문제 정의

1. 계산할 정수 2개를 먼저 입력하고, 그 다음 연산자를 쓰는 계산기를 만든다.
2. 코드에는 사칙연산을 수행하는 총 4개의 클래스(Add, Sub, Mul, Div)가 존재한다.
3. void setValue(int x, int y): 두 개의 정수 x와 y를 a와 b에 할당하는 함수를 사용한다.
4. int calculate(): a와 b의 합을 계산하고 그 결과를 반환하는 함수를 사용한다.
5. 이 프로그램은 무한 루프로 돌아간다.
6. main() 함수에 4개의 클래스 타입의 객체 a, s, m, d를 생성한다.

문제 해결 방법

1. 멤버 함수의 정의를 사용하여 해결하면 어떨까? - 정민수
2. if 함수로 다양한 연산자를 받으면 어떨까? - 정재현
3. 예전에 배운 switch case를 사용하여 연산자를 받으면 어떨까? - 정민수
4. 문제에는 없지만 무한 루프를 탈출하는 것도 넣어야 할까? - 정재현

아이디어 평가

1. 처음에는 클래스 안에 연산을 실행하는 int calculate() 함수, 문제의 조건을 보지 못하고 멤버 함수의 정의까지 생성하는 불편함 발생, 결국 2번의 과정까지 한 다음 정의 삭제 후 재생성

```
int Add::getAdd() {
    return a + b;
}

int Sub::getSub() {
    return a - b;
}

int Mul::getMul() {
    return a * b;
}

int Div::getDiv() {
    return a / b;
}
```

```
class Add {
public:
    int a;
    int b;
    int getAdd();
};
```

2. if문으로 main() 함수에서 연산자를 받기 실행, 정상 동작 완료.

```
if (z == '+')
    cout << "결과: " << add.getAdd() << endl;
else if (z == '-')
    cout << "결과: " << sub.getSub() << endl;
else if (z == '*')
    cout << "결과: " << mul.getMul() << endl;
else if (z == '/')
    cout << "결과: " << div.getDiv() << endl;
else
    cout << "오류" << endl;
```

3. 다른 언어에서 사용했던 Switch case문이 C++에서도 사용 가능함을 확인, 문제의 조건에 따라 멤버 함수의 정의를 삭제하며 만들었던 if문으로 연산자 받기 과정을 Switch문으로 수정.

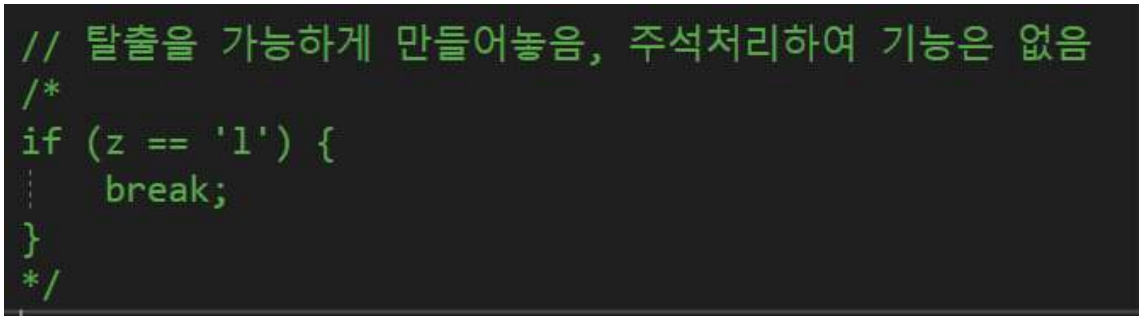
```
switch (z) {
case '+':
    a.setValue(x, y);
    cout << "결과: " << a.calculate() << endl;
    break;
case '-':
    s.setValue(x, y);
    cout << "결과: " << s.calculate() << endl;
    break;
case '*':
    m.setValue(x, y);
    cout << "결과: " << m.calculate() << endl;
    break;
case '/':
    d.setValue(x, y);
    cout << "결과: " << d.calculate() << endl;
    break;
}
```

~.setValue(x, y) & ~.calculate() -> 입력받은 두 값을 멤버에 복사 후 연산 실행

상호 평가
정재현: 예전에 배운 Switch case 문을 떠올리면서 복습하는 느낌이라 좋았다. 보기도 더 좋은 것 같으니 이걸로 하면 좋을 것 같다.

4. 무한 루프의 탈출은 문제 조건에는 없기 때문에 만들지 않으려고 하였으나, 그래도 만든 후 주석 처리하여 기능을 동작하게 하지는 않았지만, 무한 루프를 종료할 수 있는 수단 추가. 처음 숫자 2개를 아무거나 입력 후, 연산자 부분에서 알파벳 l을 입력하고 엔터를 누르면 무한 루프의 탈출이 가능하다. 반복문 안쪽에 if 함수로 넣어놓았다.

상호 평가
정민수: 문제의 조건에는 없어서 주석처리를 하지만, 무한루프의 탈출을 만들어놓는 것은 좋은 생각인 것 같다.



문제를 해결한 키 아이디어 또는 알고리즘

처음엔 멤버 함수의 정의를 사용하여 코딩하였는데, 문제의 조건을 다시 읽어보니 안쪽에 void setValue() 함수와 int calculate() 함수가 들어있었다. 수업시간에는 멤버 함수의 정의로 했어서 그대로 적용했던 건데, 문제의 조건을 보고 잠시 헷갈렸지만, "멤버 함수의 정의가 클래스의 안쪽에 들어있다"라는 것을 금방 눈치채고 수정하였다.

또한, 연산자를 받는 곳에서 if문을 사용하기에 지장은 없지만 무언가 정돈되지 못한, 지저분해 보이는 경향이 살짝 있다. 그래서 Switch case 문을 사용하였는데 이 점을 우리 조는 스스로 좋게 평가한다.