Chapter 04. 객체 포인터와 객체 배열

학습 목표

1. 객체에 대한 포인터를 선언하고 활용할 수 있다.
2. 객체의 배열을 선언하고 활용할 수 있다.
3. this 포인터의 개념을 이해하고, 활용할 수 있다.
4. string 클래스를 이용하여 문자열을 다룰 수 있다.

객체 포인터

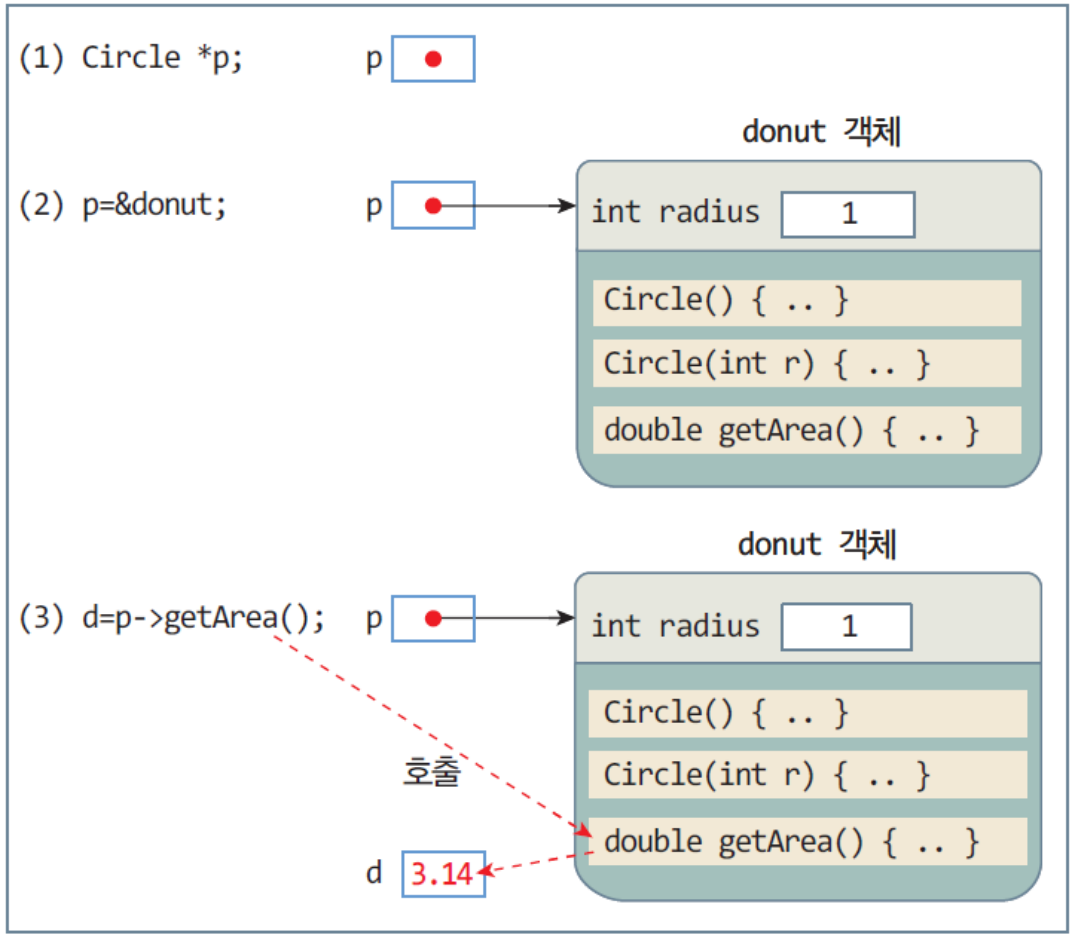
- 객체에 대한 포인터
  - C 언어의 포인터와 동일
  - 객체의 주소 값을 가지는 변수
- 포인터로 멤버를 접근할 때
  - 객체포인터->멤버

객체에 대한 포인터 선언

포인터에 객체 주소 저장

멤버 함수 호출

```
Circle donut;
double d = donut.getArea();

Circle *p; // (1)
p = &donut; // (2)
d = p->getArea(); // (3)
```



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, 생성 및 소멸

- 객체 배열 선언 가능
  - 기본 타입 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

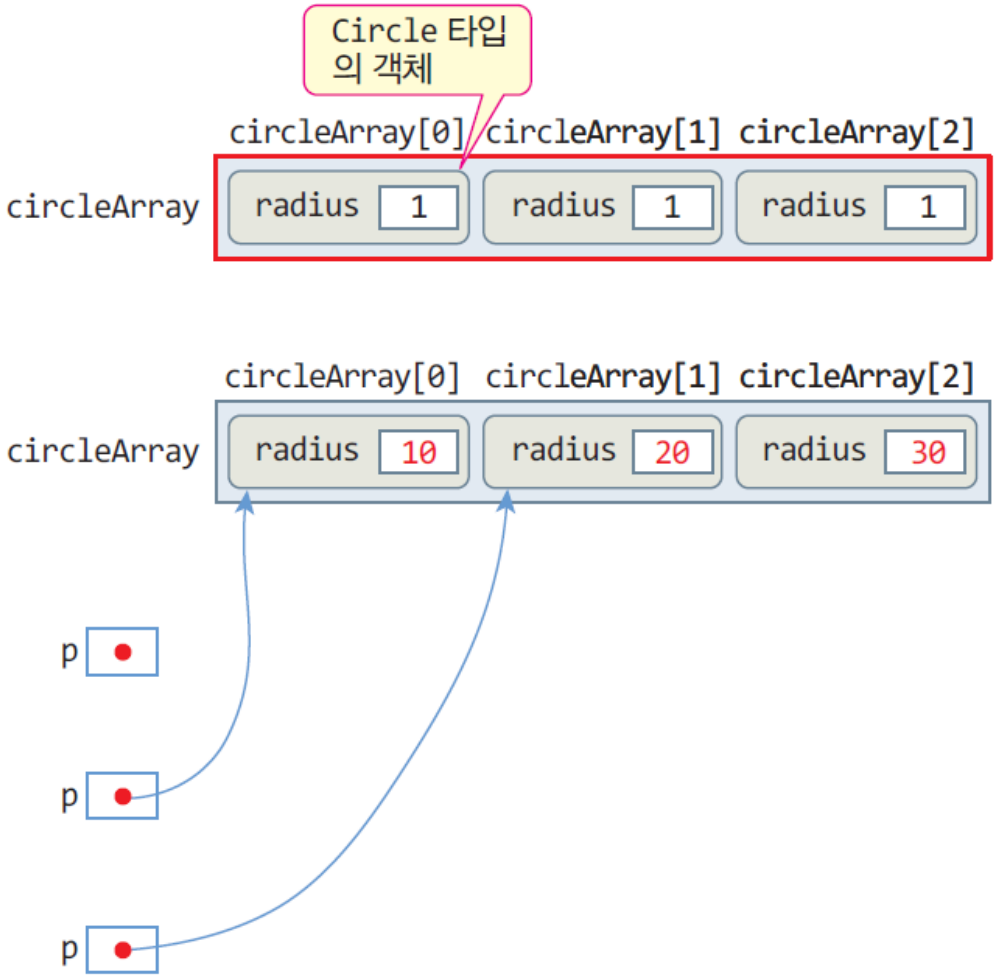
```
#include <iostream>
using namespace std;

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)

```
(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 생성자 호출

```
#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 컴파일 오류

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

(b) 기본 생성자가 없으므로 컴파일 오류

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

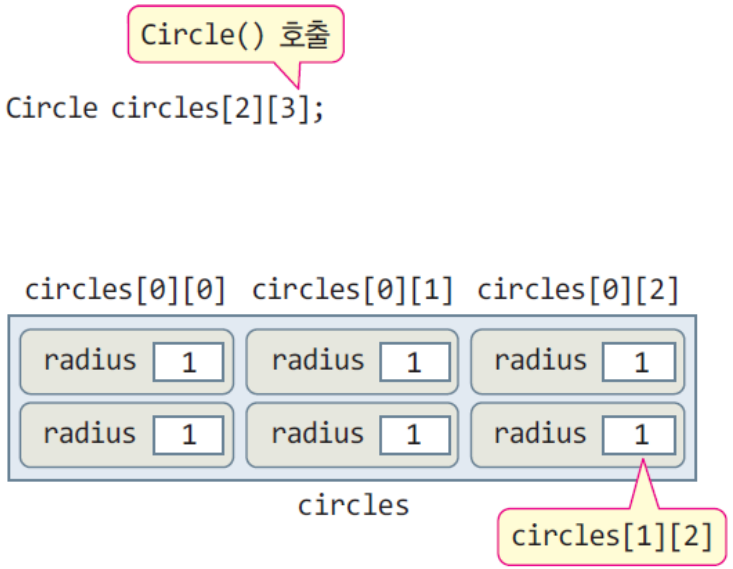
예제 4–3 객체 배열 초기화

```
#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;
}
```

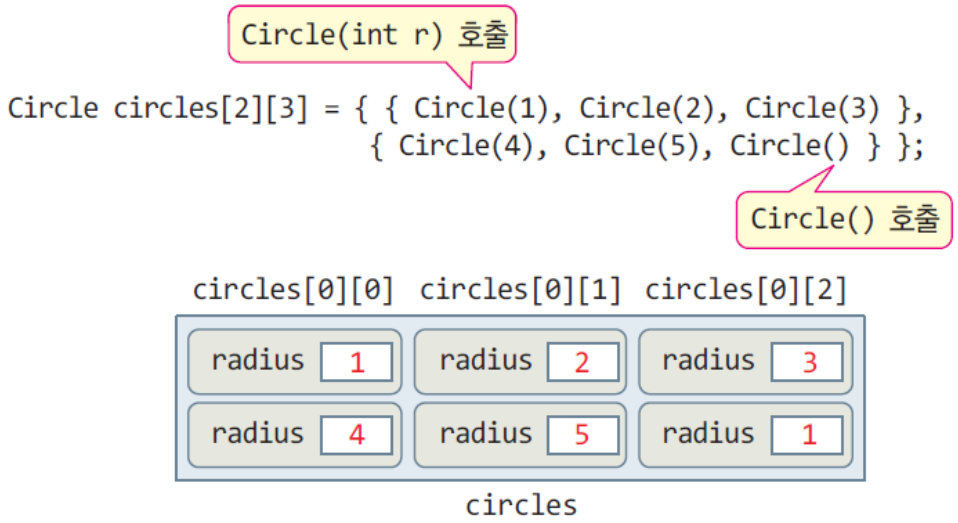
circleArray[0] 객체가 생성될 때, 생성자 Circle(10),
circleArray[1] 객체가 생성될 때, 생성자 Circle(20),
circleArray[2] 객체가 생성될 때, 기본 생성자 Circle()
이 호출된다.

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

2차원 배열



(a) 2차원 배열 선언 시



(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
```

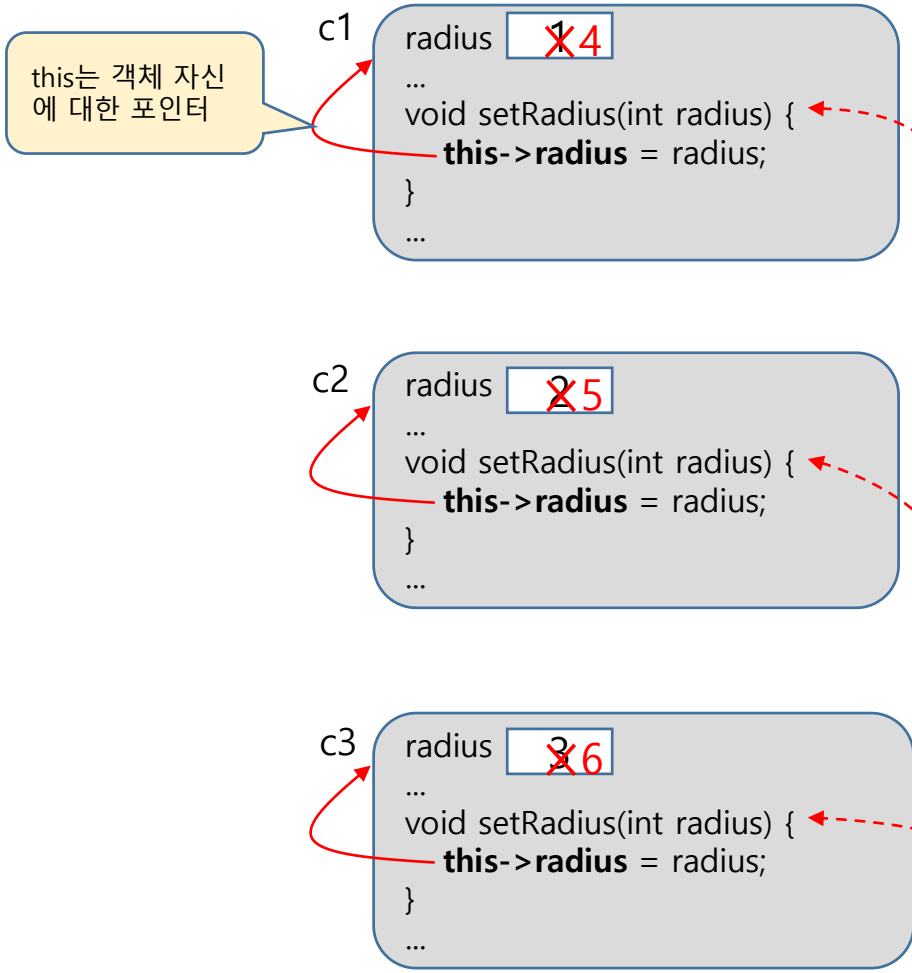
this 포인터

- 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; }
   void setRadius(int radius) { this->radius = radius; }
   ....
};
```

this와 객체

* 각 객체 속의 this는 다른 객체의 this와 다름

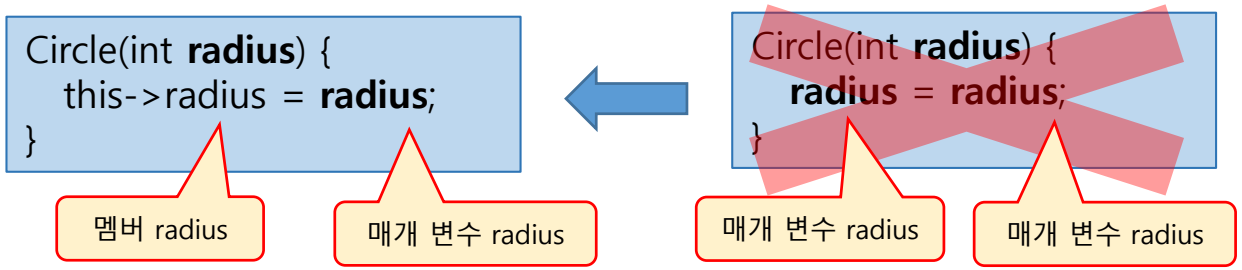



```
class Circle {
   int radius;
public:
   Circle() {
      this->radius=1;
   }
   Circle(int radius) {
      this->radius = radius;
   }
   void setRadius(int radius) {
      this->radius = radius;
   }
};
```

```
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 리턴할 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;
   }
};
```

컴파일러에 의해 변환

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

this는 컴파일러에 의해 묵시적으로 삽입된 매개 변수

(a) 개발자가 작성한 클래스

(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 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

```
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
```

- 문자열 숫자 변환
  - stoi() 함수 이용
    - 2011 C++ 표준부터

```
string s="123";
int n = stoi(s); // n은 정수 123. 비주얼 C++ 2010 이상 버전
```

```
string s="123";
int n = atoi(s.c_str()); // n은 정수 123. 비주얼 C++ 2008 이하
```

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 가진 스트링 출력

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

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 된다.

```
#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 >> Kim Nam Yun
이름 >> Chang Jae Young
이름 >> Lee Jae Moon
이름 >> Han Won Sun
이름 >> Hwang Su hee
사전에서 가장 뒤에 나오는 문자열은 Lee Jae Moon
```

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 만듦
        cout << s << endl;
    }
}
```

```
문자열을 입력하세요 (한글 안됨)
I love you
 love youI
love youI
ove youI l
ve youI lo
e youI lov
 youI love
youI love
ouI love y
uI love yo
I love you
```

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!!!

```
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

THANKS FOR YOUR ATTENTION

나사렛대학교
KOREA NAZARENE UNIVERSITY