프로그래밍입문(2)
실습

13주 차

객체 연산자 오버로딩

- **1)**계좌 정보를 저장하는 **Account** 클래스에 연산자 오버로딩을 생성하여 서로 다른 두 계좌의 금액을 더하는 프로그램을 완성하시오

- **main** 함수

```
int main(){

        Account ac1(“110”,1000);
        Account ac2(“112”,200);

        Account ac3=ac1+ac2;
        ac3.show_account();

        return 0;
}
```

- 출력결과

출력결과

계좌번호:110 계좌잔액 : 1200

객체 연산자 오버로딩

- class account

```
class Account{
        private:
                string acc_num;       //계좌번호
                int balance;          //계좌잔액
        public:
                Account(string num, int bal);
                const string getAcc_num(void) const;
                int getBalance(void) const;
                Account operator+(const Account& ac1);
                void show_account();
};
```

객체 연산자 오버로딩

- **1) Solution - class**

```
class Account{
	private:
		string acc_num;  //계좌번호
		int balance;     //계좌잔액
	public:
		Account(string num, int bal){
			acc_num = num;
			balance = bal;
		 }
		const string getAcc_num(void) const;
		int getBalance(void) const;
		Account operator+(const Account& ac1){
				if(acc_num != ac1.acc_num){
					balance+=ac1.balance;
				}
				return *this;
		}
		void show_account(){
				cout << "계좌번호 :"<<acc_num <<" "<< "계좌잔액 :"<< balance << endl;
		}
};
```

객체 연산자 오버로딩

- **1) Solution - main**

```
int main(){

        Account ac1(“110”,1000);
        Account ac2(“112”,200);

        Account ac3=ac1+ac2;
        ac3.show_account();

}
```

연산자 오버로딩

- **2) 3**차원의 좌표 **x, y, z** 값을 정수형태로 저장하는 **class Point**를 이용한 프로그램이다**.** 다음 조건을 만족시키는 프로그램을 완성하시오**.**

- **Point** 클래스

```
class Point{
        private:
        int x, y, z;

         public:
             //필요한 멤버함수 선언
}; };
```

- **main** 함수

```
int main(){
        Point p1(1, 3, 5);

        ++p1;
        p1.show_point();

        --p1;
        p1.show_point();
}
```

연산자 오버로딩

- **2) Solution**

```
#include <iostream>
using namespace std;
class Point{
        private:
        int x, y, z;
        public:
        Point(){}
        Point(int x1, int y1, int z1): x(x1), y(y1), z(z1){};

        Point operator++(){
            x++;y++;z++;
            return *this;
        }
        Point operator--(){
            x--;y--;z--;
            return *this;
        }
        void show_point(){
            cout << x << y << z << endl;
        }
};
```

```
int main(){
        Point p1(1, 3, 5);

        ++p1;
        p1.show_point();}

        --p1;
        p1.show_point();
}
```

연산자 오버로딩 응용

- **3) 2**번 문제 프로그램에서 다음과 같이 **main**함수가 추가 되었을 때, 의도한 바와 출력결과가 나올 수 있도록 프로그램을 수정하시오.

- **main** 함수

```
int main(){
	Point p1(1, 3, 5);

	++p1;
	p1.show_point();

	--p1;
	p1.show_point();

	p1++;
	p1.show_point();

	++(++p1);
	p1.show_point();
}
```

연산자 오버로딩 응용

- **3) Solution**

```
#include <iostream>
using namespace std;
class Point{
        private:
        int x, y, z;
        public:
        Point(){}
        Point(int x1, int y1, int z1): x(x1), y(y1), z(z1){};
        Point operator++(Point &p){
            x+=p.x;
            y+=p.y;
            z+=p.z;
            return *this;
        }
        Point operator--(Point &p){
            x-=p.x;
            y-=p.y;
            z-=p.z;
            return *this;
        }
        Point& operator++(){
             x++;y++;z++;
             return *this;
        }
        Point& operator++(int){
            x++;y++;z++;
            return *this;
        }
        Point& operator--(){
            x--;y--;z--;
            return *this;
        }
        void show_point(){
            cout << x << y << z << endl;
        }
};

int main(){
        Point p1(1, 3, 5);

        ++p1;
        p1.show_point();

        --p1;
        p1.show_point();

        p1++;
        p1.show_point();

        ++(++p1);
        p1.show_point();
}
```

대입 연산자

- **4)** 다음은 책의 이름과 가격을 정하는 **Book** 클래스를 이용한 프로그램이다**. main** 함수를 참고하여 출력 결과가 올바르게 나올 수 있도록 대입 연산자를 이용하여 프로그램을 작성하시오

- **Point** 클래스

```
class Book{
        string title;
        int price;

        public:
                Book(string title, int price);
                 ~Book(){}
                 void set(string title, int price);
                 void show(){ cout << title << ' ' << price << "원"<<endl;}
                 Book& operator=(const Book& b);
};
```

대입 연산자

- **4)**

- **main 함수**

```
int main(){

        Book cpp(“C++”, 10000);
        Book c=cpp;
        c.set(“CPP”, 12000);
        cpp.show();
        c.show();

        return 0;
}
```

- 출력결과

출력결과

```
C++ 10000원
CPP 12000원
```

대입 연산자

- **4) Solution**

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

class Book{
        string title;
        int price;

        public:
            Book(string title, int price);
            ~Book(){}
            void set(string title, int price);
             void show(){ cout << title << ' ' << price <<
"원"<<endl;}
            Book& operator=(const Book& b);
};

Book :: Book(string title, int price) {
        this->title=title; this->price=price;
}
```

```
void Book::set(string title, int price) {
        this->title=title; this->price=price;
}

Book& Book::operator=(const Book& b){
        title=b.title;
        price=b.price;
        return *this;
}

int main(){

        Book cpp("C++", 10000);
        Book c=cpp;
        c.set("CPP", 12000);
        cpp.show();
        c.show();

        return 0;
}
```

삽입 연산자

- **5) x, y좌표로 한 점을 표현하는 Point 클래스의 객체를 화면에 출력하는 << 연산자를 작성하라.**

- **main** 함수

```
int main(){

        Point p(3,4);
        cout << p << endl;

        Point q(1, 100), r(2, 200);
        cout << q<< r<< endl;

        return 0;
}
```

- 출력화면

출력결과

```
(3,4)
(1,100) (2,200)
```

삽입 연산자

- **5) Solution**

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

class Point{

    int x, y;

    public:

        Point(int x=0, int y=0){

            this->x=x;
            this->y=y;

        }

    friend ostream& operator<<(ostream& stream, Point a);

};
```

```
ostream& operator << (ostream& stream, Point a){

    stream << "(" << a.x << "," << a.y << ")";

    return stream;

}

int main(){

                Point p(3,4);
                cout << p << endl;

                Point q(1, 100), r(2, 200);
                cout << q<< r<< endl;

                return 0;
}
```

다중 상속

- **6)** 다음 출력결과를 보고 **Person** 클래스와 **Clothes** 클래스를 다중 상속받는 **Student** 클래스를 작성하시오**.**

- 출력 결과

출력결과

```
철수 20 롱패딩
1910 80
```

- 클래스 **Person**

```
class Person{
    string name;
    int age;

    public:
        Person(string name, int age){
            this->name=name;
            this->age=age;
        }
        string getName(){ return name; }
        void setAge();
        int getAge(){ return age; }
};
```

다중 상속

- **6)**

- 클래스 **Clothes**

```
class Clothes{
    string c_name;

    public:
        Clothes(string name):c_name(name){}
        void setName(string name);
        string getName(){ return c_name; }
};
```

- **main** 함수

```
int main(){

    Student st1("철수",20,"롱패딩",1910,80);

    st1.show();
}
```

다중 상속

- **6) Solution**

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

class Person{
    string name;
    int age;

    public:
        Person(string name, int age){
            this->name=name;
            this->age=age;
        }
        string getName(){ return name; }
        void setAge();
        int getAge(){ return age; }
};
```

```
class Clothes{
    string c_name;

    public:
        Clothes(string
name):c_name(name){}
        void setName(string name);
        string getName(){ return c_name;
}
};
```

다중 상속

- **6) Solution**

```
class Student: public Person, public Clothes{
    int stu_ID;
    int grade;

    public:
    Student(string name1, int age1, string c_name1, int stu_ID1, int grade1):Person(name1,age1),
Clothes(c_name1){
        this->stu_ID=stu_ID1;
        this->grade=grade1;
    }
    void show(){
        cout << Person::getName() << ' ' << getAge()<< ' ' << Clothes::getName()<< endl;
        cout << stu_ID << ' '<< grade << endl;
    }

};

int main(){

    Student st1("철수",20,"롱패딩",1910,80);

    st1.show();
}
```

참고문헌

- 명품 C++ Programming, 황기태 , 생능출판사, 2018

Q & A