14. 클래스의 기본문법

- 클래스를 선언하는 문법
- 접근제어 지시자 (구조체를 선언하는 것과은 다른 점)

```
class 클래스 이름
{
접근제어지시자:
    멤버변수선언;
    멤버함수선언 및 정의;
};
```

- 클래스를 이용해 객체지향 프로그램으로 변경

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

// 제작자의 코드
class USERDATA
{
public:
    // 멤버 변수 선언
    int nAge;
    char szName[32];

    // 멤버 함수 선언 및 정의
    void Print()
    {
        // nAge와 szName은 Print() 함수의 지역변수가 아니다!
        cout << nAge << " " << szName << endl;
    }
};

// 사용자의 코드
int main()
{
    USERDATA user = { 20, "철수"};
    user.Print();
    return 0;
}
```

✓ public 은 접근제어 지시자 (public, private, protected)
✓ nAge와 szName은 Print() 함수 내부에 선언된 지역변수가 아니라 Print() 함수가 속한 클래스의 멤버 변수!
✓ USERDATA user = { 20, "철수"} → 기존 구조체 초기화 방법임(C++ 생성자 함수로 초기화 필요)

■ 멤버 선언 및 정의

- 클래스 멤버 변수는 생성자를 이용해 초기화 할 수 있음(구조체와 비교)
- 생성자 함수 : 반환 자료형이 없음.
  호출하는 함수가 아니라 적절한 시기에 내부에서 자동으로 호출되는 함수.

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

// 제작자 코드
class CTest
{
public:
    // CTest 클래스의 '생성자 함수' 선언 및 정의
    CTest()
    {
        // 인스턴스가 생성되면 멤버 데이터를 '자동으로' 초기화 한다.
        m_nData = 10;
    }

    // 멤버 데이터 선언
    int m_nData;

    // 멤버 함수 선언 및 정의
    void PrintData()
    {
        // 멤버 데이터에 접근하고 값을 출력한다.
        cout << m_nData << endl;
    }
};

// 사용자의 코드
int main()
{
    CTest t; //   객체 선언 및 인스턴스 생성되면 생성자 함수 호출!!
    t.PrintData();
    return 0;
}
```

● 생성자 함수의 역할 확인

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

// 제작자 코드
class CTest
{
public:
    // CTest 클래스의 '생성자 함수' 선언 및 정의
    CTest() {
        // 인스턴스가 생성되면 멤버 데이터를 '자동으로' 초기화 한다.
        cout << "CTest() : 생성자 함수" << endl;
        m_nData = 10;
    }

    // 멤버 데이터 선언
    int m_nData;

    // 멤버 함수 선언 및 정의
    void PrintData()
    {
        // 멤버 데이터에 접근하고 값을 출력한다.
        cout << m_nData << endl;
    }
};

// 사용자의 코드
int main()
{
    cout << " main() 함수 시작 " << endl;
    CTest t;
    t.PrintData();
    cout << " main() 함수 끝 " << endl;
    return 0;
}
```

✓ 실행 결과 확인하기!
✓ 생성자 함수는 자동으로 호출!!

● 멤버 함수 선언과 정의를 분리

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

// 제작자 코드
class CTest
{
public:
    // CTest 클래스의 '생성자 함수' 선언 및 정의
    CTest() {
        // 인스턴스가 생성되면 멤버 데이터를 '자동으로' 초기화 한다.
        cout << "CTest() : 생성자 함수" << endl;
        m_nData = 10;
    }

    // 멤버 데이터 선언
    int m_nData;

    // 멤버 함수 선언, 정의는 클래스 외부로 분리됨.
    void PrintData();
};

// 클래스 외부에 분리된 멤버 함수 정의
void CTest::PrintData()
{
    // 멤버 데이터에 접근하고 값을 출력한다.
    cout << m_nData << endl;
}

// 사용자의 코드
int main()
{
    CTest t;
    t.PrintData();
    return 0;
}
```

✓ 멤버 함수 분리해서 정의할 때 함수 이름 앞에 [소속 클래스 이름::] 추가
✓ cout << m_nData << endl에서 m_nData는 CTest 클래스의 멤버 변수인 int m_nData 임.

● 생성자 초기화 목록을 이용한 멤버 변수 초기화

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

// 제작자 코드
class CTest
{
public:
    // 생성자 초기화 목록을 이용한 멤버 초기화
    CTest() : m_nData1(10), m_nData2(20)
    { }

    // 두 개의 멤버 데이터 선언
    int m_nData1;
    int m_nData2;

    // 멤버 함수 선언 및 정의
    void PrintData()
    {
        // 두 개의 멤버 데이터에 접근하고 값을 출력한다.
        cout << m_nData1 << endl;
        cout << m_nData2 << endl;
    }
};

// 사용자의 코드
int main()
{
    CTest t;
    t.PrintData();
    return 0;
}
```

✓ 생성자 초기화 목록은 함수 원형 부분과 몸체를 이루는 블록 범위 사이에 위치
✓ 초기화 목록을 이용하려면 반드시 원형 다음에 콜론(:)을 기술해야 함.
✓ 멤버변수(초기값), 멤버변수(초기값) 형태로 작성
✓ 멤버변수가 참조자 형식이면 반드시 초기화 목록을 이요해 초기화 해야함!

● C++11의 멤버 변수 초기화

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

// 제작자 코드
class CTest
{
public:
    // 생성자 함수
    CTest() { }

    // C++11 부터 선언과 동시에 멤버 변수를 초기화 할 수 있다.
    int m_nData1 = 10;
    int m_nData2 = 20;

    // 멤버 함수 선언 및 정의
    void PrintData()
    {
        cout << m_nData1 << endl;
        cout << m_nData2 << endl;
    }
};

// 사용자의 코드
int main()
{
    CTest t;
    t.PrintData();
    return 0;
}
```