컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 46강 - C++의 클래스

학습 목표

C++의 클래스

1) C++의 클래스(Class) 문법에 대해서 이해하고 이를 활용하는 방법에 대해서 소개합니다.
2) 객체 지향 프로그래밍의 이론적인 배경을 이해할 수 있습니다.

C++의 클래스

구조체와 클래스의 차이점

일반적으로 C++의 클래스(Class)는 구조체보다 더 효과적인 문법입니다. 구조체와 클래스는 거의 흡사하게 동작하지만, 클래스에서는 내부적으로 '함수' 등을 포함할 수 있습니다.

또한 클래스는 상속(Inheritance) 등의 개념을 프로그래밍에서 그대로 이용할 수 있다는 점에서 객체 지향 프로그래밍(Object-Oriented Programming)을 가능하도록 해주는 기본적인 단위입니다.

C++의 클래스

구조체와 클래스의 차이점: 구조체

```
#include <iostream>
#include <string>

using namespace std;

struct student {
  string name;
  int score;
};

int main(void) {
  struct student a;
  a.name = "나동빈";
  a.score = 100;
  cout << a.name << " : " << a.score << "점\n";
  system("pause");
}
```

C++의 클래스

구조체와 클래스의 차이점: 클래스

```
#include <iostream>
#include <string>

using namespace std;

class Student {
private:
  string name;
  int score;
public:
  Student(string n, int s) { name = n; score = s; }
  void show() { cout << name << " : " << score << "점\n"; }
};

int main(void) {
  Student a = Student("나동빈", 100);
  a.show();
  system("pause");
}
```

C++의 클래스

객체 지향 프로그래밍의 특징

C++은 클래스(Class)를 이용해 '현실 세계의 사물'인 객체(Object)를 프로그램 내에서 구현할 수 있도록 해주며, 객체 지향 프로그래밍은 다음과 같은 특징 때문에 소스코드를 보다 간결하고 생산성 높게 만들어 줍니다.

1) 추상화(Abstract)
2) 캡슐화(Encapsulation)
3) 상속성(Inheritance)
4) 정보 은닉(Data Hiding)
5) 다형성(Polymorphism)

C++의 클래스

C++의 클래스: 멤버(Member)

- 멤버 변수를 속성, 혹은 프로퍼티(Property)라고도 부릅니다.
- 멤버 함수를 메소드(Method)라고도 부릅니다.

```
class Student {
private:
  string name;
  int score;
public:
  Student(string n, int s) { name = n; score = s; }
  void show() { cout << name << " : " << score << "점\n"; }
};
```

C++의 클래스

C++의 클래스: 인스턴스(Instance)

- C++에서는 클래스를 활용해 만든 변수를 인스턴스(Instance)라고 합니다. 실제로 프로그램 상에서 객체가 살아서 동작하도록 해줍니다. 하나의 클래스에서 여러 개의 서로 다른 인스턴스를 만들 수 있습니다.

```
int main(void) {
  Student a = Student("나동빈", 100);
  a.show();
  system("pause");
}
```

C++의 클래스

C++의 클래스: 접근 한정자

- public: 클래스, 멤버 등을 외부로 공개합니다. 해당 객체를 사용하는 어떤 곳에서도 접근할 수 있습니다.
- private: 클래스, 멤버 등을 내부에서만 활용합니다. 외부에서 해당 객체에 접근할 수 없습니다.

클래스는 기본적으로 멤버를 private 형태로 간주합니다. 반대로 구조체는 기본적으로 멤버를 public으로 간주합니다. 따라서 클래스에서 'private:' 부분을 제외하면 멤버는 자동으로 private 문법을 따릅니다.

C++의 클래스

C++의 클래스: 접근 한정자

```
class Student {
private:
  string name;
  int englishScore;
  int mathScore;
  int getSum() { return englishScore + mathScore; } // 정보 은닉
public:
  Student(string n, int e, int m) {
    name = n;
    englishScore = e;
    mathScore = m;
  }
  void show() { cout << name << " : [합계 " << getSum() << "점]\n"; }
};
```

C++의 클래스

C++의 클래스: 접근 한정자

클래스 내부에서 정의된 멤버 함수를 불러올 때는 멤버 참조 연산자(.)를 사용하여 불러오게 됩니다.

```
int main(void) {
  Student a = Student("나동빈", 100, 98);
  a.show();
  cout << a.getSum(); // private 접근 한정자는 외부에서 접근할 수 없음. (오류 발생)
  system("pause");
}
```

C++의 클래스

C++의 클래스: this 포인터

기본적으로 하나의 클래스에서 생성된 인스턴스(Instance)는 서로 독립된 메모리 영역에 멤버 변수를 저장하고, 관리합니다. 다만 멤버 함수는 모든 인스턴스가 공유한다는 점에서, 함수 내에서 인스턴스를 구분할 필요가 있습니다.

C++의 this 포인터는 포인터 자료형으로, '상수'라는 점에서 값을 변경할 수 없습니다.

C++의 클래스

C++의 클래스: this 포인터

```
class Student {
private:
  string name;
  int englishScore;
  int mathScore;
  int getSum() { return englishScore + mathScore; }
public:
  Student(string name, int englishScore, int mathScore) {
    this->name = name; // 자기 자신의 멤버 변수에 접근
    this->englishScore = englishScore;
    this->mathScore = mathScore;
  }
  void show() { cout << name << " : [합계 " << getSum() << "점]\n"; }
};
```

배운 내용 정리하기

C++의 클래스

1) C++의 클래스(Class)는 객체 지향 프로그래밍을 위한 기본적인 단위입니다.