

[lab10 보고서]

자율전공학과

2019111677

김지연

1) 직원클래스

1. 소스코드

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

class Employee {
private:
    string name;
    int salary;
public:
    Employee() {
        name = "Unknown";
        salary = 0;
    }

    Employee(string name, int salary) {
        this->name = name;
        this->salary = salary;
    }

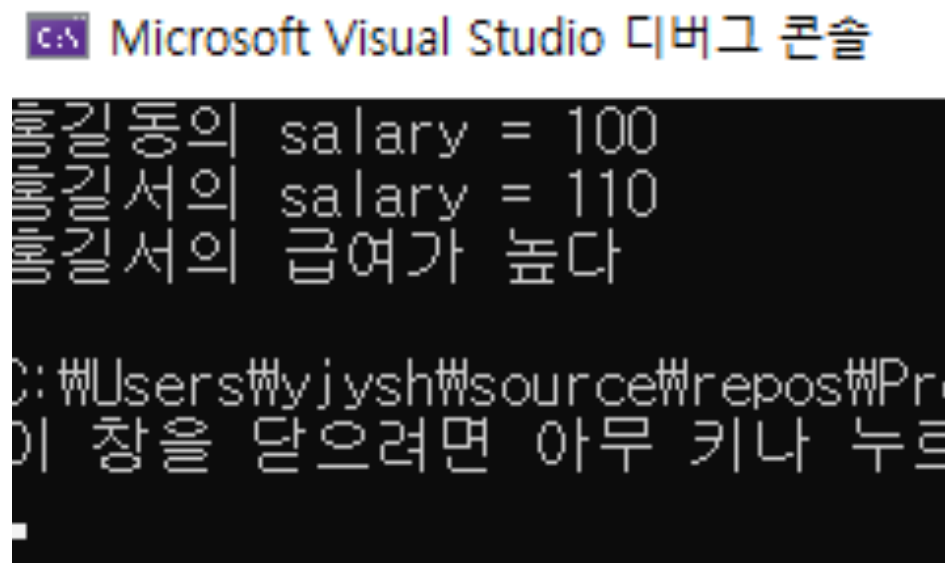
    operator int()const {           //변환생성자(Employee->int)
        return salary;
    }

    Employee(int salary) {          //변환생성자(int->Employee)
        this->salary = salary;
        this->name = "unknown";
    }

    bool operator>(const Employee& e2) {
        Employee e = 0;             //변환생성자(int->Employee) 필요함
        e.salary = this->salary;
        if (e.salary > e2.salary) { return true; }
        else { return false; }
    }
};

int main()
{
    Employee e1("홍길동", 100);
    Employee e2("홍길서", 110);
    int salary = e1;                 //변환생성자(Employee->int)
    필요
    cout << "홍길동의 salary = " << salary << endl;
    cout << "홍길서의 salary = " << e2 << endl;
    if (e1 > e2)
        cout << "홍길동의 급여가 높다" << endl;
    else
        cout << "홍길서의 급여가 높다" << endl;
    return 0;
}
```

2. 실행결과화면



```
C:\ Microsoft Visual Studio 디버그 콘솔  
홍길동의 salary = 100  
홍길서의 salary = 110  
홍길서의 급여가 높다  
C:\Users\jysh\source\repos\Pr  
이 창을 닫으려면 아무 키나 누르
```

3. 문제 정의 및 분석

정수형 변수에 객체를 넣으려고 했기 때문에 오류 생긴다.

변환 연산자 함수를 정의하여 문제를 해결한다.

객체에 정수값을 넣으려고 하였기 때문에 오류가 생긴다.

변환 연산자 함수 정의하여 문제를 해결한다.

객체와 객체의 크기를 비교(급여비교)하는 연산자가 없기 때문에 오류가 생긴다.

이를 해결하기 위해 참과 거짓을 계산하는 bool 타입과 >연산자를 오버로딩한다.

2) 날짜클래스

1. 소스코드

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

class Date {
    int year;
    int month;
    int day;
public:
    friend ostream& operator<<(ostream& os, const Date& d);
    Date() {
        year = 0;
        month = 0;
        day = 0;
    }
    Date(int y, int m, int d) {
        year = y;
        month = m;
        day = d;
    }
    ~Date() {}

    Date operator++() {
        Date d(0, 0, 0);
        d.day = day + 1; //기본적으로 다음날의 날짜는 day만 바뀐. day만
+1을 하고 나머지는 그대로.
        d.month = month;
        d.year = year;
        if (day == 31) {
            d.month = month + 1;
            d.day = 1;
            if (month == 12) { d.month = 1; d.year = year + 1; };
        }
        return d;
    }

    Date operator--() {
        Date d(0, 0, 0);
        d.day = day - 1;
        d.month = month;
        d.year = year;
        if (day == 1) {
            d.month = month - 1;
            d.day = 31;
            if (month == 1) { d.month = 12; d.year = year - 1; };
        };
        return d;
    }
};

ostream& operator<<(ostream& os, const Date& d) {
```

```

    os << d.year << "년" << d.month << "월" << d.day << "일" << endl;
    return os;
}

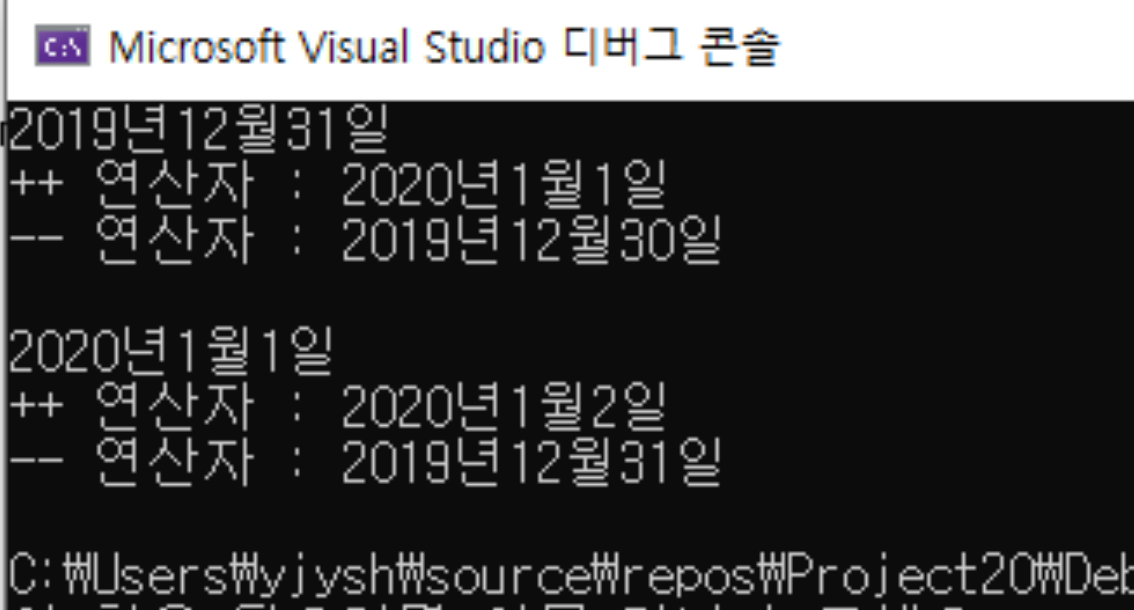
int main()
{
    Date d1(2019, 12, 31);
    cout << d1;
    cout << "++ 연산자 : ";          //2020년 1월1일.
    cout << ++d1;
    cout << "-- 연산자 : ";          //2019년 12월 30일
    cout << --d1;
    cout << endl;

    Date d2(2020, 1, 1);
    cout << d2;
    cout << "++ 연산자 : ";          //2020년 1월2일
    cout << ++d2;
    cout << "-- 연산자 : ";          //2019년 12월31일.
    cout << --d2;

    return 0;
}

```

2. 실행결과화면



```

C:\ Microsoft Visual Studio 디버그 콘솔
2019년12월31일
++ 연산자 : 2020년1월1일
-- 연산자 : 2019년12월30일

2020년1월1일
++ 연산자 : 2020년1월2일
-- 연산자 : 2019년12월31일

C:\Users\y\jysh\source\repos\Project20\Deb

```

3. 문제 정의 및 분석

++과 --연산자를 오버로딩 한다.

++과 --는 단항연산자이기 때문에 멤버함수로 선언하였을 때 매개변수가 없다.

++연산자에서 day 가 31 이면 month 가 1 늘어나고, day 는 1 로 바뀐다.

또한, day 가 31 이고 month 가 12 일 경우에는 day 는 1 로 month 도 1 로 year 은 1 이 늘어나기 때문에 if 문 안에 이 식을 포함시킨다.

--연산자의 경우도 day 가 1 이면 month 가 1 줄어들고 day 는 31 로 바뀐다.

또한, day 가 1 이고 month 도 1 인 경우에는 day 는 31 로 month 는 12 로 year 은 1 이 줄어들기 때문에 ++연산자처럼 if 문 안에 이 식을 포함시켜야 한다.

<<연산자를 오버로딩 해준다.

3) 포인터 연산자

1. 소스코드

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

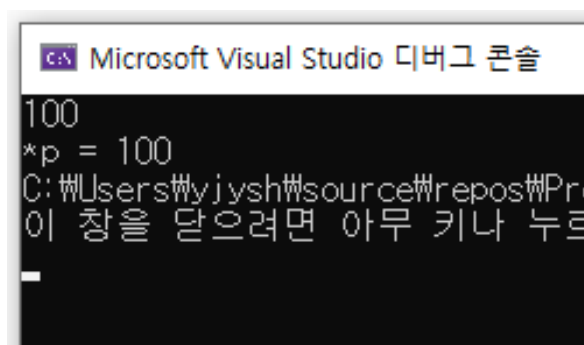
class Pointer {
    int* p;

public:
    Pointer(int* p):p(p) {};
    ~Pointer() { delete p; };
    int & operator*() {
        return *p;
    }
    void print() {
        cout << "*p = " << *p;
    }
};

int main()
{
    Pointer p(new int);

    *p = 100;
    cout << *p << endl;
    p.print();
    return 0;
}
```

2. 실행결과화면



3. 문제 정의 및 분석

간접참조연산자를 오버로딩 하는 문제

우선 생성자와 소멸자를 생성한다.

반환형은 int 연산자는 *, 이에 맞추어서 연산자를 중복정의한다.