

## 객체지향프로그래밍 LAB #11

### <기초문제>

---

1. 아래의 프로그램을 작성하시오. (\*구현\*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

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

class Point {
private:
    int x;
    int y;
    static int numCreatedObjects;
    // ifstream fin;
    // int *ary;
public:
    Point() : x(0), y(0) {
        numCreatedObjects++;
        // fin.open("test.txt");
        // ary = new int[100];
    }
    Point(int _x, int _y);

    ~Point() {
        // fin.close();
        // delete[] ary;
        cout << "Destructed..." << endl;
    }

    static int getNumCreatedObject() { return numCreatedObjects; }

    void setXY(int _x, int _y) {
        //this-> 사용한 초기화
        /*구현*/
    }
    int getX() const;
    int getY() const;

    //operator overloading(연산자 오버로딩)
    Point& operator=(Point& pt) {
        /*구현*/
    }
    // *this + pt2 ->
    Point operator+(Point& pt2) {
        /*구현*/
    }
    friend ostream& operator<<(ostream& cout, Point& pt);
    friend void print(const Point& pt);
    friend class SpyPoint;
};

//static 멤버 변수 초기화 (numCreatedObjects)
/*구현*/
```

```

Point::Point(int _x, int _y) : x(_x), y(_y) {
    numCreatedObjects++;
}

int Point::getX() const { /*구현*/ }
int Point::getY() const { /*구현*/ }

//Point operator+(Point& pt1, Point& pt2){
//    Point result(pt1.getX() + pt2.get(X), pt1.getY() + pt2.getY());
//    return result;
//}

//객체를 const로 입력받으면, 반드시 const로 선언된 멤버 함수만 사용 가능
void print(/*구현*/) {
    cout << pt.x << ", " << pt.y << endl << endl;
}

ostream& operator<<(ostream& cout, Point& pt) {
    /*구현*/
}

class SpyPoint {
public:
    //다음과 같이 출력 되도록 hack_all_info함수 구현

    //Hacked by SpyPoint
    //x: 40
    //y: 60
    //numCreatedObj.: 10

    /*구현*/

    // friend class Point;
};

int main() {
    Point pt1(1, 2);
    cout << "pt1 : ";
    print(pt1);

    //pt1을 출력하는 2가지 방법 구현
    Point* pPt1 = &pt1;
    cout << "pt1 : ";
    /*구현*/
    cout << "pt1 : ";
    /*구현*/

    //동적으로 Point* pPt2할당하여 10,20넣은 뒤 ->사용하여 출력(pt1 출력 참고)
    /*구현*/
}

```

```

// Point* pPt3;   pPt3->setXY(5, 6); //말이 안됨
int a = 3 + 5;
Point pt2(10, 20), pt3(30, 40);

//pt4 = pt2, pt3값 더하기
/*구현*/

cout << "pt2 : " << pt2 << endl; // operator<<(cout, pt2)
cout << "pt3 : " << pt3 << endl;
cout << "pt4 : " << pt4 << endl;

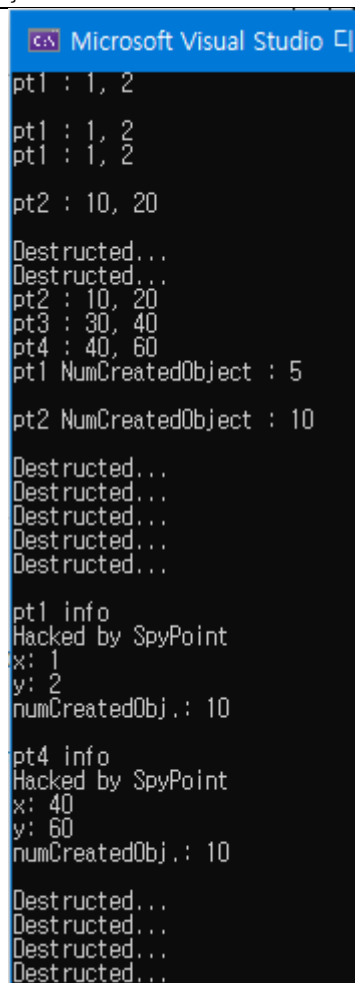
cout << "pt1 NumCreatedObject : " << /*구현*/ << endl << endl;

Point* ptAry = new Point[5];
cout << "pt2 NumCreatedObject : " << /*구현*/ << endl<<endl;
delete[] ptAry;

SpyPoint spy;
cout << endl<<"pt1 info" << endl;
spy.hack_all_info(pt1);
cout << "pt4 info" << endl;
spy.hack_all_info(pt4);

return 0;
}

```



```

Microsoft Visual Studio
pt1 : 1, 2
pt1 : 1, 2
pt1 : 1, 2
pt2 : 10, 20
Destructed...
Destructed...
pt2 : 10, 20
pt3 : 30, 40
pt4 : 40, 60
pt1 NumCreatedObject : 5
pt2 NumCreatedObject : 10
Destructed...
Destructed...
Destructed...
Destructed...
Destructed...
pt1 info
Hacked by SpyPoint
x: 1
y: 2
numCreatedObj.: 10
pt4 info
Hacked by SpyPoint
x: 40
y: 60
numCreatedObj.: 10
Destructed...
Destructed...
Destructed...
Destructed...

```

## <응용문제>

1. 세명의 User 정보(ID, Password)를 입력받은 후에 반복하여 로그인하는 프로그램을 구현하시오.

- 입력받은 User는 세명임.
- 로그인 시에 ID에 "종료"를 입력하면 프로그램이 종료함.
- ID는 중복될 수 없음. 중복된 ID를 입력할 경우에 종료함.

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

int main() {
    User user[3];
    string id, password, searchId, searchPassword;
    for (int i = 0; i < 3; i++) { /* User 정보를 입력받음 */ }
    while (1) { /* Login 기능구현 */ }
}
```

1-출력화면:

```
Microsoft Visual Studio 디버그 콘솔
===== 1 =====
ID : hideonbush
password : 12345
=====

===== 2 =====
ID : gookbobchoong
password : 999999
=====

===== 3 =====
ID : 가나다
password : 1q2w3e4r!
=====

===== Login =====
ID : hideonbush
Password : 12345
로그인 되었습니다.
=====

===== Login =====
ID : 가나다
Password : 1q2w
잘못된 ID거나 PASSWORD 입니다.
=====

===== Login =====
ID : gook
Password : 999999
잘못된 ID거나 PASSWORD 입니다.
=====

===== Login =====
ID : 종료
종료하겠습니다.
```

```
Microsoft Visual Studio 디버그 콘솔
===== 1 =====
ID : abcd
password : 1234
=====

===== 2 =====
ID : abcd
이미 존재하는 ID입니다.
종료합니다.
```

2. 좌표값 두 개를 입력받고 두 좌표 사이의 거리를 출력하시오.

- 좌표값은 초기화 시, 값을 따로 주지 않으면  $x = 0, y = 0$ 으로 초기화함.
- "좌표 - 좌표" 연산자를 Operator overloading을 활용하여 선언함.
  - $(x_1, y_1) - (x_2, y_2) = (x_1 - x_2, y_1 - y_2)$ 를 클래스 내에서 구현.
- "좌표 \* 좌표" 연산자를 Operator overloading을 활용하여 선언함.
  - $(x_1, y_1) * (x_2, y_2) = (x_1 \times x_2, y_1 \times y_2)$ 를 클래스 내에서 구현.
- 위 두 연산자를 활용하여 두 좌표 사이의 거리를 구함.
- 제곱근 사용을 위해 `#include<cmath>` 선언.

```
int main() {
    int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
    Point* pP1, * pP2, * pP3;

    cout << "첫번째 좌표(x1, y1)를 입력하세요 : ";
    cin >> x1 >> y1;

    cout << "두번째 좌표(x2, y2)를 입력하세요 : ";
    cin >> x2 >> y2;

    pP1 = new Point(x1, y1);
    pP2 = new Point(x2, y2);
    pP3 = new Point(); //x,y가 0으로 초기화

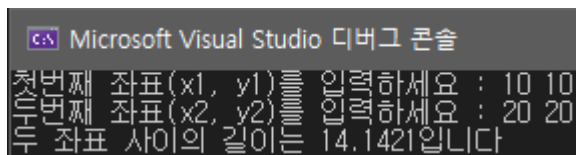
    /* 아래의 방식으로도 x, y값 설정이 가능해야 함 */
    //pP1->setPoint(x1, y1);
    //pP1->setPoint(x2, y2);
    /*******/

    *pP3 = (*pP1 - *pP2) * (*pP1 - *pP2);

    /* pP3을 활용하여 거리값을 구함 */
    cout << "두 좌표 사이의 길이는 " << /* 결과 값 */ << "입니다." << endl;

    return 0;
}
```

2-출력화면:



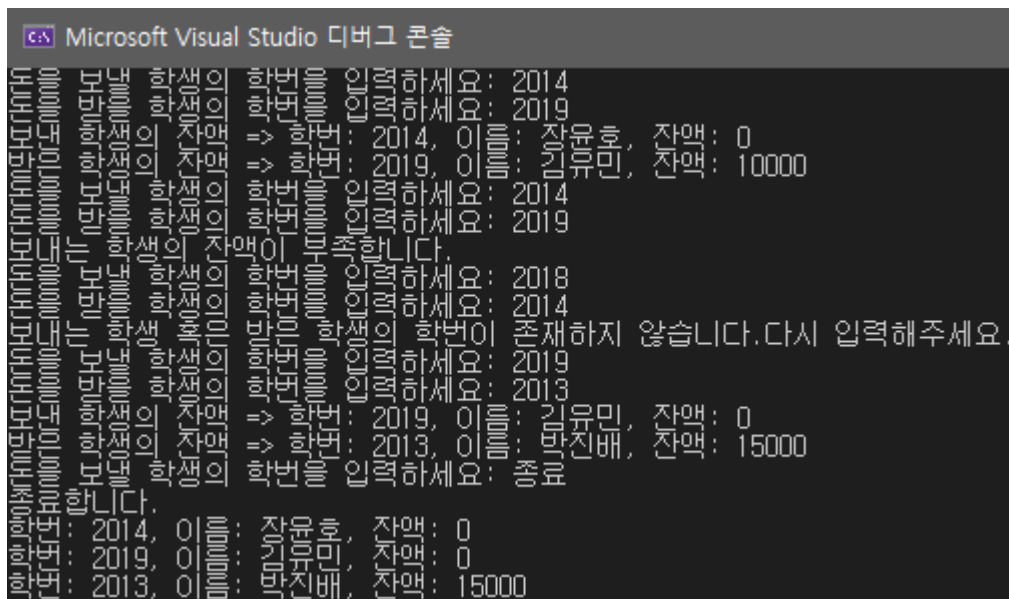
```
Microsoft Visual Studio 디버그 콘솔
첫번째 좌표(x1, y1)를 입력하세요 : 10 10
두번째 좌표(x2, y2)를 입력하세요 : 20 20
두 좌표 사이의 길이는 14.1421입니다
```

### 3. 학생들끼리 송금하는 프로그램을 작성하시오.

- 한번 송금할 때, 돈을 보내는 학생의 전 재산(balance)를 송금함.
- Account(계좌) class는 멤버변수로 string name(이름), string id(학번), int balance를 갖고 있도록 함.
- 사용자로부터 돈을 보낼 학생의 학번과 돈을 받을 학생의 학번을 입력 받음. 이때, 다음의 경우에 대해서는 다시 입력 받도록 함.
  - 돈을 보낼 학생과 돈을 받을 학생의 학번이 동일한 경우
  - 보낼 학생 혹은 받을 학생의 학번이 존재하지 않는 경우
  - 보낼 학생의 잔액이 0인 경우
- setBalance함수를 사용하지 않고, Operator overloading(+, -)를 이용하여 송금 후 보낸 학생의 잔액과 받은 학생의 잔액을 계산함.
- 송금이 완료된 후, Operator overloading(<<)을 이용하여 돈을 보낸 학생과 받은 학생의 계좌를 출력함.
- 사용자로부터 돈을 보낼 학생의 학번을 "종료"라고 입력받았을 경우 Operator overloading(<<)을 이용하여 모든 학생의 계좌를 출력하고, 프로그램을 종료함.

```
int main() {  
    Account acnt[3] = {  
        Account("장윤희", "2014", 10000),  
        Account("김유민", "2019", 0),  
        Account("박진배", "2013", 5000),  
    };  
  
    /* 구현 */  
}
```

### 3-출력화면:



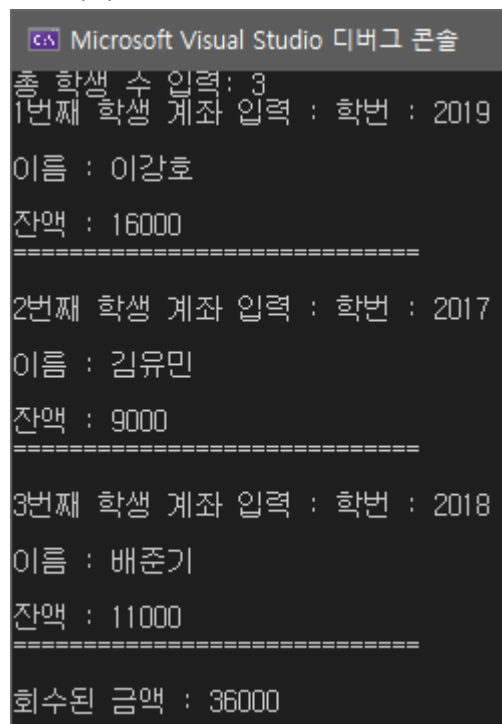
```
Microsoft Visual Studio 디버그 콘솔  
학생의 학번을 입력하세요: 2014  
학생의 이름을 입력하세요: 장윤희  
학생의 잔액 => 학번: 2014, 이름: 장윤희, 잔액: 0  
학생의 학번을 입력하세요: 2019  
학생의 이름을 입력하세요: 김유민  
학생의 잔액 => 학번: 2019, 이름: 김유민, 잔액: 10000  
학생의 학번을 입력하세요: 2014  
학생의 이름을 입력하세요: 장윤희  
학생의 잔액이 부족합니다.  
학생의 학번을 입력하세요: 2018  
학생의 이름을 입력하세요: 장윤희  
학생의 잔액이 부족합니다.  
학생의 학번을 입력하세요: 2014  
학생의 이름을 입력하세요: 장윤희  
학생의 잔액이 부족합니다.  
학생의 학번을 입력하세요: 2019  
학생의 이름을 입력하세요: 김유민  
학생의 잔액 => 학번: 2019, 이름: 김유민, 잔액: 0  
학생의 학번을 입력하세요: 2013  
학생의 이름을 입력하세요: 박진배  
학생의 잔액 => 학번: 2013, 이름: 박진배, 잔액: 15000  
학생의 학번을 입력하세요: 종료  
합계:  
2014, 이름: 장윤희, 잔액: 0  
2019, 이름: 김유민, 잔액: 0  
2013, 이름: 박진배, 잔액: 15000
```

4. 학생 계좌 정보를 입력받고 학생 계좌 정보들을 모두 삭제하여 삭제된 계좌들의 잔액 총합을 출력하는 프로그램을 작성하시오.

- 총 학생 수를 사용자로부터 입력받음.
- 학생 계좌 정보 Account class는 학번, 이름, 잔액을 멤버 변수로 가지고 있음.
- Account 클래스 배열을 동적으로 생성함.
- 각 학생의 학번, 이름, 잔액을 입력받음. 이 때, 중복된 학번을 입력받으면 프로그램을 종료함.
- 모두 입력 받은 후, Account 클래스 배열을 delete하여 모든 학생 계좌의 잔액 총합을 출력함.

Hint) Account class 내 멤버 변수에서 static 변수를 사용, Account 클래스의 소멸자를 적절히 이용.

4-출력화면:



```
Microsoft Visual Studio 디버그 콘솔
총 학생 수 입력: 3
1번째 학생 계좌 입력 : 학번 : 2019
이름 : 이강호
잔액 : 16000
=====
2번째 학생 계좌 입력 : 학번 : 2017
이름 : 김유민
잔액 : 9000
=====
3번째 학생 계좌 입력 : 학번 : 2018
이름 : 배준기
잔액 : 11000
=====
회수된 금액 : 36000
```