클래스

김석희

2023.10.28

클래스

클래스(class)는 객체 지향 프로그래밍(OOP)에서 특정 객체를 생성하기 위해 변수와 메서드를 정의하는 일종의 틀이며 내부적으로 객체를 정의하기 위한 상태 값을 의미하는 멤버 변수와 클래스의 동작인 메서드(함수)로 구성됩니다.

객체 지향 프로그래밍에서는 모든 데이터를 객체(object)로 취급하며 이 객체들의 조합으로 프로그래밍을 하는 방식을 의미합니다. C++에서 클래스(class)란 구조체(struct)의 상위 호환으로 이해할 수 있습니다. 구조체와 다른점은 접근 제어 지시자가 추가되었고 함수를 포함할 수 있게 된 점입니다.

클래스

일종의 틀이
메서드(함수)로 구성됩니다.

특정 객체를 생성하기 위해
변수와

객체를 정의하기 위한

클래스(class)란 구조체(struct)의 상위 호환
제어 지시자가 추가되었고 함수를 포함할 수 있게 된 점입니다.

구조체와 다른점은 접근

클래스

특정 객체를 생성하기 위해 일종의 틀, 객체를 정의하기 위한 변수와 메서드(함수)로 구성됩니다.

클래스(class)란 구조체(struct)의 상위 호환
구조체와 다른점은 접근 제어 지시자가 추가되었고 함수를 포함할 수 있게 된 점입니다.

일종의 틀

일종의 틀
구조체

일종의 틀
구조체의 상위호환

클래스

```
struct ME{
    string girl_friend;
    string univercity;
    int age;
    char gender;
    string hobby;
    string secret;
};
```

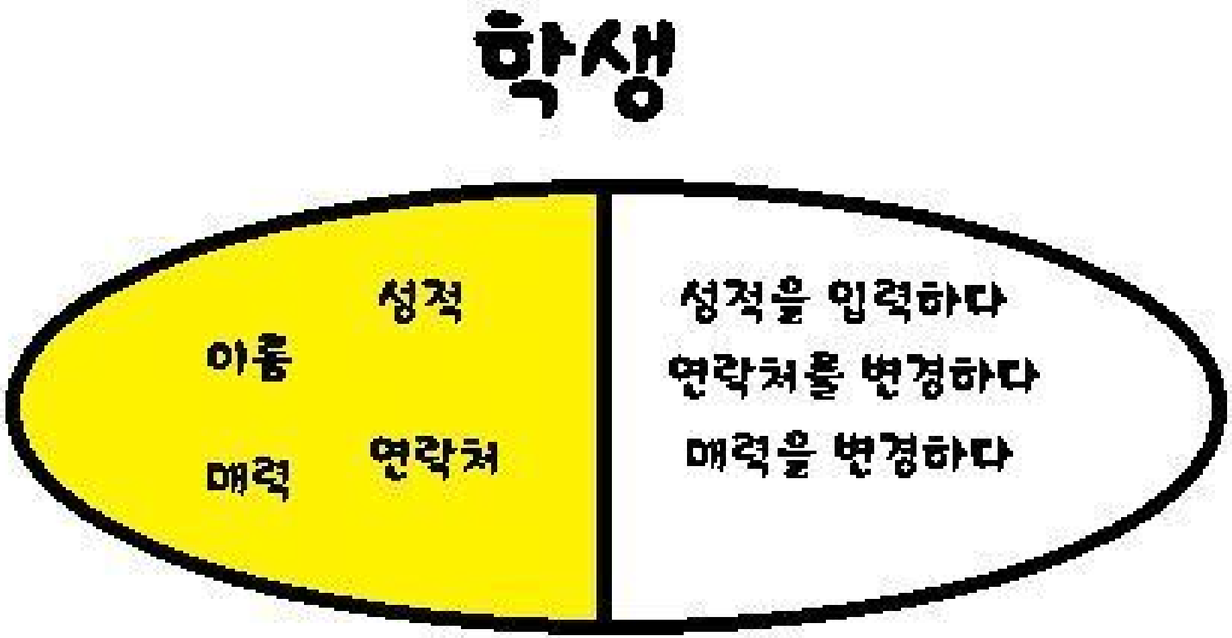
클래스

```
ME seokhui;
seokhui.gender = 'm';
seokhui.age = 22;
seokhui.girl_friend = "없음";
seokhui.univercity = "광운대학교";
seokhui.secret = "길 가면서 노래부름";
```

클래스

```
string 모르는_사람;
cin >> 모르는_사람;
if(모르는_사람 == "seokhui.secret 요청"){
    cout << seokhui.secret;
    seokhui.secret = "내 마음대로 바꾸기";
    seokhui.girl_friend = "코딩";
}
```

은닉화



은닉화

은닉화

* 은닉화 : 멤버 제어를 private으로 외부에서 접근을 막는다. (데이터 보호)
  - 건드려야할 데이터와 건드리지 말아야할 데이터를 구분

```
private:
   string girl_friend;
   string univercity;
   int age;
   char gender;
   string hobby;
   string secret;
```

'나'를 거쳐야만 접근 할 수 있음

캡슐화

캡슐화

* 캡슐화 : 클래스의 데이터와 기능을 하나로 묶어 구성요소를 외부로부터 숨긴다.

```
public:
    void 나랑_친해지기() {
        ask_age();
        ask_univercity();
        ask_hobby();
        ask_secret();
    }
```

취미를 비교하고 싶음

```
ME seokhui, jaesang;

cout << (seokhui == jaesang);
```

?

다형화

* 다형화 : 상속 관계에서 각각 멤버함수 재정의를 통한 다양한 형태를 가지는 것. 코드 유연성
  - 여러 객체가 하나의 메세지를 통해 각기 다른 형태를 취하는 것
  - 프로그래밍 유연성 극대화

구입();			
피자구입();	치킨구입();	음료구입();	커피구입();

```
bool operator==(const ME &another) const {
    if (hobby == another.hobby)
        return true;
    return false;
}
```

클래스

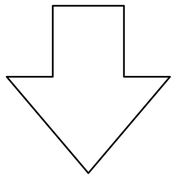
```cpp
#include <iostream>
#include <string>
using namespace std;

class ME {
  private:
    string girl_friend;
    string univercity;
    int age;
    char gender;
    string hobby;
    string secret;

    void ask_age();
    void ask_univercity();
    void ask_hobby();
    void ask_secret();

  public:
    void 나랑_친해지기() {
        ask_age();
        ask_univercity();
        ask_hobby();
        ask_secret();
    }

    bool operator==(const ME &another) const {
        return hobby == another.hobby;
    }
};
```

클래스

```
seokhui.gender = 'm';
seokhui.age = 22;
seokhui.girl_friend = "없음";
seokhui.univercity = "광운대학교";
seokhui.secret = "길 가면서 노래부름";
```

생성자

```
seokhui.gender = 'm';
seokhui.age = 22;
seokhui.girl_friend = "없음";
seokhui.univercity = "광운대학교";
seokhui.secret = "길 가면서 노래부름";
```



```
ME seokhui = ME('m', 22, "없음", "광운대학교", "길 가면서 노래부름");
```

생성자

```
ME(char _gender, int _age, string _girl_friend, string _univercity, string _secret) {
    gender = _gender;
    age = _age;
    girl_friend = _girl_friend;
    univercity = _univercity;
    secret = _secret;
}
```

총합

```
class ME {
  private:
    string girl_friend;
    string univercity;
    int age;
    char gender;
    string hobby;
    string secret;

    void ask_age();
    void ask_univercity();
    void ask_hobby();
    void ask_secret();

  public:
    ME(char _gender, int _age, string _girl_friend, string _univercity, string _secret) {
        gender = _gender;
        age = _age;
        girl_friend = _girl_friend;
        univercity = _univercity;
        secret = _secret;
    }

    void 나랑_친해지기() {
        ask_age();
        ask_univercity();
        ask_hobby();
        ask_secret();
    }

    bool operator==(const ME &another) const { return hobby == another.hobby; }
};
```

클래스 정리

특정 객체를 생성하기 위해 일종의 틀, 객체를 정의하기 위한 변수와 메서드(함수)로 구성됩니다.

클래스(class)란 구조체(struct)의 상위 호환
구조체와 다른점은 접근 제어 지시자가 추가되었고 함수를 포함할 수 있게 된 점입니다.

접근 제어 지시자 덕분에 구조체보다 안전하다.

클래스 정리

접근 제어 지시자

지시자	설명
public	어디서든 접근이 가능 (외부에서도 모두 접근 가능)
private	클래스 내부에 정의된 함수에서만 접근 허용 (중요한 정보를 감출때 사용)
protected	기본적으로는 private이지만 상속관계에 놓여있을 때, 유도 클래스에서는 접근 허용

클래스 정리

* 클래스의 4대 속성

4대 속성	내용
은닉화 (Information hiding)	접근 지정자를 통해 외부로부터 데이터를 보호한다.
캡슐화 (Encapsulation)	클래스의 데이터(멤버 변수)와 기능(멤버 함수)을 하나로 묶어 구성요소(객체의 동작 방식)를 외부로부터 숨긴다.
상속성 (Inheritance)	클래스에서 상속을 통해 같은 기능을 하는 코드를 재사용 할 수 있다. (부모클래스는 자식클래스의 추상적인 존재이다.)
다형성 (Polymorphism)	상속 관계에서 각각 멤버함수 재정의를 통한 다양한 형태를 가지는 것. 코드 유연성 (overidding)

은닉화

* 은닉화 : 멤버 제어를 private으로 외부에서 접근을 막는다. (데이터 보호)
  - 건드려야할 데이터와 건드리지 말아야할 데이터를 구분

```
private:
   string girl_friend;
   string univercity;
   int age;
   char gender;
   string hobby;
   string secret;
```

'나'를 거쳐야만 접근 할 수 있음

캡슐화

* 캡슐화 : 클래스의 데이터와 기능을 하나로 묶어 구성요소를 외부로부터 숨긴다.

```
public:
    void 나랑_친해지기() {
        ask_age();
        ask_univercity();
        ask_hobby();
        ask_secret();
    }
```

다형화

* 다형화 : 상속 관계에서 각각 멤버함수 재정의를 통한 다양한 형태를 가지는 것. 코드 유연성
  - 여러 객체가 하나의 메세지를 통해 각기 다른 형태를 취하는 것
  - 프로그래밍 유연성 극대화

구입();			
피자구입();	치킨구입();	음료구입();	커피구입();

```
bool operator==(const ME &another) const {
    if (hobby == another.hobby)
        return true;
    return false;
}
```

상속

미리 만들어놓은 클래스를 물려받아서

새로운 클래스에서
미리 만들어 놓은 클래스에 있는 함수를 쓸 수 있는 것.

하지만 이 강의에서 제대로 다루지는 않음

클래스 정리

생성자와 소멸자

생성자와 소멸자는 클래스 객체가 생성 및 소멸될 때 자동으로 호출되는 함수입니다. 생성자 소멸자의 경우 따로 기술하지 않아도 컴파일러가 알아서 만들어서 넣습니다. 단 이렇게 생성된 생성자, 소멸자는 아무런 기능이 없습니다. 반대로 아래와 같이 따로 프로그래머가 만들어줄 수도 있습니다.

Copy

```
class MyCar {
private:
    //멤버변수
    int fuel = 0;
    bool power = false;

public:
    //생성자
    MyCar() {
        this->fuel = 100;
        this->power = true;
    }
    //생성자 다중 정의
    MyCar(int n) {
        this->fuel = n;
        this->power = true;
    }
    //소멸자(다중 정의 불가능)
    ~MyCar() {
        std::cout << "소멸되었습니다." << std::endl;
    }
};
```

클래스

원할때만 배열에 값을 넣고 또 빼버릴 수 있는 그런 자료형 -> vector
정렬을 알아서 해주는 그런 함수들을 누가 구현해주면 안되나 -> algorithm
값 두개를 동시에 저장해야 하는데 두개를 동시에 저장해주는 자료형은 없나 -> pair
제곱을 어떻게 구현하지? -> cmath
char 배열은 너무 까다로운 것 같아 -> string
이 문제 입력값이 너무 많은데 메모장에 저장한다음 메모장으로 실행할 수는 없을까? -> fstream
stack, queue 구현 너무 귀찮아 -> stack, queue

외 등등 수많은 헤더파일들

cmath

//서비스함수
ceil(x) : 정수 올림
floor(x) : 정수 내림
abs(x) : 절대값
min(x,y) : x, y중 최소값 반환
max(x,y) : x,y중 최대값 반환

//지수함수
pow(a,b) : a의 b제곱 (a^b)
sqrt(x) : x의 제곱근(루트value)
log(x) : x의 자연로그 값 반환
log10(x) : x의 상용로그 값 반환

//삼각함수
sin(radians) : 라디안 값의 사인값 반환
cos(radians) : 라디안 값의 코사인값 반환
tan(radians) : 라디안 값의 탄젠트값 반환
asin(a) : 입력된 사인값의 라디안값 반환
acos(a) : 입력된 코사인값의 라디안값 반환
atan(a) : 입력된 탄젠트값의 라디안값 반환

+PI

cmath

boj.kr/28701

세제곱의 합 성공 다국어

☆ 한국어

브론즈 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.25 초	1024 MB	2210	1662	1573	76.211%

문제

은하는 수업 때 1부터 N까지 수의 합과 1부터 N까지 수의 세제곱의 합과 관련된 다음 공식을 배웠습니다.

- $(1 + 2 + \cdots + N)^2 = 1^3 + 2^3 + \cdots + N^3$

믿을 수 없었던 은하는 직접 코딩을 해서 검증해 보기로 했습니다. 1부터 N까지 수의 합과 그 수를 제곱한 수, 또 1부터 N까지 수의 세제곱의 합을 차례대로 출력하세요.

입력

첫 줄에 문제의 정수 N이 주어집니다. $(5 \leq N \leq 100)$

출력

세 줄을 출력하세요.

- 첫 줄에는 1부터 N까지 수의 합 $1 + 2 + \cdots + N$을 출력하세요.
- 둘째 줄에는 1부터 N까지 수의 합을 제곱한 수 $(1 + 2 + \cdots + N)^2$을 출력하세요.
- 셋째 줄에는 1부터 N까지 수의 세제곱의 합 $1^3 + 2^3 + \cdots + N^3$을 출력하세요.

예제 입력 1 복사

예제 출력 1 복사

cmath

boj.kr/28701

- 첫 줄에는 1부터 N까지 수의 합 $1 + 2 + \cdots + N$을 출력하세요.
- 둘째 줄에는 1부터 N까지 수의 합을 제곱한 수 $(1 + 2 + \cdots + N)^2$을 출력하세요.
- 셋째 줄에는 1부터 N까지 수의 세제곱의 합 $1^3 + 2^3 + \cdots + N^3$을 출력하세요.

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

int main() {

  ios::sync_with_stdio(false);
  cin.tie(NULL);

  int n;
  cin >> n;
  int sum = n * (n + 1) / 2;
  int sumSqrd = sum * sum;
  int sumOfCubed = 0;

  for (int i = 1; i <= n; i++)
    sumOfCubed += pow(i, 3);

  cout << sum << "\n"
       << sumSqrd << "\n"
       << sumOfCubed << "\n";

  return 0;
}
```

pair

기본함수

선언문

- pair < 자료형, 자료형 > p;

생성

- make_pair(자료형, 자료형) : 두개의 원소를 묶은 pair를 만든다.

조회

- first : 첫번째 인자를 반환
- second : 두번째 인자를 반환

기본 응용

- vector
- v.push_back(pair(자료형, 자료형)) : 보통 좌표값을 배열에 저장할때 사용

pair

기본함수

선언문

· pair < 자료형, 자료형 > p;

생성

· make_pair(자료형, 자료형) : 두개의 원소를 묶은 pair를 만든다.

조회

· first : 첫번째 인자를 반환
· second : 두번째 인자를 반환

기본 응용

· vector
· v.push_back(pair(자료형, 자료형)) : 보통 좌표값을 배열에 저장할때 사용

```
//////////선언/////////
pair<int, int> p;

//////////생성//////
//둘다 차이 없으니 편한걸 사용
p = make_pair(1, 2);
p = { 1,2 };
```

```
cout << p.first << " " << p.second;
```

```
1 2
```

algorithm

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
    int n;
    int arr[100];
    cin >> n;
    for (int i = 0; i < n; i++) cin >> arr[i];

    sort(arr, arr + n);

    for (int i = 0; i < n; i++) cout << arr[i] << " ";

    return 0;
}
```

첫번째 인자로는 정렬 하는 배열의 첫 주소
두번째 인자로는 정렬 하는 배열의 끝 주소

algorithm

boj.kr/2750

수 정렬하기 성공

☆

브론즈 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	187623	106919	73513	57.911%

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000)이 주어진다. 둘째 줄부터 N개의 줄에는 수가 주어진다. 이 수는 절댓값이 1,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

예제 입력 1 복사

```
5
5
2
3
4
1
```

예제 출력 1 복사

```
1
2
3
4
5
```

11021, 9498, 2884, 2739, 2562 - 기초

28431, 11047*, 10808* - 배열 & 반복문
1157*, 11365, 4949*, 28125* - 문자열
2442*, 2523 - 별찍기 시리즈(5,13)
1110, 2869, 1475* - 구현

2292, 5525*,28125*, 18311, 1074*, 11729- 어려움