2. 클래스(class)

C++은 클래스를 이용하여 객체지향 프로그램을 작성할 수 있다.

// C++의 클래스와 객체(instance)

```
class Location          //Location이라는 클래스 정의
{
  private:              //전용 멤버, 비공개부
    int x = 1;
    int y = 2;
  public:               //공용 멤버, 공개부(비공개부의 자료를 취급)
    Location(){ cout<<"기본생성자"; }          //생성자(기본생성자)
    ~Location(){ cout<<"소멸자"; }             //소멸자, 소멸자 앞에는 기호 ~를 붙인다.
    void setx(int sx) { x = sx; }
    void sety(int sy) { y = sy; }
    int getx(void) { return x; }
    int gety(void) { return y; }
};

Location *L = new Location();                  //클래스 Location에 대한 객체 L 생성
delete L;                                      //객체 소멸, 소멸자 호출
```

//접근지정자

public	**공용 멤버** 지정자로 **외부**에서 접근 가능하도록 한다.
protected	**보호 멤버** 지정자로 **하위클래스의 멤버함수와 친구함수**가 접근할 수 있도록 한다.
private	**전용 멤버** 지정자로 **클래스 내부의 멤버함수와 친구함수**가 접근할 수 있도록 한다. 클래스 외부에서는 접근할 수 없다.

- C++은 **struct**과 **class**를 제공한다.
- C++의 **struct**은 멤버함수를 포함할 수 있으므로 C의 구조체보다 좀 더 확장된 것이다.
- C++의 **struct**은 생성자, 멤버함수(메서드), 속성을 포함할 수 있으며, **상속**도 허용한다.
- C++에서 **struct**과 **class**의 차이는 기본 접근제어 차이일 뿐, 나머지는 거의 같다.
- C++의 **class**와 **struct**에서 **접근지정자**를 생략할 경우에 디폴트 값은 **class**에서는 **private**이 되고,
- **struct**에서는 **public**이 된다. → class와 struct 차이점

// 생성자와 소멸자

C++에서 클래스가 상속 관계에 있을 때,
최하위클래스의 객체가 생성/소멸될 때는 부모클래스의 생성자/소멸자가 모두 자동 호출된다.

```
#include <iostream.h>
class A{                                //클래스 A
  private: int x;
  public :
    void set_x(int k){ x = k; }         //x = 3003
    int get_x(){ return x; }            //x = 3003 반환
    A(){ cout<<"A1"; }                  //생성자(기본생성자)
    ~A(){ cout<<"A2"; }                 //소멸자, 소멸자 앞에는 기호 ~를 붙인다.
};
class B : public A{                     //자식클래스 : 부모클래스
  public : B() : A() { cout<<"B1"; } //A()는 생략 가능, B() { cout<<"B1"; }와 같다.
           ~B(){ cout<<"B2"; }
};
class C : public B{
  public : C() : B() { cout<<"C1"; } //B()는 생략 가능, C() { cout<<"C1"; }와 같다.
           ~C(){ cout<<"C2"; }
};
void main(){
  C *p = new C();       //최하위클래스 객체 p 생성, new가 있으면 반드시 포인터로 선언
  p->set_x(3003);       //값 3003 전달
  cout<<p->get_x();     //출력 : 3003 (몸체)
  delete p;             //객체 소멸, 소멸자 호출
}
```

실행결과 : A1B1C13003C2B2A2

- C++에는 **super()**가 없다. 부모생성자 호출은 부모생성자를 **직접 기술**하는 방식이다.
- 실행 순서 : A()→B()→C()→**몸체**→~C()→~B()→~A()
- 생성자 호출 : 부모클래스에서 자식클래스 순(위에서 아래로)
- 소멸자 호출 : 자식클래스에서 부모클래스 순(아래에서 위로)

C *p = new C();가 실행될 때 우선 **기본생성자**가 모두 **자동 호출**되고(부모클래스 생성자 포함),
프로그램이 종료되면서 **소멸자**를 호출하면**(delete),** 모든 소멸자가 **자동 호출**된다.

기출문제 분석

1. 다음 C++ 프로그램의 수행 결과는? [2016년 군무원 9급 유형]

```
#include <iostream>
using namespace std;
class A { public: A(){ cout<<"대한"; } };
class B : public A {
  public: B() { cout<<"민국"; }
};
class C : public B
{
  public: C(): B() { cout<<"만세"; }
};
void main() { C *p = new C(); }
```

① 대한민국만세　② 만세민국대한
③ 민국만세　④ 만세

♤ C++ 프로그램의 기본

```
// 실행순서는 다음과 같다. (상속관계에서 부모클래스의 기본 생성자는 생략 가능)
class A
{
  public: A(){ cout<<"대한"; }          //가장 먼저 출력 : 대한
};
class B : public A                      //자식클래스 : 부모클래스
{
  public: B(): A() { cout<<"민국"; }    //두 번째로 출력 : 민국, A()가 생략되어 있음
};
class C : public B                      //자식클래스 : 부모클래스
{
  public: C(): B() { cout<<"만세"; }    //마지막으로 출력 : 만세, B()는 생략 가능
};

void main() { C *p = new C(); }         //클래스 C의 기본 생성자 C()를 호출
```

• 생성자의 접근지정자는 반드시 public으로 해야 한다.

정답 : ①

2. 다음 C++ 프로그램의 수행 결과는? [2017년 군무원 9급 유형]

```
-----------------------------------------------------
#include <iostream>
using namespace std;
class A{
  int x;
  public: A(int n){x=n; cout<<++x<<",";}
          ~A(){cout<<++x<<",";}
};
class B : public A{
  int y;
  public: B(int m, int n): A(n){y=m; cout<<y++<<",";}
           ~B(){cout<<y++<<",";}
};
void main() {
  B *p = new B(10, 20);
  cout<<"end,";
  delete p;
}
-----------------------------------------------------
```

① 21,10,end,11,22,　　② 10,21,end,22,11,

③ 20,10,11,22,end,　　④ 10,21,22,11,end,

♤ **생성자 / 소멸자**

B *p = new B(10, 20);

↓ B() 생성자 호출

B(int m, int n) : A(n)

↓ 부모 생성자 A(n) 호출

A(int n){x=n; cout<<++x<<",";} → x(20) 값을 1증가시킨 후 출력(21)

↓ 되돌아와서 다음 수행

B(int m, int n){y=m; cout<<y++<<",";} → y 값을 출력(10) 후 1증가시킴(11)

↓ main()으로 돌아가서 다음 수행

cout<<"end,"; → **"end," 출력**

↓ ~B() 소멸자 자동 호출(객체를 회수할 때, 소멸자 자동 호출)

~B(){cout<<y++<<",";} → y 값을 출력(11) 후 1증가시킴(12)

↓ ~A() 소멸자 자동 호출(객체를 회수할 때, 소멸자 자동 호출)

~A(){cout<<++x<<",";} → x(21) 값을 1증가시킨 후 출력(22)

정답 : ①