05

함수와 참조, 복사 생성자

학습 목표

1. 복사생성자의 필요성과 활용을 이해하고, 작성할 수 있다.

묵시적 복사 생성

- 명시적 복사 생성
  - 같은 클래스형의 객체를 생성자에 인자로 전달할 때

```
Circle src(10);          // 생성자
Circle dest(src);        // 복사 생성자를 명시적 호출
```

- 묵시적 복사 생성
  - 컴파일러가 복사 생성자를 자동으로 호출
  - 객체가 생성과 동시에 같은 자료형의 객체로 초기화 되는 경우
    - 객체를 생성과 동시에 같은 자료형의 객체로 초기화할 때
    - 값에 의한 호출 방식으로 객체를 함수에 전달할 때
    - 함수가 객체를 반환할 때

메모리 할당과 동시에 초기화 하는 경우

- 변수 선언과 동시에 초기값 대입

```
int num1 = num2;        // 메모리 할당과 초기화
```

```
int num1;               // 메모리 할당
...
num1 = num2;            // 대입연산
```

메모리 할당과 동시에 초기화 하는 경우

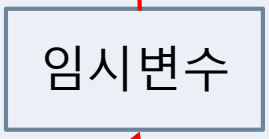
- 함수 호출시 매개변수 할당과 인자로 초기화

```
int main(void)
{
      int num = 10, result;
      result = simplefn(num);
}

int simplefn(int n) // 매개변수 할당과 인자로 초기화
{
      …                          int n = num;
}
```

메모리 할당과 동시에 초기화 하는 경우

- 함수가 리턴할 때 임시변수가 할당되고 리턴값으로 초기화 -> 함수 호출문장은 임시변수(반환값)로 치환됨

```
int main(void)
{
    int num = 10, result;
    result = simplefn(num);      // result = 임시변수;
}

int simplefn(int n)
{
    int m;
    ...
    return m;
}
```

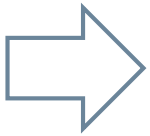



```
// 임시변수 할당과 반환값으로 초기화
int 임시변수 = m;
```

객체로 초기화하여 객체를 생성하는 경우

- 같은 클래스의 객체로 초기화 하는 경우 복사 생성자를 호출
  -> 초기값을 복사 생성자의 인자로 전달됨

```
Person father(1, "Kitae");
Person daughter = father;
```



```
Person father(1, "Kitae");
Person daughter(father);
```

- 객체 생성 후 따로 대입하는 경우와 다름

```
Person father(1, "Kitae");
Person son;                 // 매개변수 없는 생성자 실행
son = father;               // 대입연산자 실행
```

값에 의한 호출로 객체가 전달될 경우

- 함수의 매개변수 객체가 생성되고 인자로 초기화 -> 생성자 대신에 복사 생성자가 호출됨

```
int main() {
    Person father(1, "Kitae");
    fun(father);
    ...
    return 0;
}
```

```
void fun(Person person)
{
    ....    Person person = father;
}           => Person person(father);
```

객체가 생성과 동시에 같은 클래스의 객체로 초기화 되는 경우

함수가 객체를 리턴할 경우

- 함수가 객체를 리턴할때 임시객체가 생성되고 반환 객체로 초기화 -> 여기서 복사생성자가 실행됨

```
int main() {
    Person father;
    father = get();
    ...
}
```

```
Person get() {
    Person mother(2, "Jane");
    return mother;
}
```

임시객체

Person 임시객체 = mother;
=> Person 임시객체(mother);

객체가 생성과 동시에 같은 클래스의 객체로 초기화 되는 경우

묵시적으로 복사 생성자를 호출하는 이유

- 객체생성시 생성자 호출 + 객체로 대입시 대입연산자 함수 호출 -> **2번의 함수 호출로 실행시간 증가**
- 2번의 함수 호출을 복사생성자 1번의 호출로 대체 -> **실행시간 감소로 성능 향상**

예제 5-12 묵시적 복사 생성


```
복사 생성자 실행,1,Kitae
복사 생성자 실행,1,Kitae
복사 생성자 실행,1,dummy
```

```
#include <iostream>
#include <string>
using namespace std;
class Person {
	string name;
	int id;
public:
	Person(int id, string name);
	Person(Person& person);
	void changeName(string name);
	void show() { cout << id << ',' << name << endl; }
};
Person::Person(int id, string name) {
	this->id = id;
	this->name = name;
}
```

예제 5-12 묵시적 복사 생성


```
Person::Person(Person& person) {
	this->id = person.id;
	this->name = person.name;
	cout << "복사 생성자 실행," << this->id <<',' << this->name << endl;
}
void Person::changeName(string name) {
	this->name = name;
}
Person fun(Person person) {
	person.changeName("dummy");
	return person;                    // 복사생성자 호출
}
int main(void) {
	Person father(1, "Kitae");
	Person son = father;              // 복사생성자 호출
	Person result = fun(father);      // 복사생성자 호출
	return 0;
}
```

```
복사 생성자 실행,1,Kitae
복사 생성자 실행,1,Kitae
복사 생성자 실행,1,dummy
```

실습과제1

- 270페이지 문제 중에서 1,2,3,4,5번 풀어서 제출하시오.

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
```