

06

함수 중복과 static 멤버

학습 목표

1. 함수 중복의 개념을 이해하고, 중복 함수를 작성할 수 있다.
2. 디폴트 매개 변수를 이해하고 작성할 수 있다.
3. 함수 중복 시 발생하는 모호성의 경우를 판별할 수 있다.
4. `static` 속성으로 선언된 멤버의 특성을 이해하고, `static` 속성을 활용할 수 있다.

함수 중복

3

□ 함수 중복

▣ 동일한 이름의 함수가 공존

- 다형성
- C 언어에서는 불가능

▣ function overloading

▣ 함수 중복이 가능한 범위

- 보통 함수들 사이
- 클래스의 멤버 함수들 사이
- 상속 관계에 있는 기본 클래스와 파생 클래스의 멤버 함수들 사이

□ 함수 중복 성공 조건

- ▣ 중복된 함수들의 이름 동일
- ▣ 중복된 함수들의 매개 변수 타입이 다르거나 개수가 달라야 함
- ▣ 리턴 타입은 함수 중복과 무관

함수 중복 성공 사례

4

```
int sum(int a, int b, int c) {  
    return a + b + c;  
}  
  
double sum(double a, double b) {  
    return a + b;  
}  
  
int sum(int a, int b) {  
    return a + b;  
}
```

성공적으로 중복된 sum() 함수들

```
int main(){  
    cout << sum(2, 5, 33);  
  
    cout << sum(12.5, 33.6);  
  
    cout << sum(2, 6);  
}
```

중복된 sum() 함수 호출.
컴파일러가 구분

함수 중복 실패 사례

5

- 리턴 타입이 다르다고 함수 중복이 성공하지 않는다.

```
int sum(int a, int b) {  
    return a + b;  
}  
double sum(int a, int b) {  
    return (double)(a + b);  
}
```

함수 중복 실패

```
int main() {  
    cout << sum(2, 5);  
}
```

컴파일러는 어떤 sum() 함수를 호출하는지 구분할 수 없음

함수 중복의 편리함

6

- 동일한 이름을 사용하면 함수 이름을 구분하여 기억할 필요 없고, 함수 호출을 잘못하는 실수를 줄일 수 있음

```
void msg1() {  
    cout << "Hello";  
}  
void msg2(string name) {  
    cout << "Hello, " << name;  
}  
void msg3(int id, string name) {  
    cout << "Hello, " << id << " " << name;  
}
```

(a) 함수 중복하지 않는 경우



```
void msg() {  
    cout << "Hello";  
}  
void msg(string name) {  
    cout << "Hello, " << name;  
}  
void msg(int id, string name) {  
    cout << "Hello, " << id << " " << name;  
}
```

(b) 함수 중복한 경우

함수 중복하면 함수 호출의 편리함.
오류 가능성 줄임

예제 6-1 big() 함수 중복 연습

7

큰 수를 리턴하는 다음 두 개의 big 함수를 중복 구현하라.

```
int big(int a, int b);    // a와 b 중 큰 수 리턴
int big(int a[], int size); // 배열 a[]에서 가장 큰 수 리턴
```

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

```
int main() {
    int array[5] = {1, 9, -2, 8, 6};
    cout << big(2,3) << endl;
    cout << big(array, 5) << endl;
}
```

예제 6-2(실습) sum() 함수 중복 연습

8

함수 sum()을 호출하는 경우가 다음과 같을 때, 함수 sum()을 중복 구현하라. sum()의 첫 번째 매개 변수는 두 번째 매개 변수보다 작은 정수 값으로 호출된다고 가정한다.

```
sum(3,5); // 3~5까지의 합을 구하여 리턴
sum(3);   // 0~3까지의 합을 구하여 리턴
sum(100); // 0~100까지의 합을 구하여 리턴
```

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

```
int main() {
    cout << sum(3, 5) << endl;
    cout << sum(3) << endl;
    cout << sum(100) << endl;
}
```

12
6
5050

생성자 함수 중복

9

□ 생성자 함수 중복 가능

▣ 생성자 함수 중복 목적

- 객체 생성시, 매개 변수를 통해 다양한 형태의 초깃값 전달

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    ...  
};  
  
int main() {  
    Circle donut;           // Circle() 생성자 호출  
    Circle pizza(30);       // Circle(int r) 생성자 호출  
}
```

□ 소멸자 함수 중복 불가

▣ 소멸자는 매개 변수를 가지지 않음

▣ 한 클래스 내에서 소멸자는 오직 하나만 존재

string 클래스의 생성자 중복 사례

10

```
class string {  
    ....  
public:  
    string(); // 빈 문자열을 가진 스트링 객체 생성  
    string(string& str); // str을 복사한 새로운 스트링 객체 생성  
    string(char* s); // '₩0'로 끝나는 C-스트링 s를 스트링 객체로 생성  
    ....  
};
```

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

디폴트 매개 변수

11

- 디폴트 매개 변수(default parameter)
 - ▣ 매개 변수에 값이 넘어오지 않는 경우, 디폴트 값을 받도록 선언된 매개 변수
 - ‘매개 변수 = 디폴트값’ 형태로 선언

- 디폴트 매개 변수 선언 사례

```
void star(int a=5); // a의 디폴트 값은 5
```

- 디폴트 매개 변수를 가진 함수 호출

```
star(); // 매개 변수 a에 디폴트 값 5가 전달됨. star(5);와 동일  
star(10); // 매개 변수 a에 10을 넘겨줌
```

디폴트 매개 변수 사례

12

□ 사례 1

```
void msg(int id, string text="Hello"); // text의 디폴트 값은 "Hello"
```

```
msg(10); // msg(10, "Hello"); 호출과 동일. id에 10, text에 "Hello" 전달
```

```
msg(20, "Good Morning"); // id에 20, text에 "Good Morning" 전달
```

```
msg(); // 컴파일 오류. 첫 번째 매개 변수 id에 반드시 값을 전달하여야 함
```

호출 오류

```
msg("Hello"); // 컴파일 오류. 첫 번째 매개 변수 id에 값이 전달되지 않았음
```

□ 디폴트 매개 변수는 보통 매개 변수 앞에 선언될 수 없음

■ 디폴트 매개 변수는 끝 쪽에 몰려 선언되어야 함

컴파일 오류

```
void calc(int a, int b=5, int c, int d=0); // 컴파일 오류
```

```
void sum(int a=0, int b, int c); // 컴파일 오류
```

```
void calc(int a, int b=5, int c=0, int d=0); // 컴파일 성공
```

매개 변수에 값을 정하는 규칙

13

□ 사례 2

```
void square(int width=1, int height=1);
```

디폴트 매개 변수를
가진 square()

```
void square(int width=1, int height=1);
```

| | | | | |
|----------------------|---|----------------------------------|---|--------------------------------|
| square(); | → | square(<u> </u> , <u> </u>); | → | square(1 , 1); |
| square(5); | → | square(5, <u> </u>); | → | square(5, 1); |
| square(3, 8); | → | square(3, 8); | → | square(3, 8); |

컴파일러에 의해
변환되는 과정

디폴트 매개 변수 사례

14

□ 사례 3

```
void g(int a, int b=0, int c=0, int d=0);
```

디폴트 매개 변수를
가진 함수

```
void g(int a, int b=0, int c=0, int d=0);
```

| | | | | |
|--------------------------|---|---|---|---------------------|
| g(10); | → | g(10, <u> </u> , <u> </u> , <u> </u>); | → | g(10, 0, 0, 0); |
| g(10, 5); | → | g(10, 5 , <u> </u> , <u> </u>); | → | g(10, 5, 0, 0); |
| g(10, 5, 20); | → | g(10, 5, 20, <u> </u>); | → | g(10, 5, 20, 0); |
| g(10, 5, 20, 30); | → | g(10, 5, 20, 30); | → | g(10, 5, 20, 30); |

컴파일러에 의해
변환되는 과정

예제 6-3 디폴트 매개 변수를 가진 함수 선언 및 호출

15

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

```
// 원형 선언
```

```
void star(int a=5);
void msg(int id, string text="");
```

디폴트
매개 변수 선언

```
// 함수 구현
```

```
void star(int a) {
    for(int i=0; i<a; i++)
        cout << '*';
    cout << endl;
}
```

```
void msg(int id, string text) {
    cout << id << ' ' << text << endl;
}
```

동일한
코드

```
void star(int a=5) {
    for(int i=0; i<a; i++)
        cout << '*';
    cout << endl;
}
```

```
void msg(int id, string text="") {
    cout << id << ' ' << text << endl;
}
```

```
int main() {
    // star() 호출
    star();
    star(10);
```

star(5);

```
    // msg() 호출
    msg(10);
    msg(10, "Hello");
}
```

msg(10, "");

```
*****
*****
10
10 Hello
```

예제 6-4(실습) 디폴트 매개 변수를 가진 함수 만들기 연습

16

함수 f()를 호출하는 경우가 다음과 같을 때 f()를 디폴트 매개 변수를 가진 함수로 작성하라.

빈 칸이 10개 출력됨

```
%%%%%%%%%  
aaaaaaaaa  
aaaaaaaaa  
aaaaaaaaa  
aaaaaaaaa  
aaaaaaaaa  
aaaaaaaaa
```

```
f(); // 한 줄에 빈칸을 10개 출력한다.  
f('%'); // 한 줄에 '%'를 10개 출력한다.  
f('@', 5); // 다섯 줄에 '@'를 10개 출력한다.
```

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

```
int main() {  
    f(); // 한줄에 빈칸을 10개 출력한다.  
    f('%'); // 한 줄에 '%'를 10개 출력한다.  
    f('@', 5); // 5 줄에 '@' 문자를 10개 출력한다.  
}
```


함수 중복 간소화

17

▣ 디폴트 매개 변수의 장점 - 함수 중복 간소화

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    .....  
};
```

```
class Circle {  
    .....  
public:  
    Circle(int r=1) { radius = r; }  
    .....  
};
```

2 개의 생성자 함수를
디폴트 매개 변수를 가진
하나의 함수로 간소화

▣ 중복 함수들과 디폴트 매개 변수를 가진 함수를 함께 사용 불가

```
class Circle {  
    .....  
public:  
    Circle() { radius = 1; }  
    Circle(int r) { radius = r; }  
    Circle(int r=1) { radius = r; }  
    .....  
};
```

중복된 함수와
동시 사용 불가

예제 6-5(실습) 디폴트 매개 변수를 이용하여 중복 함수 간소화 연습

18

다음 두 개의 중복 함수를 디폴트 매개 변수를 가진 하나의 함수로 작성하라.

```
void fillLine() { // 25 개의 '*' 문자를 한 라인에 출력
    for(int i=0; i<25; i++) cout << '*';
    cout << endl;
}
void fillLine(int n, char c) { // n개의 c 문자를 한 라인에 출력
    for(int i=0; i<n; i++) cout << c;
    cout << endl;
}
```

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

void fillLine(int n=25, char c='*') { // n개의 c 문자를 한 라인에 출력
    for(int i=0; i<n; i++) cout << c;
    cout << endl;
}

int main() {
    fillLine(); // 25개의 '*'를 한 라인에 출력
    fillLine(10, '%'); // 10개의 '%'를 한 라인에 출력
}
```

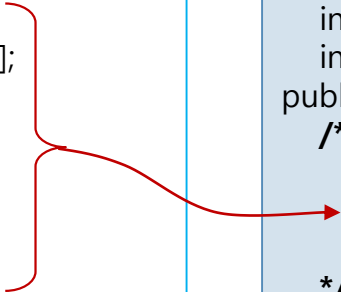
```
*****
%%%%%%%%%
```

예제 6-6(실습) 생성자 함수의 중복 간소화

19

다음 클래스에 중복된 생성자를 디폴트 매개 변수를 가진 하나의 생성자로 작성하라.

```
class MyVector{
    int *p;
    int size;
public:
    MyVector() {
        p = new int [100];
        size = 100;
    }
    MyVector(int n) {
        p = new int [n];
        size = n;
    }
    ~MyVector() { delete [] p; }
};
```



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

```
class MyVector{
    int *p;
    int size;
public:
    /*
```

이곳에 디폴트 매개변수를 가진 생성자 작성하라

```
*/
    ~MyVector() { delete [] p; }
};
```

```
int main() {
    MyVector *v1, *v2;
    v1 = new MyVector(); // 디폴트로 정수 100개의 배열 동적 할당
    v2 = new MyVector(1024); // 정수 1024개의 배열 동적 할당

    delete v1;
    delete v2;
}
```