컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 49강 - C++의 오버로딩

학습 목표

C++의 오버로딩

1) C++의 오버로딩의 필요성과 활용 방법에 대해서 이해할 수 있습니다.

C++의 오버로딩

C++의 함수 오버로딩

C++은 동일한 이름의 멤버 함수를 다양한 방식으로 활용하기 위해서 오버로딩을 사용할 수 있습니다.

C++의 오버로딩

C++의 함수 오버로딩

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

using namespace std;

class Person {
private:
  string name;
public:
  Person() { name = "임꺽정";  }
  Person(string name): name(name) { }
  void showName() { cout << "이름: " << name << '\n'; }
};

int main(void) {
  Person person1;
  person1.showName();
  Person person2("나동빈");
  person2.showName();
  system("pause");
}
```

C++의 오버로딩

C++의 연산자 오버로딩

C++은 연산자 오버로딩 문법을 활용해 연산자 또한 원하는 방식으로 수정하여 사용할 수 있다는 특징이 있습니다.

1) 기존에 존재하는 연산자만 정의할 수 있습니다.
2) 멤버 연산자(.), 범위 지정 연산자(::) 등의 몇몇 연산자는 오버로딩 처리할 수 없습니다.
3) 기본적인 연산자의 규칙을 따라야합니다.
4) 오버로딩이 된 연산자의 피연산자 중 하나는 사용자 정의 자료형 이어야만 합니다.

C++의 오버로딩

C++의 연산자 오버로딩

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

using namespace std;

class Person {
private:
  string name;
public:
  Person() { name = "임꺽정"; }
  Person(string name): name(name) { }
  Person operator +(const Person& other) { return Person(name + " & " + other.name); }
  void showName() { cout << "이름: " << name << '\n'; }
};

int main(void) {
  Person person1;
  Person person2("나동빈");
  Person result = person1 + person2;
  result.showName();
  system("pause");
}
```

배운 내용 정리하기

C++의 오버로딩

1) C++에서는 함수 오버로딩을 통해서 동일한 이름의 함수를 약간씩 변형하여 사용할 수 있습니다.
2) C++에서는 자주 이루어지는 특정한 계산을 연산자 오버로딩을 통해서 정리할 수 있습니다.