

객체지향프로그래밍 LAB #11

<기초문제>

1.

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

class Point {
private:
    int x;
    int y;
    static int numCreatedObjects;

public:
    Point() : x(0), y(0) {
        numCreatedObjects++;
    }

    // int _x 와 int _y를 입력으로 받는 생성자
    Point(int _x, int _y) {
        x = _x;
        y = _y;
        numCreatedObjects++;
    }

    ~Point() {
        cout << "Destructed..." << endl;
    }

    void setXY(int _x, int _y) {
        // this-> 사용한 초기화
        this->x = _x;
        this->y = _y;
    }

    int getX() const { return x; }
    int getY() const { return y; }

    // *this + pt2
    Point operator+(Point& p) {
        return Point(this->x + p.x, this->y + p.y);
    }
}
```

```

// operator overloading(연산자 오버로딩)
Point& operator=(Point& t) {
    if (this != &t) {
        this->x = t.x;
        this->y = t.y;
    }
    return *(this);
}

static int getNumCreatedObject() { return numCreatedObjects; }
friend void print(const Point& pt);
friend ostream& operator<<(ostream& cout, Point& pt);
friend class SpyPoint;
};

// static 멤버 변수 초기화 (numCreatedObjects)
int Point::numCreatedObjects = 0;

// 객체 call by reference 시: const로 함수 입력 시 const method만 함수에서 사용 가능
// const: 객체 내부의 member data가 상수(변하지 않는다)
void print(const Point& p) {
    cout << p.x << ", " << p.y << endl;
}

ostream& operator<<(ostream& cout, Point& p) {
    cout << p.x << ", " << p.y;
    return cout;
}

class SpyPoint {
public:
    void hack_all_info(const Point& p) {
        cout << "Hacked by SpyPoint" << endl;
        cout << "x: " << p.x << endl;
        cout << "y: " << p.y << endl;
        cout << "numCreatedObj.: " << Point::numCreatedObjects << endl << endl;
    }
};

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

```

```

// 포인터
Point* pPt1 = &pt1;
// pPt1의 값을 통해 getX, getY를 호출하여 출력
cout << "pt1 : ";
cout << pPt1->getX() << ", " << pPt1->getY() << endl;
// pPt1를 통해 호출 getX, getY를 호출하여 출력
cout << "pt1 : ";
cout << (*pPt1).getX() << ", " << (*pPt1).getY() << endl;

cout << endl;

// 동적으로 Point* pPt2 할당하여 10, 20 넣은 뒤 -> 사용하여 출력(pt1 출력 참고)
Point* pPt2 = new Point(10, 20);
cout << "pt2 : ";
cout << pPt2->getX() << ", " << pPt2->getY() << endl;
cout << endl;

// pPt1, pPt2의 메모리 해제
delete pPt2;

cout << "pt1 NumCreatedObject : ";
cout << Point::getNumCreatedObject() << endl;

// 연산자 오버로딩
//pt4 = pt2, pt3값 더하기
Point pt2(10, 20);
Point pt3(30, 40);
Point pt4 = pt2 + pt3;
cout << "pt2 : " << pt2 << endl;
cout << "pt3 : " << pt3 << endl;
cout << "pt4 : " << pt4 << endl;

cout << "pt1 NumCreatedObject : ";
cout << Point::getNumCreatedObject() << endl << endl;

// object array
Point* ptAry = new Point[5];
cout << "pt2 NumCreatedObject : ";
cout << Point::getNumCreatedObject() << endl;
cout << endl;

// ptAry 메모리 해제
delete[] ptAry;

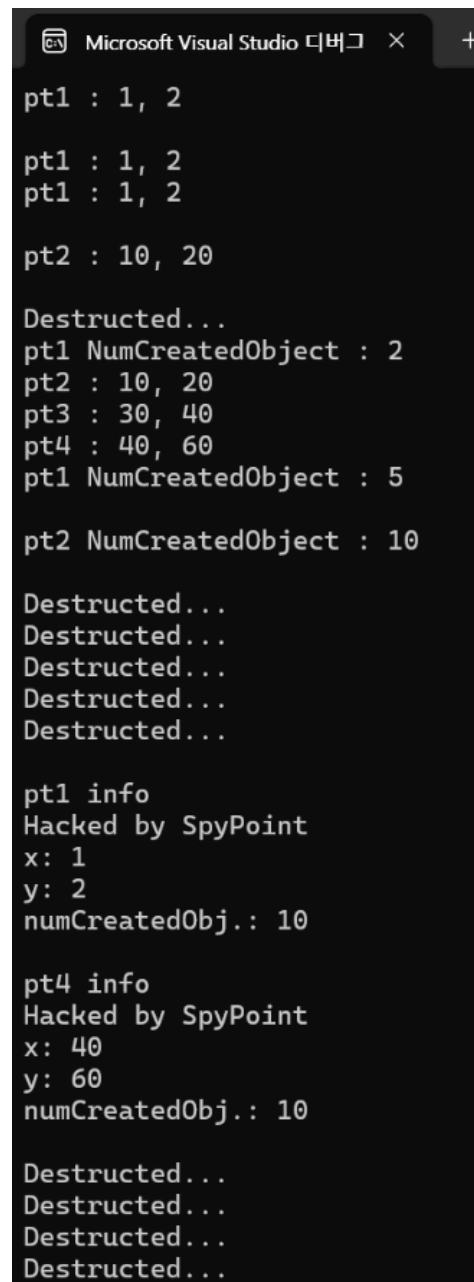
```

```

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

    return 0;
}

```



The screenshot shows the output of a C++ program running in the Visual Studio debugger. The output is displayed in a dark-themed window titled "Microsoft Visual Studio 디버그". The logs show the creation and destruction of objects, and the execution of the `hack_all_info` function on two points, `pt1` and `pt4`.

```

pt1 : 1, 2

pt1 : 1, 2
pt1 : 1, 2

pt2 : 10, 20

Destructed...
pt1 NumCreatedObject : 2
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.

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

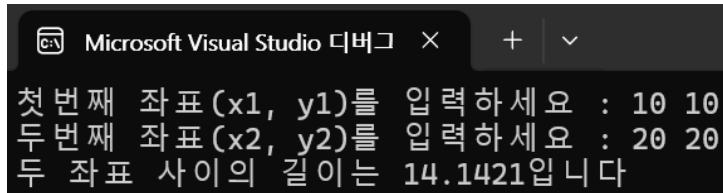
class Point {
private:
    int x, y;
public:
    Point() : x(0), y(0) {}
    Point(int x1, int y1) : x(x1), y(y1) {}
    void setPoint(int x1, int y1) {
        x = x1;
        y = y1;
    }
    int getX() const { return x;}
    int getY() const { return y;}
    Point operator-(const Point& p) {
        return Point(this->x - p.x, this->y - p.y);
    }
    Point operator*(const Point& p) {
        return Point(this->x * p.x, this->y * p.y);
    }
};

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();
    *pP3 = (*pP1 - *pP2) * (*pP1 - *pP2);
    double a = sqrt(pP3->getX() + pP3->getY());
    cout << "두 좌표 사이의 길이는 " << a << "입니다" << endl;
    delete pP1;
    delete pP2;
```

```

delete pP3;
return 0;
}

```



The screenshot shows the Microsoft Visual Studio 디버그 (Debug) window. The output text is as follows:

```

첫 번째 좌표 (x1, y1)를 입력하세요 : 10 10
두 번째 좌표 (x2, y2)를 입력하세요 : 20 20
두 좌표 사이의 길이는 14.1421입니다

```

2.

```

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

```

```

class Account {

```

```

    string name;
    string id;
    int balance;

```

```

public:

```

```

    Account(string n, string i, int b) : name(n), id(i), balance(b) {}
    string getid() const { return id; }
    int getbalance() const { return balance; }

```

```

    Account& operator--(int a) {
        balance -= a;
        return *this;
    }

```

```

    friend ostream& operator<<(ostream& o, const Account& a) {
        o << "학번: " << a.id << ", 이름: " << a.name << ", 잔액: " << a.balance;
        return o;
    }

```

```

    Account& operator+=(int a) {
        balance += a;
        return *this;
    }

```

```

};

```

```

int findAcc(Account acnt[], int s, string id) {
    for (int i = 0; i < s; i++) {
        if (acnt[i].getid() == id)
            return i;
    }
    return -1;
}

```

```

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

    while (true) {
        string sid, rid;
        cout << "돈을 보낼 학생의 학번을 입력하세요: ";
        cin >> sid;
        if (sid == "종료") {
            cout << "종료합니다.\n";
            for (int i = 0; i < 3; i++) {
                cout << acnt[i] << endl;
            }
            break;
        }
        cout << "돈을 받을 학생의 학번을 입력하세요: ";
        cin >> rid;
        int c = findAcc(acnt, 3, sid);
        int d = findAcc(acnt, 3, rid);

        if (c == -1 || d == -1) {
            cout << "보내는 학생 혹은 받은 학생의 학번이 존재하지 않습니다.다시 입력해주세요.\n";
            continue;
        }
        if (c == d) {
            cout << "학번이 동일합니다.\n";
            continue;
        }
        if (acnt[c].getbalance() == 0) {
            cout << "보내는 학생의 잔액이 부족합니다.\n";
            continue;
        }
    }
}

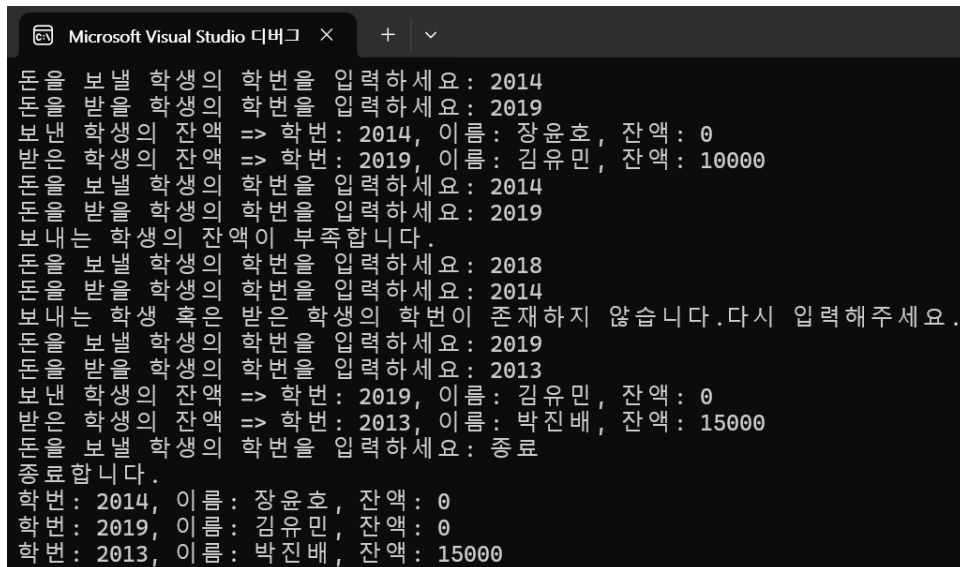
```

```

        int a = acnt[c].getbalance();
        acnt[c] -= a;
        acnt[d] += a;
        cout << "보낸 학생의 잔액 => " << acnt[c] << "\n받은 학생의 잔액 => " << acnt[d] << endl;
    }

    return 0;
}

```



```

Microsoft Visual Studio 디버그
돈을 보낼 학생의 학번을 입력하세요 : 2014
돈을 받을 학생의 학번을 입력하세요 : 2019
보낸 학생의 잔액 => 학번 : 2014, 이름 : 장윤희, 잔액 : 0
받은 학생의 잔액 => 학번 : 2019, 이름 : 김유민, 잔액 : 10000
돈을 보낼 학생의 학번을 입력하세요 : 2014
돈을 받을 학생의 학번을 입력하세요 : 2019
보내는 학생의 잔액이 부족합니다.
돈을 보낼 학생의 학번을 입력하세요 : 2018
돈을 받을 학생의 학번을 입력하세요 : 2014
보내는 학생 혹은 받은 학생의 학번이 존재하지 않습니다. 다시 입력해주세요.
돈을 보낼 학생의 학번을 입력하세요 : 2019
돈을 받을 학생의 학번을 입력하세요 : 2013
보낸 학생의 잔액 => 학번 : 2019, 이름 : 김유민, 잔액 : 0
받은 학생의 잔액 => 학번 : 2013, 이름 : 박진배, 잔액 : 15000
돈을 보낼 학생의 학번을 입력하세요 : 종료
종료합니다.
학번 : 2014, 이름 : 장윤희, 잔액 : 0
학번 : 2019, 이름 : 김유민, 잔액 : 0
학번 : 2013, 이름 : 박진배, 잔액 : 15000

```

3.

```

#include <iostream>
#include <string>
using namespace std;
class A {
    string i, n;
    int b;
    static int t;
public:
    A(string x, string y, int z) : i(x), n(y), b(z) {
        t += z;
    }
    string getI() {
        return i;
    }

    static int getT() {
        return t;
    }
}

```



```

int getbalance() {
    return b;
}

};

int A::t = 0;
int main() {
    int x;

    cout << "총 학생 수 입력: ";
    cin >> x;
    A** arr = new A * [x];
    for (int y = 0; y < x; y++) {
        string i, n;
        int b;
        cout << y + 1 << "번째 학생 계좌 입력 : 학번 : ";
        cin >> i;
        cout << endl;
        for (int z = 0; z < y; z++) {
            if (arr[z]->getI() == i) {
                cout << "중복된 학번" << endl;

                for (int w = 0; w < y; w++) {
                    delete arr[w];
                }

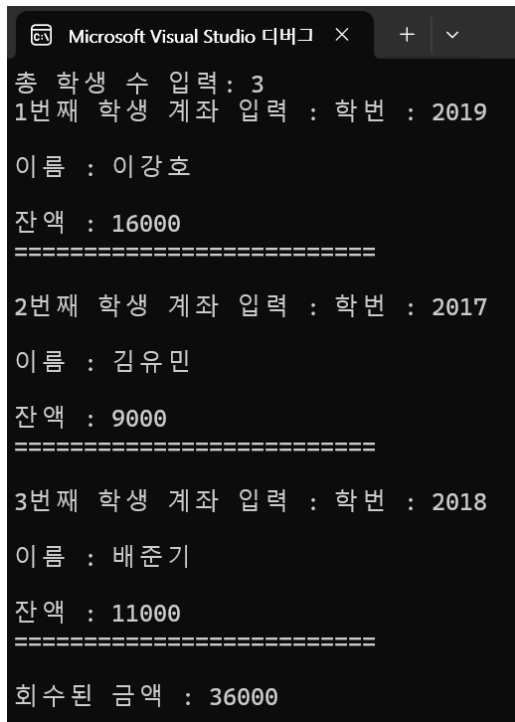
                delete[] arr;
                return 0;
            }
        }

        cout << "이름 : ";
        cin >> n;
        cout << endl;
        cout << "잔액 : ";
        cin >> b;
        arr[y] = new A(i, n, b);
        cout << "===== " << endl << endl;
    }

    int total = 0;
    for (int y = 0; y < x; y++) {
        total += arr[y]->getbalance();
        delete arr[y];
    }
}

```

```
delete[] arr;  
cout << "환수된 금액 : " << total << endl;  
return 0;  
}
```



The screenshot shows the Microsoft Visual Studio Debug Console with a dark background and light gray text. The output of the program is as follows:

```
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
=====

환수된 금액 : 36000