Chap4

객체 포인터

- 객체의 주소를 가르키는 포인터 변수
- 객체를 가르키므로 타입 또한 객체의 클래스 타입

```
Circle donut;
Circle *p;
p = &donut;
```

- 객체 포인터의 경우 객체의 멤버에 접근하려면 ->를 사용

  d = p ->getArea( );

- -> 을 사용하는 대신 (*객체 포인터변수) == 객체로 생각할 수 있기 때문에 이 경우 . 을 써서 객체 멤버에 접근 가능

  d = (*p).getArea( );

객체 배열

- 원소가 객체인 배열

  Circle circleArray[3];

- 배열의 각 원소를 인덱스로 접근 가능하고 . 을 이용해 멤버에 접근
- 객체 배열 선언은 반드시 기본 생성자 호출

  (기본 생성자 외 중복 생성자 존재 시, 기본 생성자 자동 생성 안 되므로 반드시 미리 작성)
- 객체 배열 생성시, 초기화 가능

  Circle circleArray[2] = {Circle(10),Circle( )};

  => 매개변수에 맞게 생성자가 호출되면서 객체가 생성/초기화
- 각 원소 객체마다 소멸자 호출(뒤에서부터 소멸~)

객체 배열의 포인터

- 객체 포인터에 배열의 주소 저장 가능
- 배열은 이름이 곧 주소

  Circle *p = circleArray;
- 포인터변수++ => 다음 배열의 원소를 가르킨다
- 객체 배열의 포인터는 배열의 인덱스를 이용해 멤버에 직접 접근

  p[0].getArea( );

  ((*p).getArea( ); 도 가능)

동적 메모리 할당 및 반환 !

- 필요한 만큼 메모리 할당 /반환
- new 연산자 ->힙(heap)이라는 시스템공간으로부터 메모리 할당

```
데이터타입 *포인터변수 = new 데이터타입;
```

  - 데이터타입의 크기만큼 할당
  - 초깃값 설정 가능 ex) new 데이터타입(10);
  - 포인터변수는 할당받은 메모리의 주소 가르킴
  - 힙메모리 부족하면 NULL 반환(검사필요)

```
int *p = new int; // 힙으로부터 int 타입의 정수 공간 할당
if(!p) { // if(p==NULL)과 동일. p가 NULL이면
    return; // 메모리 할당받기 실패
}
*p = 5; // 할당받은 정수 공간에 5 기록
int n = *p; // 할당받은 정수 공간에서 값 읽기. n = 5
delete p; // 할당받은 정수 공간 반환
```

- delete 연산자 -> 할당 메모리 다시 힙으로 반환
  - 반드시 new로 생성된 동적 메모리 반환

```
delete 포인터변수;
```

배열의 동적 할당/반환

```
데이터타입 *포인터변수 = new 데이터차입[크기]

delete [ ] 포인터변수;
```

- 포인터변수를 통해 보통 배열처럼 접근

  1) 인덱스 이용

  p[i] =1;

  2) 포인터변수가 가르키는 값을 * 통해 접근

  *(p+i) = 2;

- 배열 동적 할당 시, 초기화 가능(원소 하나하나 해줌)

객체 동적 할당/반환

```
클래스이름 *포인터변수 = new 클래스이름; //기본생성자

클래스이름 *포인터변수 = new 클래스이름(매개변수);

delete 포인터변수;
```

- delete 시, 객체 소멸 -> 객체 반환(역순)

객체배열 동적할당

```
클래스이름 *포인터변수 = new 클래스이름[크기];
```

- 배열의 원소의 멤버에 접근하기 위해서는
  1) 인덱스 사용
     p[0].setArea( );
  2) * 이용
     *(p+3).setArea( );
  3)->이용
     (p+3) -> setArea( );

this 포인터

- 객체 자신에 대한 포인터(객체의 주소가 저장)
- 클래스의 멤버 함수 내에서만 사용
- 멤버 변수와 매개변수 이름이 동일 할 때 사용하면 good
- 함수 내에서 return this; 를 통해서 자기 자신을 반환
  리턴타입이 클래스 * (클래스 포인터 타입)
- 정적멤버함수에서는 X
  (정적멤버함수는 객체 생성 전 호출 가능
  but, 객체 자기자신을 반환하는 this 포인터를 쓸 수 X

* 컴파일러는 this를 해당 객체의 주소를 전달하게 만듦
* -> 사용해 멤버에 접근

string 클래스

- c== 표준 라이브러리에서 제공하는 클래스
- 문자열 객체를 다룸
- #include <string>
  using namespace std; 포함 필수!!
- cin, cout 사용 가능

string 객체 생성

1) 빈 문자열
   string str;
2) 문자열 초기화
   string str2("hello");
3) 다른 문자열 객체를 복사해 생성
   string str3(str2);
4) 배열 객체
   char text[] = {'l','o','v','e'};
   string title(text);

string 객체의 동적 생성

- string *p = new string(초기화 문자열);
  delete p;