5.1 함수의 인자 전달 방식 리뷰

인자 전달 방식 값에 의한 호출 호출한 코드에서 넘겨주는 실인자 값이 함수의 매개변수에 복사되어 전달되는 방식

함수 내에서 실인자 값 손상 불가

매개변수에 복사되는 시간 소요

주소에 의한 호출 참조에 의한 호출이 아님

주소를 직접 포인터 타입의 매개 변수에 전달받는 방식

ex) 함수 호출 시, 배열이 전달되는 경우: 배열의 이름 = 배열의 주소

의도적으로 함수 내에서 실인자의 값을 변경하고자 할 때 사용

check time 1. 값, 복사, 공유하지 않는, 포인터 타입, 주소, 변경

5.2 함수 호출 시 객체 전달

값에 의한 호출로 객체 전달 값에 의한 호출 과정 복사당하는(매개변수로 주어지는) 객체는 변경되지 않음

ex) void increase(Circle c) {

int r = c.getRadius(); // waffle이 c로 복사되기 때문에, r = 30

c.setRadius(r + 1); //waffle의 반지름은 변경되지 않음

}

int main() {

Circle waffle(30);

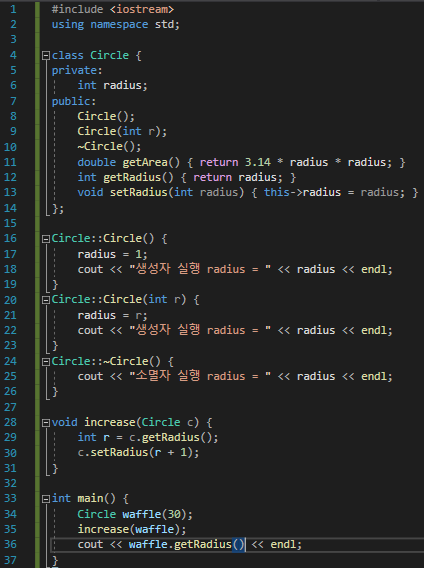
increase(waffle);

}

값에 의한 호출로 객체를 전달할 때의 문제점 함수의 매개변수 객체의 생성자는 실행되도록 컴파일 X

함수의 매개변수 객체의 소멸자는 실행되도록 컴파일 O

예제 5-1



왜 매개변수객체의 생성자는 실행되도록 컴파일되는가? 객체를 복사받음 🡪 생성자 실행 🡪 복사받은상태를 잃어버림

중대한 문제 발생 🡪 해결: 복사생성자(*5.5절로*)

주소에 의한 호출로 객체 전달 주소에 의한 호출 과정 void increase(Circle \*c) { //생성되는 것이 객체가 아닌 포인터 변수이기 때문에 생성자 상관 X

int r = c->getRadius();

c->setRadius(r + 1);

}

int main() {

Circle waffle(30);

increase(&waffle); //waffle의 주소 전달

cout << waffle.getRadius(); //31

}

주소에 의한 호출의 특징 복사하는 시간 소요X

생성자∙소멸자 비대칭 문제X

의도치 않게 원본객체를 손상시킬 가능성이 있음

5.3 객체 치환 및 객체 리턴

객체치환 객체 치환시 객체의 모든 데이터가 비트 단위로 복사됨

동일한 클래스 타입에 대해서만 적용

ex) Circle c1(5);

Circle c2(30);

c1 = c2; //c1의 반지름이 30이됨

c1, c2 하나의 객체가 되는 것은 아님

내용물만 같음

함수의 객체 리턴 함수가 객체를 리턴하는 경우

ex) Circle getCircle() {

Circle tmp(30);

return tmp; //return문이 실행되면 tmp의 복사본이 생기고 복사본이 return됨, tmp의 원본은 소멸

}

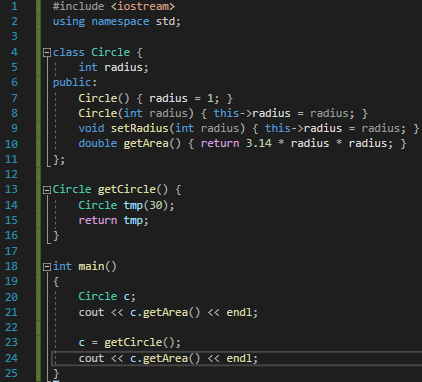
int main() {

Circle c1(1);

c1 = getCircle(); //tmp객체의 복사본이 c에 치환됨

}

예제5-2



5.4 참조와 함수

참조 개념 C언어에는 없는 참조 개념 도입

참조변수 선언에는 &이용

& 참조자

이미 선언된 변수에 대한 별명(alias)

참조 변수 참조변수 선언 참조변수는 이미 선언된 변수(원본변수)에 대한 별명

&를 이용해 선언

선언시 반드시 원본변수로 초기화

ex) int n = 2;

int& refn = n;

Circle circle;

Circle& refc = circle;

참조변수는 다른 변수 공간을 할당받지 않고, 원본변수의 메모리 공간을 공유함

참조변수 사용 보통 변수와 사용방법 동일

참조변수의 사용은 바로 원본 변수의 사용

참조변수에 대한 포인터를 만들 수 있음 = 원본변수에 대한 포인터

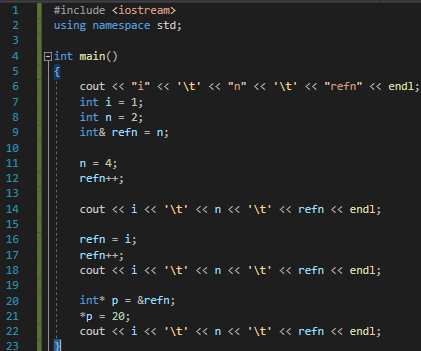
참조변수 선언 시 주의사항 초기화가 없다면 컴파일 오류가 발생한다

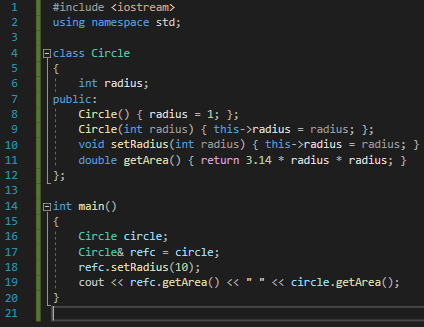
&의 위치는 무관하다(포인터 선언시의 \*과 동일)

참조자 &의 사용에 유의해야 한다

참조변수의 배열을 만들 수 없다

참조변수에 대한 참조선언이 가능하다, 이 경우 2개의 참조변수는 원본변수를 참조한다

예제5-3 

예제5-4 

check time 1. (1) 1

1. (2) Sample& x=a;

Sample\* y= &a;

x.show();

y->show();

참조에 의한 호출 call by reference

매개변수를 참조타입으로 선언 매개변수가 함수를 호출하는 쪽의 실인자 참조 🡪 실인자와 공간 공유

참조 매개 변수

ex) void swap(int& a,int& b) {

int tmp;

tmp = a;

a = b;

b = tmp;

}

int main() {

int m = 2, n = 9;

swap(m, n);

cout << m << ' ' << n;

}

참조 매개 변수가 필요한 사례 ex) int average(int a[], int size) {

if (size <= 0) return 0;

int sum = 0;

for (int i = 0; i < size; i++) sum += a[i];

return sum / size;

}

int main() {

int x[] = { 1,2,3,4 };

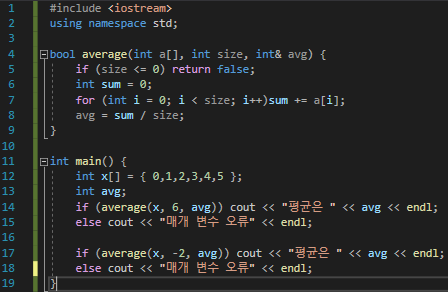
int avg = average(x, -1);

cout << avg;

}

size가 잘못되면 0 return 🡪 평균이 0이라고 착각할 수 있음 🡪 예제5-5에서 해결

예제5-5



참조에 의한 호출의 장점 주소의 의한 호출 포인터로 인한 실수의 가능성, 긴장감, 가독성 문제

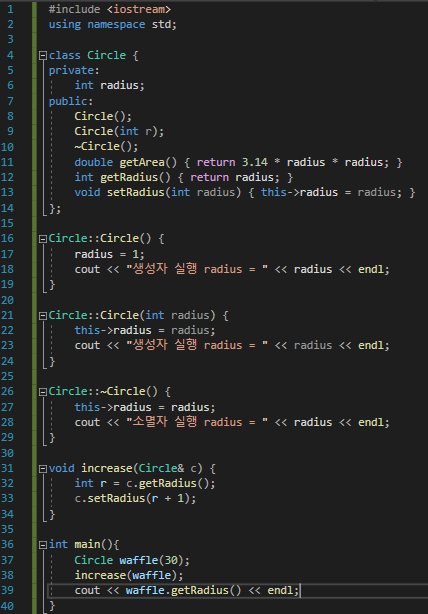
참조의 의함 호출 쉽고 보기 좋은 코드

참조에 의한 호출로 객체 전달 값에 의한 호출로 객체를 매개변수에 전달할 때 1. 함수 내에서 매개변수객체를 변경해도, 원본객체 변경 X

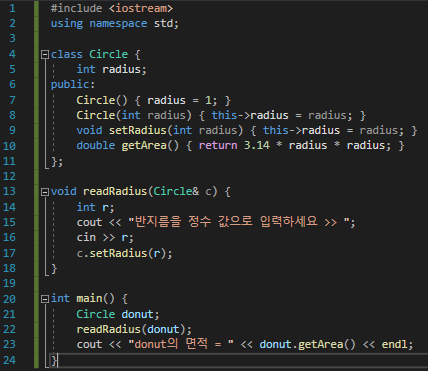
2. 생성자 실행 X, 소멸자만 실행됨

참조에 의한 호출로 객체를 매개변수에 전달할 때 1. 참조매개변수로 이루어진 모든 연산은, 원본 객체에 대한 연산

2. 생성자 실행 X, 소멸자 실행 X

예제5-6 

예제5-7



참조리턴 함수가 참조를 리턴하는 것

변수등과 같이 현존하는 공간에 대한 참조의 리턴

ex) char c = 'a';

char& find() {

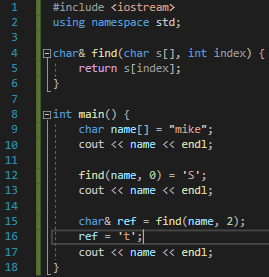
return c;

}

참조리턴에 대한 치환문 ex) char a = find(); //a='a'

char& ref = find(); //ref가 c의 참조가 됨

예제5-8



check time 1. 1

2. (1) 5

2. (2) -5

2. (3) -5

3. (1) [100,1,3,5,7]

3. (2) [16,1,3,5,7]

3. (3) [0,1,4,5,7]

5.5 복사생성자

얕은복사와 깊은복사 얕은복사 충돌발생

깊은복사 충돌발생X

동적할당배열을 가진 객체의 얕은 복사 복사된 객체의 동적할당배열이 원본 배열의 동적할당배열이 가리키는 메모리를 가리킴

= 포인터가 복사됨

= 사본객체에서 동적할당배열을 수정하면, 원본객체의 동적할당배열도 수정됨

동적할당배열을 가진 객체의 깊은 복사 포인터가 기리키는 메모리까지 복사

= 메모리가 복사됨

복사생성 및 복사생성자(얕은복사) 복사생성자 객체가 생성될 때, 원본객체를 복사하여 생성되는 경우

복사생성자 선언 ex) class ClassName {

ClassName(const ClassName& c);

};

복사생성자는 클래스에 1개만 선언가능

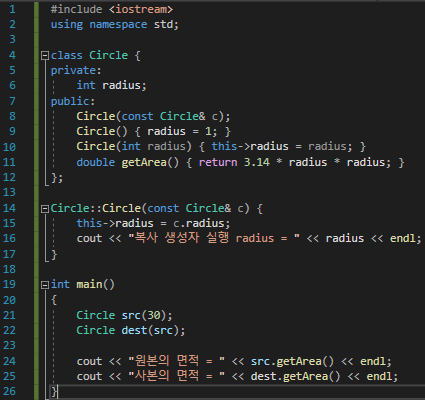
복사생성자의 매개변수는 1개 🡨 자기클래스에 대한 참조

복사생성자 실행 Circle src(30);

Circle dest(src);

컴파일러가 복사 생성자 호출하도록 컴파일

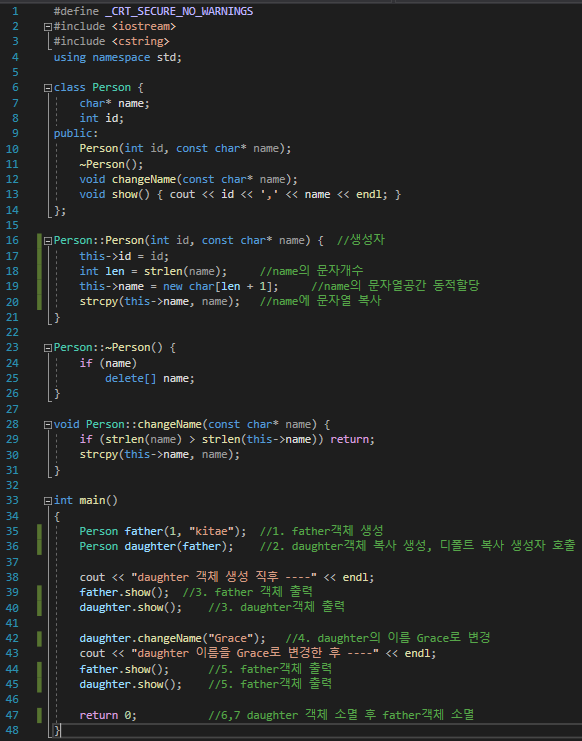
예제5-9



디폴트 복사 생성자 복사생성자를 선언하지 않아도 얕은 복사 수행

원본 객체의 모든 멤버변수를 사본객체에 복사

얕은 복사 생성자의 문제점 포인터타입의 멤버변수가 없는 경우엔 얕은 복사해도 공유의 문제 X

예제5-10 

실행 디폴트 복사 생성자 자동 삽입 Person::Person(const Person& p) {

this->id = p.id;

this->name = p.name;

}

main함수 실행 1. father객체 생성

2. father를 복사한 daughter객체 생성

3. father와 daughter 객체 출력 father.name과 daughter.name은 동일한 메모리를 가리킴 🡪 동일한 메모리 출력

4. daughter 객체 name 변경

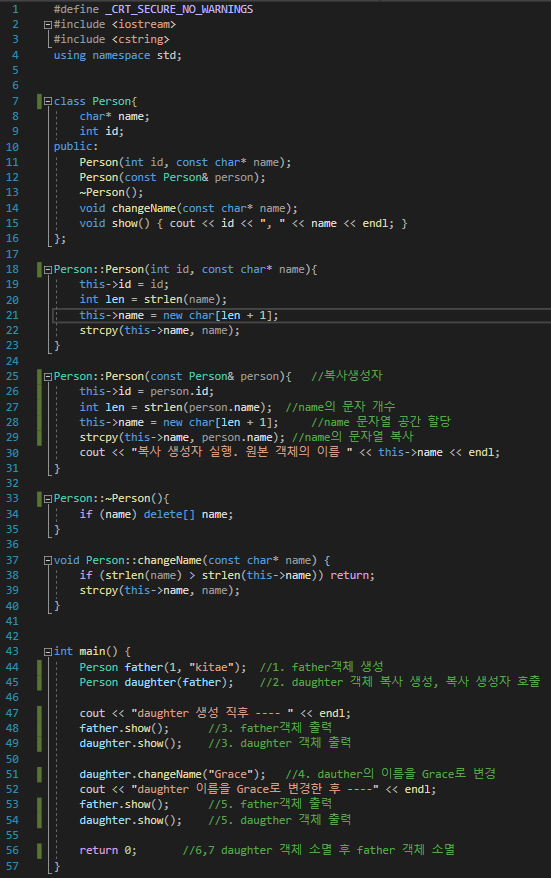
5. father와 daughter 객체 출력 father.name과 daughter.name은 동일한 메모리를 가리킴 🡪 father.name도 변경됨

6. daughter 객체 소멸 daughter.name 메모리 힙에 반환

7. father 객체 소멸 father.name 메모리를 힙에 반환해야 하나, daughter.name 메모리가 반환되어 반환 불가

🡪 실행시간 오류 발생 🡪 비정상 종료

사용자 복사 생성자 작성 깊은 복사 생성자 만들기

예제 5-11 

실행 1. father 객체 생성

2. father를 복사한 daughter 객체 생성 Person::Person(const Person& person){ //복사생성자

this->id = person.id;

int len = strlen(person.name); //name의 문자 개수

this->name = new char[len + 1]; //name 문자열 공간 할당

strcpy(this->name, person.name); //name의 문자열 복사

cout << "복사 생성자 실행. 원본 객체의 이름 " << this->name << endl;

}

실행결과, daughter.name에 메모리가 따로 동적할당되고 문자열이 복사됨

3. father와 daughter 객체 출력

4. daughter 객체의 이름 변경

5. father와 daughter 객체 출력

6. daughter 객체 소멸 daughter.name 동적배열 반환

7. father 객체 소멸 father.name 동적배열 반환

묵시적 복사 생성 컴파일러가 복사 생성자를 자동으로 호출하는 경우 1. 객체로 초기화하여 객체가 생성될 때 ex) Person father(1, "kitae");

Person son = father;

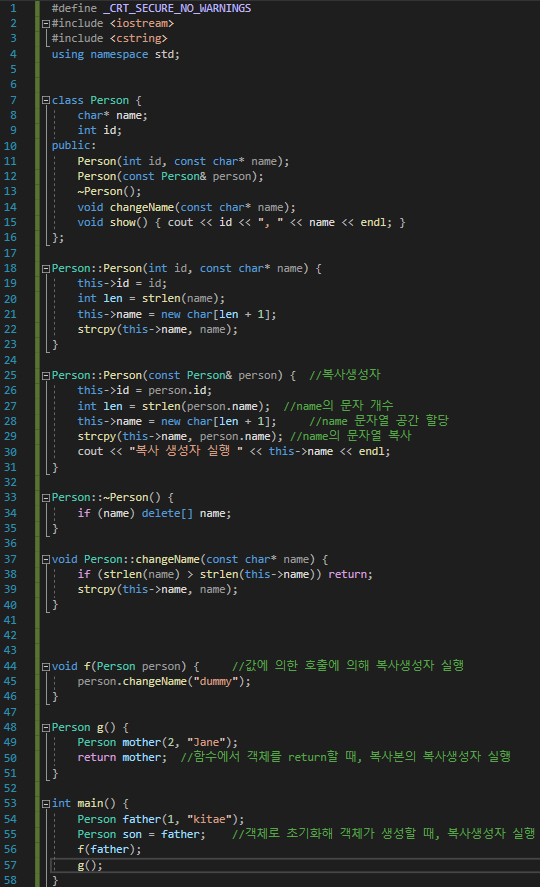
치환문과 헷갈리지 말 것 ex) Person son

son = father;

2. 값에 의한 호출로 객체가 전달될 때 함수의 매개변수 객체가 생성될 때 복사 생성자가 실행됨

생성자는 실행안되지만 복사생성자가 실행됨

3. 함수가 객체를 리턴할 때 return문이 리턴객체의 복사본을 생성하여 호출한 곳으로 전달

예제5-12 

ㄴㅇㄹ

ㅁㄴㅇㄹ