chapter4.객체 포인터와 객체 배열, 객체의 동적 생성

객체 포인터

C++에서 객체를 다루기 위해 객체에 대한 포인터 변수를 선언하고 이 포인터 변수로 객체의 멤버 변수를 읽고 값을 쓰거나 멤버함수를 호출할 수 있음

객체에 대한 포인터 변수 선언 ex) Circle\* p;

선언된 포인터 변수는 아무 객체도 가리키고 있지 않음

포인터 변수에 객체 주소 지정 객체의 주소는 객체 이름 앞에 &연산자를 사용해 표현

ex) Circle\* p = &donut;

포인터를 이용한 객체 멤버 접근 객체 이름으로 멤버를 접근할 때는 .연산자 이용

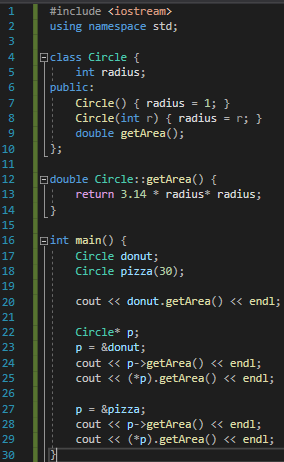
ex) d = donut.getArea();

객체 포인터로 멤버에 접근할 때는 ->연산자 사용

ex) d = p->getArea();

ex) d = (\*p).getArea();

예제4-1



check time 1. (1) p = &poly;

p->draw();

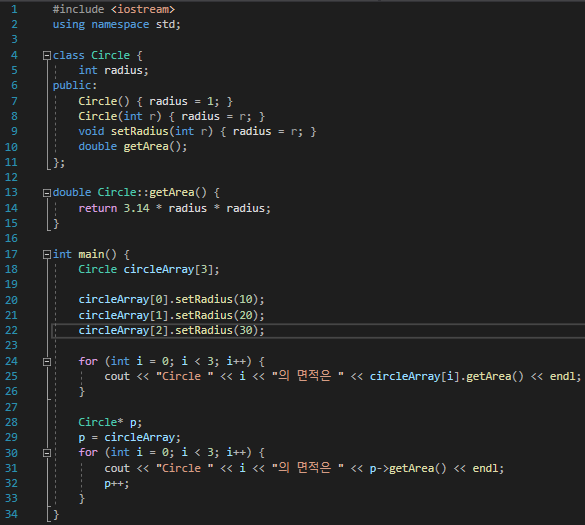
1. (2) 4

객체 배열

객체 배열 선언 및 활용 객체가 원소인 배열

기본타입의 배열을 선언하고 활용하는 방법과 동일

예제4-2



객체 배열 선언 Circle circleArray[3];

circleArray는 3개의 Circle객체를 원소로 가지는 배열

객체 배열 선언문은 기본 생성자만 호출한다 객체배열 선언문은 오직 매개변수 없는 기본 생성자만 호출

매개변수가 있는 생성자만 선언된 객체 배열을 만들 경우, 컴파일 오류

객체 배열 사용 배열의 각 원소 객체는 [] 연산자로 구분

원소객체와 멤버사이의 .연산자 사용

포인터를 사용한 배열 사용 Circle\* p;

p = circleArray;

for (int i = 0; i < 3; i++) {

cout << "Circle " << i << "의 면적은 " << p->getArea() << endl;

p++;

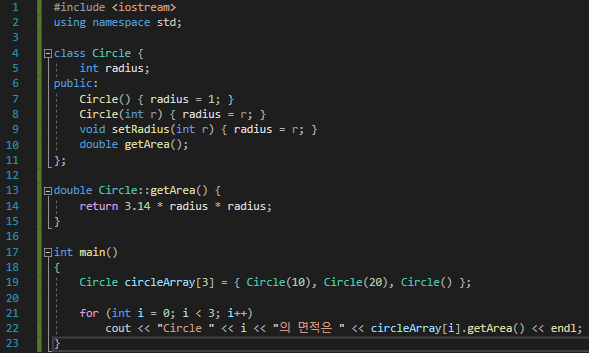
}

객체 배열 초기화 객체 배열을 생성할 때 매개변수 있는 생성자 사용가능

{ } 안에 호출할 생성자 지정

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

예제 4-3



다차원 객체 배열 배열 선언 ex) 2행 3열의 2차원 배열

Circle circles[2][3];

멤버 변수 초기화 멤버함수 이용

ex) circles[0][0].setRadius(1);

circles[0][1].setRadius(2);

circles[0][2].setRadius(3);

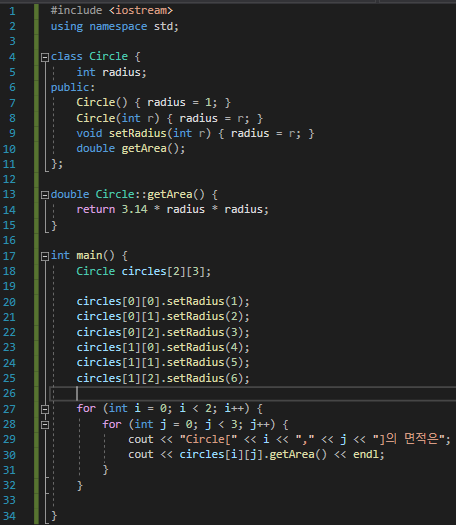
circles[1][1].setRadius(4);

생성자 지정 배열 초기화 { }안에 생성자 지정

ex) Circle circles[2][3] = { {Circle(1), Circle(2), Circle(3)},

{Circle(4), Circle(5), Circle()} };

예제4-4



check time 1. (1) 100 100 100

1. (2) 6 8 5 100

1. (3) Sample arr[3];

Sample\* p;

p = arr;

int sum = 0;

for (int i = 0; i < 3; i++)

sum += (p + i)->get();

cout << endl << sum;

1. (4) Sample arr2D[2][2] = { {Sample(2,3),Sample(2,4)} ,{Sample(5),Sample()} };

int sum = 0;

for (int i = 0; i < 2; i++)

sum += (\*arr2D+i)->get(); //객체 다차원 배열의 이름으로 포인터 연산을 할때 \*을 붙여야 가능하다

//왜냐면 객체 다차원 배열의 이름은 내부 배열의 이름(포인터)을 가리키기 때문이다

//내부 배열 자체가 객체의 포인터이기 때문에 연산이 되지않는다

//만약 \*를 붙인다면 다차원 배열이 가리키는 원소 배열을 받으므로

//원소 배열에서 포인터 연산이 가능해진다

//즉 원소배열의 원소 객체를 가리키게 된다

cout << endl << sum;

동적 메모리 할당 및 반환

C언어에서는 동적 메모리 할당 및 반환을 위해 malloc(), free() 사용

C++에서는 new와 delete 연산자 사용

new연산자 heap에서 메모리 할당 받음

delete연산자 할당받은 메모리 heap으로 반환

new연산자와 delete연산자 new와 delete의 기본활용 데이터타입 \*포인터변수 = new 데이터타입;

delete 포인터변수;

new 연산자 데이터타입의 크기만큼 힘으로부터 메모리를 할당받음

주소 리턴

힙 메모리가 부족하면 NULL 리턴

ex) int\* pInt = new int;

char\* pChar = new char;

Circle\* pCircle = new Circle();

delete pInt;

delete pChar;

delete pCircle;

동적 할당 메모리 초기화 데이터타입 \*포인터변수 = new 데이터타입(초깃값);

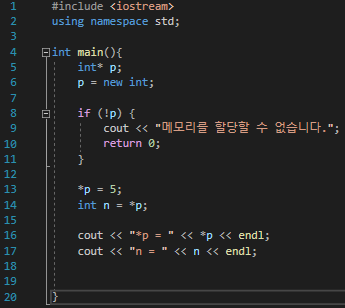
ex) int\* pInt = new int(20);

char\* pChar = new char('a');

delete 사용시 주의 동적으로 할당 받지 않은 메모리 반환 X

동일한 메모리 반환 반복 X

예제4-5



배열의 동적 할당 및 반환 배열의 동적할당∙반환의 기본 형식 데이터타입 \*포인터변수 = new 데이터타입 [배열의크기]

delete [] 포인터변수;

new연산자 배열의 크기만한 배열을 할당 🡪 주소 리턴

delete연산자 포인터변수가 가리키는 배열 메모리 반환

ex) int\* p = new int[5];

if (!p)

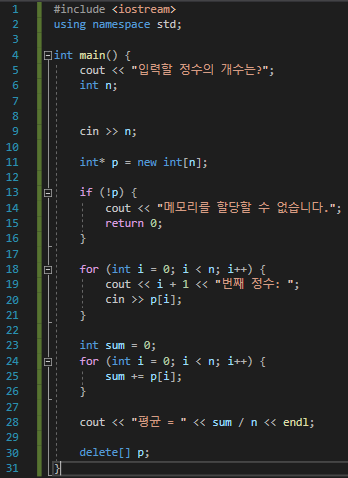
return;

for (int i = 0; i < 5; i++)

p[i] = i; //\*(p+i)=I

delete[] p;

예제4-6



배열을 초기화할 때 주의사항 생성자를 통한 배열의 초기값 지정불가

{ }을 통한 초기값 지정 ex) int\* pArray = new int[] {1, 2, 3, 4};

배열을 delete할 때 주의사항 []를 꼭 붙일 것

check time 1. (1) double \*dp = new double(3.14)

1. (2) int\* ip = new int[5];

cin >> ip[0];

int highnum = \*ip;

for (int i = 1; i < 5; i++) {

cin >> ip[i];

if (highnum < ip[i]) {

highnum = ip[i];

}

}

delete ip;

2. (1) delete [] p; 🡪 delete p;

2. (2) delete [] q; 🡪 delete q;

객체와 객체 배열의 동적 생성 및 반환

객체의 동적 생성 및 반환 new를 이용한 객체의 동적생성과 생성자 클래스이름\* 포인터변수 = new 클래스이름;

클래스이름\* 포인터변수 = new 클래스이름(매개변수);

ex) Circle\* p = new Circle;

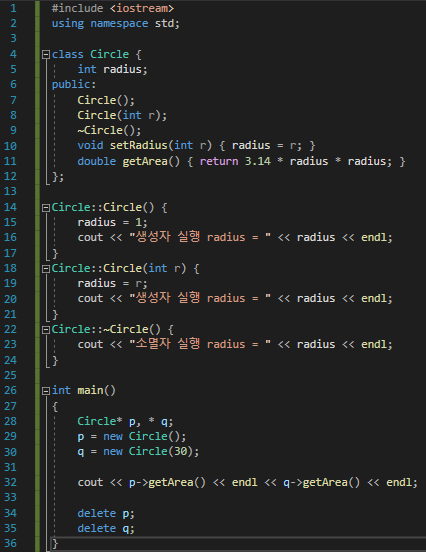
Circle\* q = new Circle(30);

delete를 이용한 객체 반환과 소멸자 delete 포인터 변수

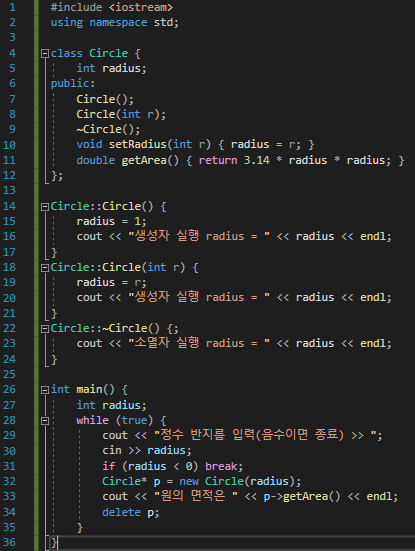
delete p;

delete q;

예제4-7



예제4-8



객체배열의 동적 생성 및 변환 객체 배열의 동적 생성과 생성자 클래스이름 \*포인터변수 = new 클래스이름[배열크기]

ex) Circle\* pArray = new Circle[3];

Circle\* pArray = new Circle[3]{ Circle(1),Circle(2), Circle(3) };

객체 배열의 사용 보통 객체 배열처럼 사용

ex) Circle\* pArray = new Circle[3];

pArray[0].setRadius(10);

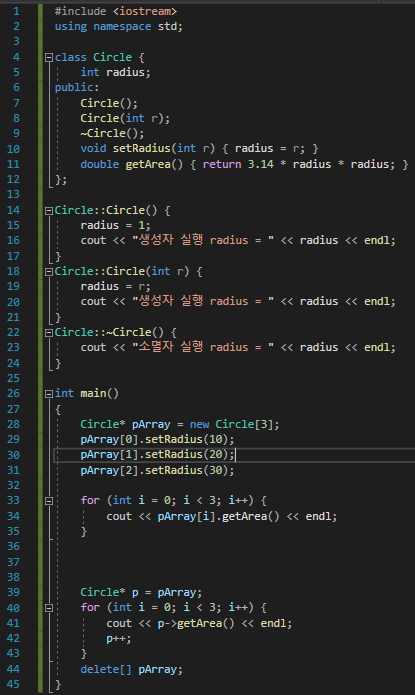
pArray[1].setRadius(20);

pArray[2].setRadius(30);

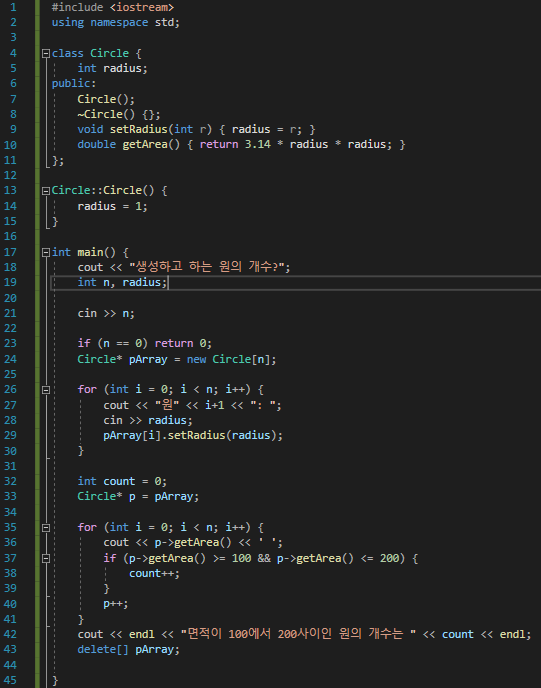
배열의 반환과 소멸자 delete [] 포인터변수

ex) delete[] pArray;

예제4-9



예제4-10



this 포인터

this의 기본 개념 객체 자신에 대한 포인터로서 클래스의 멤버함수 내에서만 사용됨

객체의 멤버함수가 호출될 때, 전달되는 객체에 대한 주소

ex) class Circle {

int radius;

public:

Circle() { this->radius = 1; }

Circle(int radius) { this->radius = radius; }

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

...

};

this와 객체 this는 객체 자신에 대한 포인터

각 객체 속의 this는 다른 객체 속의 this와 다름

컴파일러의 this 변환 멤버함수의 매개변수 void setRadius(int r) {this->radius = r;} 🡪 void setRadius(Circle\* this, int r) {this->radius = r;}

멤버함수 호출 p.setRadius(4); 🡪 p.setRadius(&p, 4);

this가 필요한 경우 1. 멤버변수와 동일한 이름으로 매개변수이름을 짓고자 하는 경우 ex) Circle(int radius) { this->radius = radius; }

//멤버 radius = 매개변수 radius

2. 객체멤버 함수에서 객체 자신의 주소를 리턴할 때 ex) class Sample

{

public:

Sample\* f() {

...;

return this;

}

};

this의 제약 조건 클래스의 멤버함수에서만 사용가능 객체에 속하지 않는 함수들에서는 사용 불가

정적 멤버함수에선 사용불가 정적멤버함수는 객체가 생성되기 전에 호출될 수 있음

check time 1. 3

string 클래스를 이용한 문자열 사용

string 클래스 개요 C-스트링 C언어에서 사용해오던 전통적인 문자열

널문자로 끝나는 클래스

메모리 크기 이상의 문자열

string 클래스

#include <string> 필요

스스로 메모리 크기를 조절함 🡪 사용하기 매우 편리

string 객체 생성 및 출력 string 객체 생성 다양한 생성자 string(); 빈 문자열을 가진 스트링 객체 생성

string(const string& str); str을 복사한 스트링 객체 생성

string(const char\* s); C-스트링 s의 문자열을 복사한 스트링 객체 생성

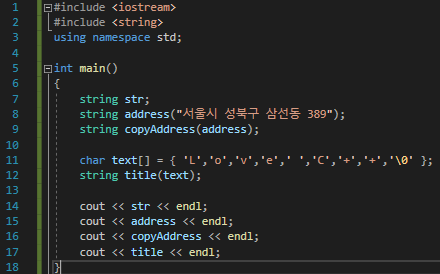
string(const char\* s, int n); 문자열s에서 n개의 문자를 복사한 스트링 객체 생성

문자열 크기 제한X

string 객체가 가진 문자열 출력 cout << string 객체

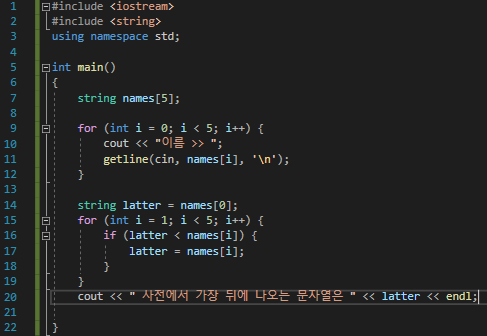
string 객체의 동적 생성 new, delete 이용

예제4-11



string 객체에 문자열 입력 공백문자 제외 cin>>string객체

공백문자 포함 getline(cin, string객체, ‘\n’)

예제4-12 

문자열 다루기 문자열 치환 =연산자 이용 ex) string a = "Java", b = "C++";

a=b; //a="C++"이 된다. a는 b를 복사한 문자열을 가진다

문자열 비교 int compare(const string& str) 리턴 두 문자열이 같으면 0

str보다 사전 순으로 앞에 오면 음수

str보다 사전 순으로 뒤에 오면 양수

ex) string name = "kitae";

string alias = "kito";

name.compare(alias) //음수 리턴

비교연산자(>, <, ==) 사전 순을 비교할 때는 비교연산자를 이용하는 게 좋음

== 동일 여부 확인

>,< 사전 순으로 먼저오는 지 확인

크면 늦게 옴

문자열 연결 string& append(const string& str) 문자열 끝에 str연결

ex) string a("I");

a.append(" love "); //a="I love "

+, - 연산자

문자열 삽입 string& insert(int pos, const string& str) 인덱스 pos위치에 str문자열 삽입

ex) string a("I love C++");

a.insert(2, "really "); //a="I really love C++"

string& replace(int pos, int n, const string& str) 인덱스 pos위치부터 n개의 문자를 str로 대체

ex) string a("I really love C++");

a.replace(2, 11, "study"); //a="I study C++"

문자열 길이 문자열에 포함된 문자 개수

int size() 리턴 문자열의 길이

ex) string a("I study C++");

int length = a.length(); //length = 11

int length() 리턴 문자열의 길이

ex) string a("I study C++");

int size = a.size(); //size = 11

int capacity() 리턴 할당된 메모리의 크기

ex) string a("I study C++");

int capacity = a.capacity(); //변할 수 있음

문자열 삭제 string& erase(int pos, int n) 인덱스 pos부터 n개의 문자열 삭제

ex) string a("I love C++");

a.erase(0, 7); //a[0]부터 a[6]까지 문자 삭제 a="C++"

void clear() 문자열 완전 삭제

ex) string a("I love C++");

a.clear(); //문자열 삭제 a=""

서브스트링 string substr(int pos, int n) pos(생략 시, 0부터)부터 n까지의 문자열 발췌

리턴 pos부터 n까지를 발췌한 문자열

발췌당하는 문자열은 변함없음

ex) string a("I love C++");

string b = a.substr(2, 4); //b="love", a="I love C++"

string c = a.substr(2); //c="love C++"

문자열 검색 int find(const string& str, int pos) 인덱스 pos(생략 시, 0부터)부터 문자열에서 str를 찾음

리턴 특정 문자, 문자열을 찾으면 첫 번째 인덱스

찾지 못하면 -1

ex) string a("I love love C++");

int index1 = a.find("love"); //index1=2

int index2 = a.find("C#"); //index2=-1

int index3 = a.find('v', 7); //index3=9

문자열의 각 문자 다루기 char& at(int pos) 문자 수정 불가

리턴 인덱스 pos의 문자

ex) string a("I love C++");

char ch1 = a.at(7);

[] 연산자 문자 수정 가능

리턴 [인덱스]의 문자

ex) string a("I love C++");

char ch2 = a[7];

char ch3 = a[a.length() - 1];

a[7] = 'D'; //a="I love D++"

문자열의 숫자 변환, stoi() 문자열을 숫자로 변환하는 함수

문자열이 숫자가 아니라면 컴파일 오류

리턴 숫자가 된 숫자로 이루어진 문자열

ex) string year = "2014";

int n = stoi(year); //n=2014

문자 다루기 <locale>헤더파일 include

toupper(char character) 리턴 character의 대문자값

isdisit(char character) character가 숫자인지 판별

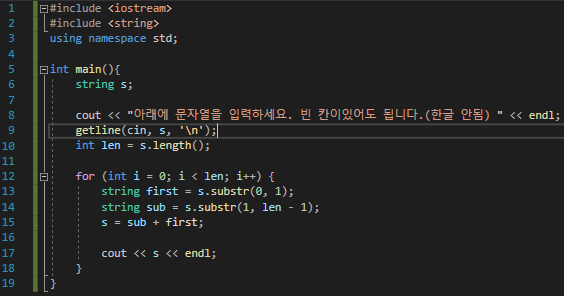
리턴 숫자이면 true

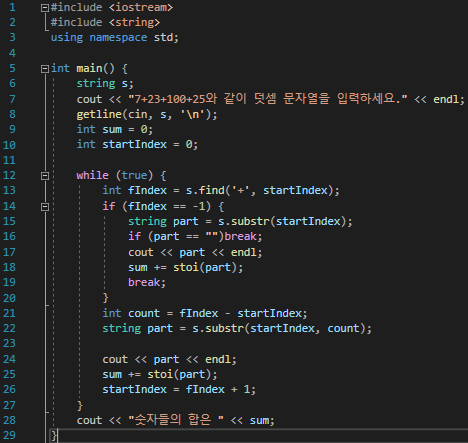
숫자가 아니면 false

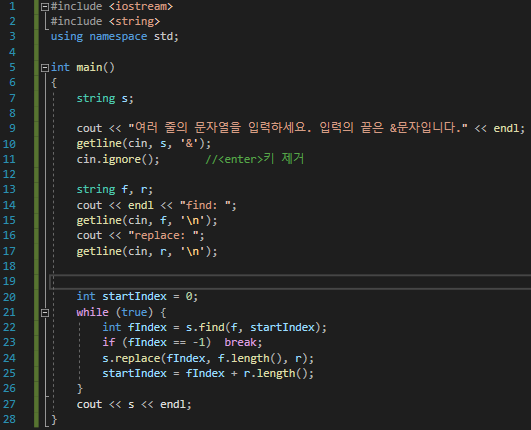
isalpha(char character) character가 알파벳인지 판별

리턴 알파벳이면 true

알파벳이 아니면 false

예제4-13 

예제4-14 

예제4-15 

check time 1. 9

Hello C++!!

C

-1

Ho C++!!