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(function overloading)

- 동일한 이름의 함수를 여러 개 만들 수 있는 기능
- 다형성의 한 사례, C언어에서는 불가능
- 함수 중복이 가능한 범위
  - 멤버함수, 전역함수(비멤버함수)에 모두 적용가능
  - 상속 관계에 있는 기본 클래스와 파생 클래스의 멤버 함수들 사이에 도 가능
- 함수 중복 조건
  - 함수들의 이름 동일
  - 함수들의 매개 변수 타입이 다르거나 개수가 달라야 함
  - 리턴 타입(반환형)은 함수 중복과 무관 -> 이름과 매개변수가 모두 같고 반환형이 다른 함수는 허용 안됨
  - C언어는 이름만으로 함수를 구분하지만 C++는 이름과 매개변수로 함수를 구분

함수 중복 성공 사례

- 인수의 형태에 따라 중복된 함수를 구분해줌.

```
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;
}
```

```
int main() {
    cout << sum(2, 5, 33);

    cout << sum(12.5, 33.6);

    cout << sum(2, 6);
}
```

중복된 sum() 함수 호출.
컴파일러가 구분

성공적으로 중복된 sum() 함수들
-> 이름은 같지만 매개변수가 다르므로
서로 다른 함수로 구분됨

함수 중복 실패 사례

- 리턴 타입이 다르다고 함수 중복이 허용되지 않음 -> 함수의 구분은 함수이름과 매개변수로 결정

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

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

?

컴파일러는 어떤 sum() 함수를 호출하는지 구분할 수 없음

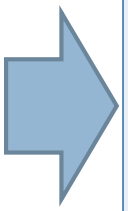
함수 중복 실패

함수 중복의 편리함

- 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

예제 6-1 big() 함수 중복 연습

- 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

// 배열에 저장된 데이터를 매개변수로 받을 때는 포인터로 변환됨
// 함수 호출시는 배열의 시작주소(배열명)를 인수로 전달함
int big(int a[], int size); -> int big(int* a, int size);
```

예제 6-1 big() 함수 중복 연습


```
#include <iostream>
using namespace std;
int big(int a, int b) {                    // a와 b 중 큰 수 리턴
	if (a > b) return a;
	else return b;
}
int big(int a[], int size) {               // int big(int* a, int size)
	int res = a[0];
	for (int i = 1; i < size; i++)
		if (res < a[i]) res = a[i];
	return res;
}
int main() {
	int array[5] = { 1, 9, -2, 8, 6 };
	cout << big(2, 3) << endl;
	cout << big(array, 5) << endl;
}
```

```
3
9
```

예제 6-2(실습) sum() 함수 중복 연습

- 함수 sum()을 호출하는 경우가 다음과 같을 때, 함수 sum()을 중복 구현하라. sum()의 첫 번째 매개 변수는 두 번째 매개변수보다 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
```

예제 6-2(실습) sum() 함수 중복 연습


```
#include <iostream>
using namespace std;
int sum(int a, int b) {                    // a에서 b까지 합하기
	int s = 0;
	for (int i = a; i <= b; i++) s += i;
	return s;
}
int sum(int a) {                           // 0에서 a까지 합하기
	int s = 0;
	for (int i = 0; i <= a; i++) s += i;
	return s;
}
int main() {
	cout << sum(3, 5) << endl;
	cout << sum(3) << endl;
	cout << sum(100) << endl;
}
```

```
12
6
5050
```

생성자 함수 중복

- 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
}
```

string 클래스의 생성자 중복 사례


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

```
string str;                                 // 빈 문자열을 가진 스트링 객체
string address("서울시 성북구 삼선동 389");
string copyAddress(address);                // string객체로 초기화한 string객체생성
```

소멸자 함수 중복

- 소멸자 함수 중복 불가
  - 소멸자는 매개 변수를 가지지 않음
  - 한 클래스 내에서 소멸자는 오직 하나만 존재

실습과제1

- C언어와 C++언어에서 컴파일러가 호출할 함수를 결정하는 방식의 차이를 설명하라.

실습과제2

- 아래 실행 결과가 나오도록 중복된 함수 3개를 정의하라.
- 힌트: 매개변수를 각각 정수, 실수, string 타입으로 정의

```
큰 정수값은 20
큰 실수값은 3.14
사전에서 뒤에 나오는 단어는 world
```

```
#include <iostream>
#include <string>
using namespace std;
// 중복된 함수선언
int main() {
	int x = big(10, 20);
	cout << "큰 정수값은"<< x << endl;
	double y = big(3.14, 1.05);
	cout << "큰 실수값은"<< y << endl;
	string z = big("hello", "world");
	cout << "사전에서 뒤에 나오는 단어는"<< z << endl;
	return 0;
}
// 중복된 함수정의
```

실습과제3

- 아래 실행 결과가 나오도록 클래스 Rectangle를 정의하라. 생성자는 3개를 선언하라.
- 힌트: 멤버변수는 폭과 높이, 생성자는 3개, 기본생성자는 폭=높이=1, 멤버함수 show는 정보출력

```
// 클래스 선언 및 구현
int main() {
	Rectangle rect0;
	rect0.show();
	Rectangle rect1(10);
	rect1.show();
	Rectangle rect2(10, 20);
	rect2.show();
	return 0;
}
```

```
사각형 폭은 1 높이는 1
사각형 폭은 10 높이는 1
사각형 폭은 10 높이는 20
```

실습과제4

- 아래 실행 결과가 나오도록 클래스 Point3D를 정의하라. 생성자는 4개를 선언하라.
- 힌트: 멤버변수는 3개의 좌표, 생성자는 4개, 기본생성자는 3개 좌표 =0 , 멤버함수 show는 멤버변수 정보출력

```
// 클래스 선언 및 구현
int main() {
	Point3D p0;
	p0.show();
	Point3D p1(1);
	p1.show();
	Point3D p2(1, 2);
	p2.show();
	Point3D p3(1, 2, 3);
	p3.show();
	return 0;
}
```

```
3차원 점의  좌표는 (0,0,0)
3차원 점의  좌표는 (1,0,0)
3차원 점의  좌표는 (1,2,0)
3차원 점의  좌표는 (1,2,3)
```

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ************************************************
//    제   목 :  정수 4개의 평균을 구하는 프로그램
//    날   짜 :  2023년 9월10일
//    작성자 :  15010101 홍길동
// ************************************************

// 소스코드 작성
```