

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

    // int _x 와 int _y를 입력으로 받는 생성자
    /*구현*/

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

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

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

    // *this + pt2 ->
    Point operator+(Point& pt2) {
        /*구현*/
    }

    //operator overloading(연산자 오버로딩)
    Point& operator=(Point& pt) {
        /*구현*/
    }

    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)
/*구현*/
```

```

//객체 call by reference시: const로 함수 입력시 const method만 함수에서 사용가능
// const: 객체 내부의 member data가 상수(변하지 않는다)
void print(/*구현*/) {
    cout << pt.x << ", " << pt.y << endl;
}

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

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*/
};

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 : ";
    /*구현*/
    // pPt1를 통해 호출 getX, getY를 호출하여 출력
    cout << "pt1 : ";
    /*구현*/

    cout << endl;

    //동적으로 Point* pPt2할당하여 10,20넣은 뒤 ->사용하여 출력(pt1 출력 참고)
    /*구현*/
    cout << "pt2 : ";
    /*구현*/
    cout << endl;
}

```

```

//pPt1, pPt2의 메모리 해제
/*구현*/

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

// 연산자 오버로딩
//pt4 = pt2, pt3값 더하기
/*구현*/
cout << "pt2 : ";
cout << pt2 << endl;
cout << "pt3 : ";
cout << pt3 << endl;
cout << "pt4 : ";
cout << pt4 << endl;

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

// object array
Point* ptAry = /*구현*/;
cout << "pt2 NumCreatedObject : ";
cout << /*구현*/ << endl;
cout << endl;

// ptAry 메모리 해제
/*구현*/

cout << endl;

// friend class
SpyPoint spy;
cout << "pt1 info" << endl;
/*구현*/
cout << "pt4 info" << endl;
/*구현*/

return 0;
}

```

```
pt1 : 1, 2

pt1 : 1, 2
pt1 : 1, 2

pt2 : 10, 20

Destructed...
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. 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

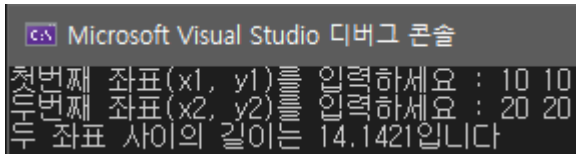
    /* 아래의 방식으로도 x, y값 설정이 가능해야 함 */
    //pP1->setPoint(x1, y1);
    //pP1->setPoint(x2, y2);
    /*******/

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

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

    return 0;
}
```

1-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
```

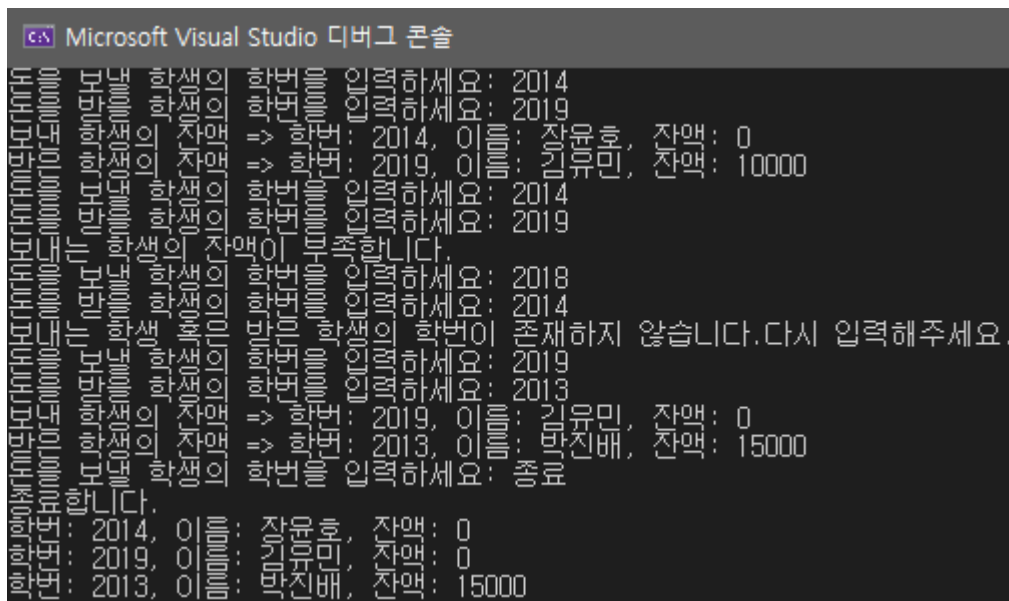
2. 학생들끼리 송금하는 프로그램을 작성하시오.

- 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 */
}
```

2-출력화면:



```
Microsoft Visual Studio 디버그 콘솔
학생의 학번을 입력하세요: 2014
학생의 학번을 입력하세요: 2019
=> 학번: 2014, 이름: 장윤희, 잔액: 0
=> 학번: 2019, 이름: 김유민, 잔액: 10000
학생의 학번을 입력하세요: 2014
학생의 학번을 입력하세요: 2019
학생의 잔액이 0입니다.
학생의 학번을 입력하세요: 2018
학생의 학번을 입력하세요: 2014
학생의 학번이 존재하지 않습니다. 다시 입력해주세요.
학생의 학번을 입력하세요: 2019
학생의 학번을 입력하세요: 2013
=> 학번: 2019, 이름: 김유민, 잔액: 0
=> 학번: 2013, 이름: 박진배, 잔액: 15000
학생의 학번을 입력하세요: 종료
합니다.
: 2014, 이름: 장윤희, 잔액: 0
: 2019, 이름: 김유민, 잔액: 0
: 2013, 이름: 박진배, 잔액: 15000
```

3. 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.

3-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
```

Week 9 Optional

※ **Optional** 문제는 성적 산출에 반영되지 않습니다.

기반 지식

우리는 개체 생성에 있어 생성자, 연산자 오버로딩, 소멸자 등의 기본적인 구조를 배웠으며, 기존 수업에서 동적 할당과 같은 개념을 배웠다. 그렇다면 이 지식을 이용해서 직접 개체를 구현해보도록 할 것이다.

문제

문제 개요

계산기 회사에 앞으로 유저가 사용했던 숫자들을 저장해서 재활용을 하려고 한다. 이를 위한 저장 수단을 개발하려고 한다. 배열을 사용하기엔 유저가 얼마나 많은 숫자를 사용할지 몰라, 동적으로 배열의 크기가 변하는 자료구조를 개발하려고 한다. 해당 자료구조는 앞으로 팀 내의 다른 프로그래머들이 사용해야하기 때문에 개발자들을 위해 제대로 잘 만들도록 하자.

드디어 신입에서 팀의 핵심으로 도약하는 순간이다. 단순 표면적인 기능 구현에서 전체 팀에서 사용할 툴을 개발하는 것이다. 열심히 하도록 하자.

문제 설명

당신이 개발해야 하는 클래스는 `IntVector` 클래스이다. 해당 클래스는 동적으로 저장 공간이 바뀌는 클래스로, 실제 `#include <vector>`에 정의된 `std::vector<T>`와 동일하게 작동한다. 일종의 `std::vector<int>`와 동일한 클래스가 되어야한다.

당신이 개발해야 하는 멤버함수는 다음과 같다:

기본 멤버 함수	
----------	--

생성자1 - Default Constructor	유저가 사용이 불가능하게 처리
생성자2 - 생성자	공간의 용량을 입력으로 받아 해당 용량의 벡터를 생성
생성자3 - 복사생성자	복사생성자
소멸자	소멸자
할당연산자	할당연산자
용량 관련 함수	
GetSize	벡터의 크기를 반환
Resize	벡터의 크기를 입력으로 받아 벡터의 크기 수정
GetCapacity	벡터의 용량을 반환
IsEmpty	벡터가 비어있는지 여부를 반환
Reserve	벡터의 용량을 입력으로 받아 벡터의 용량을 수정
원소 참조 함수	
참조연산자	주어진 subscript/index에 위치한 원소를 반환. 주어진 subscript/index의 값이 범위를 벗어날 경우 assert를 활용하여 에러를 낸다
수정 함수	
PushBack	벡터의 마지막 원소 뒤에 새로운 원소를 추가.
PopBack	벡터의 마지막 원소를 제거
Insert	주어진 index에 주어진 원소를 추가하고 나머지 내용물을 뒤로 옮긴다
Erase	주어진 index에 위치한 원소를 제거하고 나머지 내용물을 앞으로 땡긴다
Clear	벡터를 비운다

문제 규칙

- cassert, cstdint, iostream을 제외한 그 어떠한 헤더파일의 사용을 금한다.
- 메모리 누수가 나서는 안된다.
- 변수는 카멜 표기법을 사용한다
 - int randomInt
 - float randomFloat

■ `IntVector randomIntVector`

- 위의 명세에 주어진 멤버 함수를 제외한 그 어떠한 멤버 함수도 추가될 수 없다
- `public` 멤버 함수의 경우 파스칼 표기법을 사용한다

■ `void PublicFunction();`

- `private` 멤버 함수의 경우 카멜 표기법을 사용한다

■ `void privateFunction();`

- `bool`형 변수는 앞에 `b`를 붙인다

■ `bool blsFull;`

- 클래스 멤버 변수는 앞에 `m`을 붙인다

■ `class SomeClass`

{

`public:`

`void SomeFunction();`

`private:`

`char mSomeChar;`

`bool mbIsFull;`

};

- 루프에서 단순 반복이 아닌 의미가 있는 데이터를 사용 시 `i, j, k`가 아닌 `index, row, column`과 같은 명백한 변수명을 사용한다

- 대문자로 이루어져 있더라도 표기법 그대로 따른다

■ `int khuComputerScience;`

■ `std::string studentId;`

- `include` 전처리문과 코드 사이에는 반드시 빈 줄이 있어야한다

- 중괄호는 반드시 새로운 줄에 연다

■ `void function() {
} // X`

■ `void function()`

■ `{
} // O`

- 원칙적으로 모든 곳에 `const`를 사용한다 (당연히 수정할 변수 제외)

- 개체를 수정하지 않는 멤버 함수에도 모두 `const`를 붙여야한다

- 반환할 때 `const`를 붙일 경우, `pointer/reference`에만 `const`를 붙인다. 값에는 `const`

를 붙이지 않는다.

➤ 클래스 안에서 멤버변수/멤버함수의 순서는 다음과 같다

- 1. friend 클래스
- 2. public 멤버함수
- 3. protected 멤버함수
- 4. private 멤버함수
- 5. protected 멤버변수
- 6. private 멤버변수

➤ `const_cast`와 같이 `const`를 파괴하는 형변환은 불허한다

➤ 파일 이름은 클래스 이름인 `IntVector`와 같아야하며, 반드시 선언과 정의를 분리해야한다 (헤더와 `cpp` 분리)

➤ 디폴트 매개변수는 불허한다

➤ 매크로 정의(`#define`)는 불허한다

➤ 모든 컴파일러 경고는 고친다

➤ 포인터/참조 기호는 자료형에 붙인다

■ `int &ref; // X`

■ `int& ref; // O`

➤ 변수 가리기(variable shadowing)는 불허한다

■

```
class Point
{
public:
    int x;
    int y;
    Point(int _x, int _y); // X
};
```

■

```
class Point
{
public:
    int mX;
    int mY;
    Point(int x, int y); // O
};
```

➤ 변수는 한 줄에 하나만 선언한다

■ `int i = 0, j = 0; // X`

■ `int i = 0;`

```
int j = 0; // O
```

- 모든 반환형이 있는 멤버함수는 단 한 번만 반환한다
- 멤버 변수 초기화는 초기화 리스트 사용이 기본이며, 한 줄에 한 변수를 초기화한다
 - `Point::Point(int x, int y)`
 `: mX(x), mY(y) // X`
 {
 - `Point::Point(int x, int y)`
 `: mX(x)`
 `, mY(y) // O`
 {
- 포인터엔 `NULL` 대신 `nullptr`를 사용한다
- 어떤 개체인지 프로그래머 입장에서 명확하지 않거나, 더 나아가 컴파일러 입장에 서도 명확하지 않다면 `auto`의 사용을 절대 금한다

[참고 링크](#)

예제 코드

```
int main()
{
    // IntVector iVec1; // 컴파일 오류
    IntVector iVec2(5); // 용량: 5, 크기: 0
    IntVector iVec3 = iVec2; // 용량: 5, 크기: 0
    iVec3 = iVec3; // 용량: 5, 크기: 0

    iVec2.GetSize(); // 0
    iVec2.Resize(5); // 크기: 5
    iVec2.IsEmpty(); // false
    iVec3.Reserve(3); // 용량: 3
    iVec3.GetCapacity(); // 3

    // iVec3[0]; // 런타임 오류
    iVec2[0]; // 0
    iVec2[iVec2.GetSize() - 1]; // 0

    iVec3.PushBack(1); // 1
    iVec3.PushBack(2); // 1 2
    iVec3.PushBack(3); // 1 2 3
    iVec3.PushBack(4); // 1 2 3 4
    iVec3.PopBack(); // 1 2 3
    iVec3.Insert(0, 5); // 5 1 2 3
    // iVec3.Insert(6, 6); // 런타임 오류
    iVec3.Insert(3, 7); // 5 1 2 7 3
    iVec3.Erase(3); // 5 1 2 3
```

```
iVec3.Erase(0); // 1 2 3  
iVec3.Clear(); //
```

```
return 0; // 메모리 누수 x
```

```
}
```