```
p->append(" Great!!"); // p가 가리키는 스트링이 "C++ Great!!"이 됨
```

```
cout << *p; // "C++ Great!!" 출력
```

```
delete p; // 스트링 객체 반환
```

예제 4-11 string 클래스를 이용한 문자열 생성 및 출력

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

int main() {
    // 스트링 생성
    string str; // 빈 문자열을 가진 스트링 객체 생성
    string address("서울시 성북구 삼선동 389");
    string copyAddress(address); // address의 문자열을 복사한 스트링 객체 생성

    char text[] = {'L', 'o', 'v', 'e', ' ', 'C', '+', '+', '\0'}; // C-스트링
    string title(text); // "Love C++" 문자열을 가진 스트링 객체 생성

    // 스트링 출력
    cout << str << endl; // 빈 스트링. 아무 값도 출력되지 않음
    cout << address << endl;
    cout << copyAddress << endl;
    cout << title << endl;
}
```

string 클래스를
사용하기 위해
반드시 필요

빈 문자열을 가
진 스트링 출력

서울시 성북구 삼선동 389
서울시 성북구 삼선동 389
Love C++

string 클래스의 주요 멤버 함수

멤버함수	설명
string& append(string& str)	문자열 뒤에 str 추가
string& insert(int pos, string& str)	문자열의 pos 위치에 str 삽입
string& replace(int pos, int n, string& str)	문자열의 pos 위치부터 n문자를 str 문자열로 대치
int size()	문자열의 길이 리턴, 바이트 수
int length()	문자열의 길이 리턴, size()와 동일
string& erase(int pos, int n)	pos위치부터 n개 문자 삭제
void clear()	문자열 모두 삭제, 크기를 0으로 만듦
char& at(int pos)	pos위치의 문자 리턴
int find(string& str)	문자열의 처음부터 str을 검색하여 발견한 처음 인덱스 리턴. 없으면 -1 리턴
int compare(string& str)	문자열과 str을 비교하여 같으면 0을, 사전 순으로 현재 문자열 이 앞에 오면 음수, 뒤에 오면 양수 리턴
string substr(int pos, int n)	pos위치부터 n개 문자를 새로운 서브스트링으로 생성, 리턴
void swap(string& str1, string str2)	str1과 str2를 서로 교환함

string 클래스의 연산자

String s="C++", String s1="C", String s2="Java"

연산자	설명	사용 예	결과
s1 = s2	s2를 s1에 치환	s1 = s2	s1="Java"
s[]	s의 []인덱스에 있는 문자	char c=s[1]	c='+'
s1+s2	s1과 s2를 연결한 새로운 문자열	s1+s2	"CJava"
s1+=s2	s1에 s2문자열 연결	s1+=s2	s1="CJava"
s1 == s2	s1과 s2가 같은 문자열이면 true	s1==s2	false
s1 != s2	s1과 s2가 다른 문자열이면 true	s1!=s2	true
s1 < s2	s1이 사전 순으로 s2보다 앞에 오면 true	s1 < s2	true
s1 > s2	s1이 사전 순으로 s2보다 뒤에 오면 true	s1 > s2	false
s1 <= s2	s1이 s2와 같거나 앞에 오면 true	s1 <= s2	true
s1 >= s2	s1이 s2와 같거나 뒤에 오면 true	s1 >= s2	false

string 클래스의 함수 및 연산자 사용 예

```
string str = "I" ;  
str.append( " love" );  
str.erase( 3, 4);
```

```
string newstr( "I love C++" );  
string str2 = newstr.substr(2, 4);  
newstr.insert( 2, "really" );  
newstr.replace(2, 11, "study" );
```

```
string a( "I love C++" );  
a.erase(0,7);  
a.clear();
```

```
string e = "I love love C++" ;  
int index = e.find(love);
```

예제 4-12 string 배열 선언과 문자열 키 입력 응용

5 개의 string 배열을 선언하고
getline()을 이용하여 문자열을 입력
받아 사전 순으로 가장 뒤에 나오는
문자열을 출력하라. 문자열 비교는
<, > 연산자를 간단히 이용하면 된
다.

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

int main() {
    string names[5]; // 문자열 배열 선언

    for(int i=0; i<5; i++) {
        cout << "이름 >> ";
        getline(cin, names[i], '\n');
    }

    string latter = names[0];
    for(int i=1; i<5; i++) {
        if(latter < names[i]) { // 사전 순으로 latter 문자열이 앞에 온다면
            latter = names[i]; // latter 문자열 변경
        }
    }
    cout << "사전에서 가장 뒤에 나오는 문자열은 " << latter << endl;
}
```

```
이름 >> Jeong Won Seok
이름 >> Han Hong Jin
이름 >> Lee Young Hee
이름 >> Han Won Sun
이름 >> Hwang Su hee
사전에서 가장 뒤에 나오는 문자열은 Lee Young Hee
```

예제 4-13 문자열을 입력 받고 회전시키기

빈칸을 포함하는 문자열을 입력 받고, 한 문자씩 왼쪽으로 회전하도록 문자열을 변경하고 출력하라.

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

int main() {
    string s;

    cout << "문자열을 입력하세요(한글 안됨) " << endl;
    getline(cin, s, '\n'); // 문자열 입력
    int len = s.length(); // 문자열의 길이

    for(int i=0; i<len; i++) {
        string first = s.substr(0,1); // 맨 앞의 문자 1개를 문자열로 분리
        string sub = s.substr(1, len-1); // 나머지 문자들을 문자열로 분리
        s = sub + first; // 두 문자열을 연결하여 새로운 문자열로 만들
        cout << s << endl;
    }
}
```

문자열을 입력하세요 (한글 안됨)

I love you
love youl
love youl
ove youl I
ve youl lo
e youl lov
youl love
youl love
oul love y
ul love yo
I love you

예제 4-14 문자열 처리 응용 - 덧셈 문자열을 입력받아 덧셈 실행

4+125+4+77+102 등으로 표현된 덧셈식을 문자열로 입력받아 계산하는 프로그램 작성하라.

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

int main() {
    string s;
    cout << "7+23+5+100+25와 같이 덧셈 문자열을 입력하세요." << endl;
    getline(cin, s, '\n'); // 문자열 입력
    int sum = 0;
    int startIndex = 0; // 문자열 내에 검색할 시작 인덱스
    while(true) {
        int flIndex = s.find('+', startIndex);
        if(flIndex == -1) { // '+' 문자 발견할 수 없음
            string part = s.substr(startIndex);
            if(part == "") break; // "2+3+", 즉 +로 끝나는 경우
            cout << part << endl;
            sum += stoi(part); // 문자열을 수로 변환하여 더하기
            break;
        }
        int count = flIndex - startIndex; // 서브스트링으로 자를 문자 개수
        string part = s.substr(startIndex, count); // startIndex부터 count 개의 문자로 서브스트링 만들기
        cout << part << endl;
        sum += stoi(part); // 문자열을 수로 변환하여 더하기
        startIndex = flIndex+1; // 검색을 시작할 인덱스 전진시킴
    }
    cout << "숫자들의 합은 " << sum;
}
```

7+23+5+100+25와 같이 덧셈 문자열을 입력하세요.
66+2+8+55+100
66
2
8
55
100
숫자들의 합은 231

예제 4-15 문자열 find 및 replace

&가 입력될 때까지 여러 줄의 영문 문자열을 입력 받고, 찾는 문자열과 대치할 문자열을 각각 입력 받아 문자열을 변경하라.

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

int main() {
    string s;
    cout << "여러 줄의 문자열을 입력하세요. 입력의 끝은 &문자입니다." << endl;
    getline(cin, s, '&'); // 문자열 입력
    cin.ignore();
    string f, r;
    cout << endl << "find: ";
    getline(cin, f, '\n'); // 검색할 문자열 입력
    cout << "replace: ";
    getline(cin, r, '\n'); // 대치할 문자열 입력

    int startIndex = 0;
    while(true) {
        int fIndex = s.find(f, startIndex); // startIndex부터 문자열 f 검색
        if(fIndex == -1)
            break; // 문자열 s의 끝까지 변경하였음
        s.replace(fIndex, f.length(), r); // fIndex부터 문자열 f의 길이만큼 문자열 r로 변경
        startIndex = fIndex + r.length();
    }
    cout << s << endl;
}
```

& 뒤에 따라 오는 <Enter> 키를 제거하기 위한 코드!!!

예제 4-15 실행 결과

여러 줄의 문자열을 입력하세요. 입력의 끝은 &문자입니다.

It's now or never, come hold me tight. Kiss me my darling, be mine tonight

Tomorrow will be too late. It's now or never, my love won't wait&

find: now

replace: Right Now

It's Right Now or never, come hold me tight. Kiss me my darling, be mine tonight

Tomorrow will be too late. It's Right Now or never, my love won't wait

검색할 단어

대치할 단어

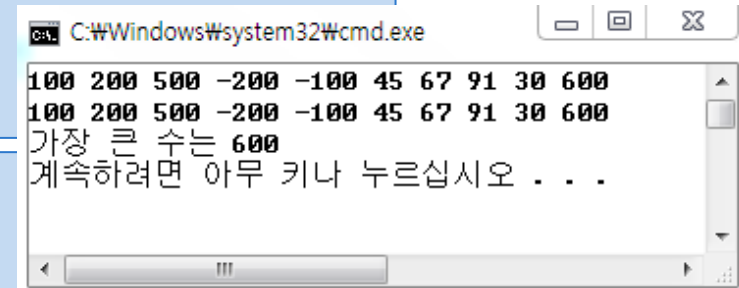
& 뒤에<Enter>키
입력

연습문제 1 – Sample클래스

- 다음과 같은 Sample 클래스가 있다. main()함수가 실행되도록 Sample 클래스를 완성하시오.

```
class Sample {  
    int *p;  
    int size;  
  
public:  
    Sample(int n) { // 생성자  
        size = n; p = new int [n]; // n개 정수 배열의 동적 생성  
    }  
    void read(); // 동적 할당받은 정수 배열 p에 사용자로부터 정수를 입력 받음  
    void write(); // 정수 배열을 화면에 출력  
    int big(); // 정수 배열에서 가장 큰 수 리턴  
    ~Sample(); // 소멸자  
};
```

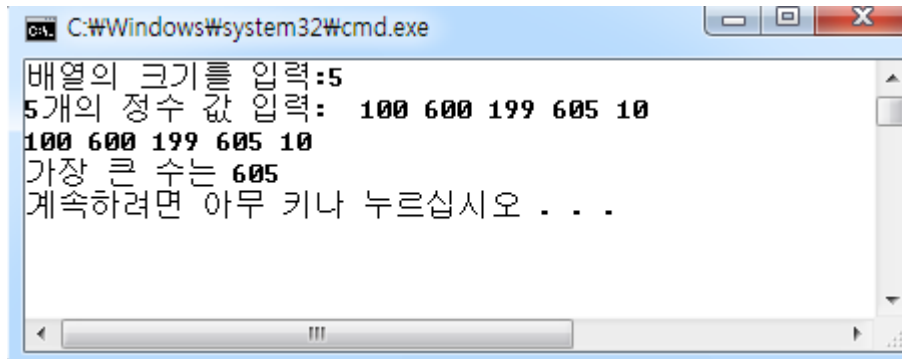
```
int main() {  
    Sample s(10); // 10개 정수 배열을 가진 Sample 객체 생성  
    s.read(); // 키보드에서 정수 배열 읽기  
    s.write(); // 정수 배열 출력  
    cout << "가장 큰 수는 " << s.big() << endl; // 가장 큰 수 출력  
}
```



```
C:\Windows\system32\cmd.exe  
100 200 500 -200 -100 45 67 91 30 600  
100 200 500 -200 -100 45 67 91 30 600  
가장 큰 수는 600  
계속하려면 아무 키나 누르십시오 . . .
```

연습문제 1 - 수정

- 앞의 연습문제를 다음과 같이 수행되도록 main()함수를 수정하십시오.

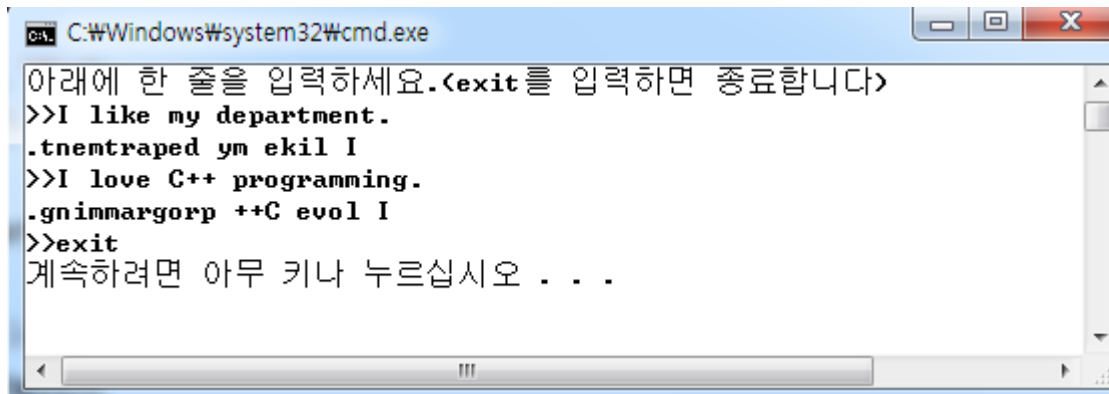


A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains the following text:

```
배열의 크기를 입력:5  
5개의 정수 값 입력: 100 600 199 605 10  
100 600 199 605 10  
가장 큰 수는 605  
계속하려면 아무 키나 누르십시오 . . .
```

연습문제 2

- string 클래스를 이용하여 사용자가 입력한 영문 한 줄을 문자열로 입력 받아 거꾸로 출력하는 프로그램을 작성하시오.



```
C:\Windows\system32\cmd.exe
아래에 한 줄을 입력하세요.<exit를 입력하면 종료합니다>
>>I like my department.
.tnemtrapd ym ekil I
>>I love C++ programming.
.gnimmargorp ++C evol I
>>exit
계속하려면 아무 키나 누르십시오 . . .
```

연습문제 3 – Family 클래스

- 다음에서 Person은 사람을, Family는 가족을 추상화한 클래스로서 완성되지 않은 클래스이다.

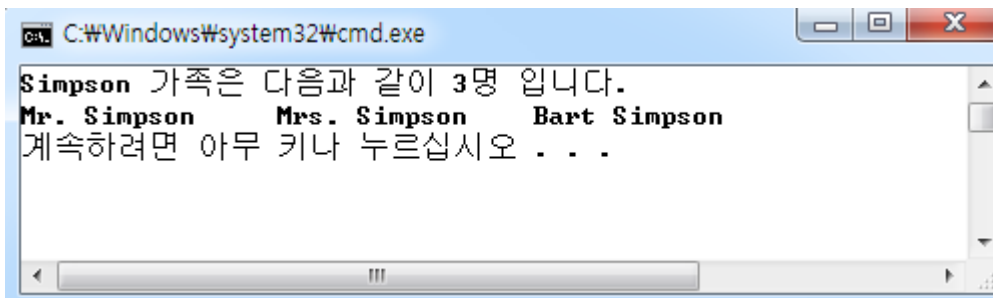
```
class Person {
    string name;
public:
    Person() { name=""; }
    Person(string name) { this->name = name; }
    string getName() { return name; }
    void setName(string name) { this->name = name; }
};

class Family {
    string name;
    Person* p; // Person 배열 포인터
    int size; // Person 배열의 크기. 가족 구성원 수
public:
    Family(string name, int size); // size 개수만큼 Person 배열 동적 생성
    void setName(int index, string name);
    void show(); // 모든 가족 구성원 출력
    ~Family();
};
```

연습문제 3 – Family 클래스

- 다음 main()이 작동하도록 Person과 Family 클래스에 필요한 멤버들을 추가하고 코드를 완성하라.

```
int main() {  
    Family *simpson = new Family("Simpson", 3); // 3명으로 구성된 Simpson 가족  
    simpson->setName(0, "Mr. Simpson");  
    simpson->setName(1, "Mrs. Simpson");  
    simpson->setName(2, "Bart Simpson");  
    simpson->show();  
    delete simpson;  
}
```



The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
Simpson 가족은 다음과 같이 3명 입니다.  
Mr. Simpson    Mrs. Simpson    Bart Simpson  
계속하려면 아무 키나 누르십시오 . . .
```

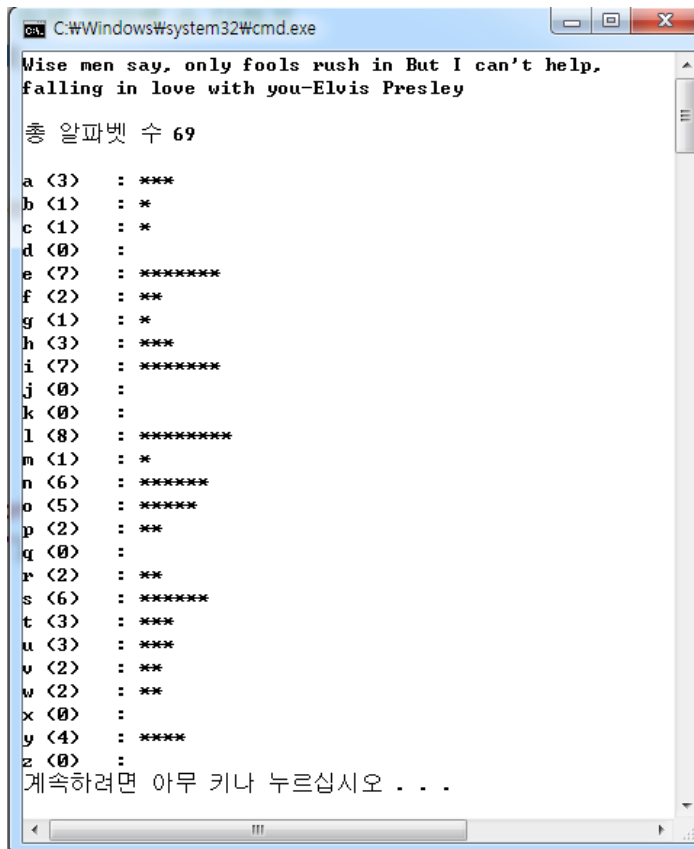

연습문제 4 – Histogram 클래스

- 영문자로 구성된 텍스트에 대해 각 알파벳에 해당하는 문자가 몇 개인지 출력하는 히스토그램 클래스 Histogram을 만들어 보자. 대문자는 모두 소문자로 변환하여 처리한다.
- Histogram클래스를 활용하는 사례와 실행 결과는 다음과 같다.

```
int main() {  
    Histogram elvisHisto("Wise men say, only fools rush in But I can't help, \n");  
    elvisHisto.put("falling in love with you");  
    elvisHisto.putc('-');  
    elvisHisto.put("Elvis Presley");  
    elvisHisto.print();  
}
```

연습문제 4 – Histogram 클래스

- 실행 결과



```
C:\Windows\system32\cmd.exe
Wise men say, only fools rush in But I can't help,
falling in love with you-Elvis Presley

총 알파벳 수 69

a <3> : ***
b <1> : *
c <1> : *
d <0> :
e <7> : *****
f <2> : **
g <1> : *
h <3> : ***
i <7> : *****
j <0> :
k <0> :
l <8> : *****
m <1> : *
n <6> : *****
o <5> : *****
p <2> : **
q <0> :
r <2> : **
s <6> : *****
t <3> : ***
u <3> : ***
v <2> : **
w <2> : **
x <0> :
y <4> : ****
z <0> :
계속하려면 아무 키나 누르십시오 . . .
```