实验报告

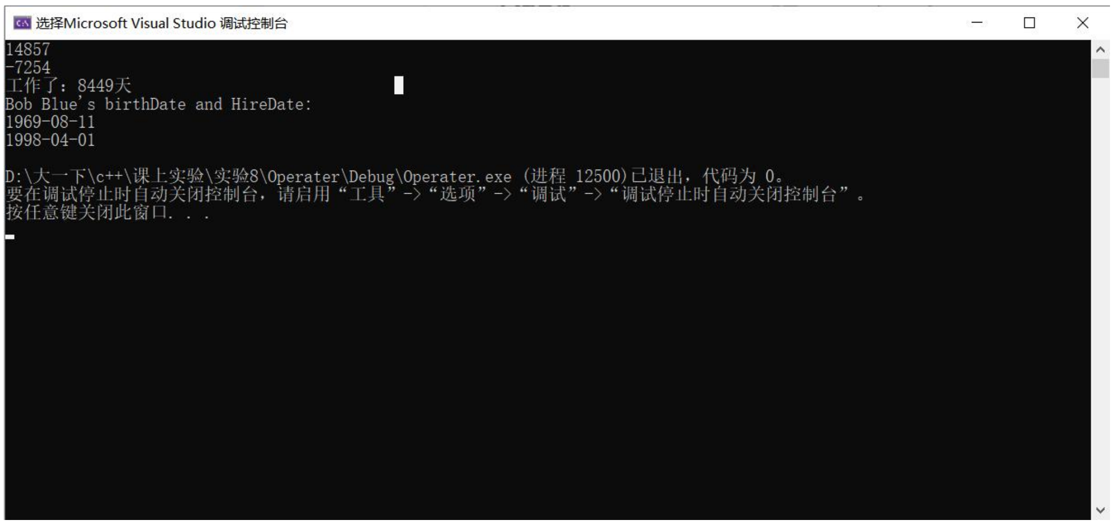
代码：

```
#include<iostream>
#include<iomanip>
#include<string>
using namespace std;
class Date {
public:
    /* 默认构造函数，以 fullyear 的形式给出年月日，默认值为 1990 年 1 月 1 日，同时设置日期
分隔符为“-” */
    Date(int year = 1990, int month = 1, int day = 1) {
        this->year = year;
        this->day = day;
        this->month = month;
    }
    /* get、set 方法 */
    // 设置日期，如果有非法的月或日，将其置为 1
    void setDate(int year, int month, int day) {
        this->year = year;
        this->day = day;
        this->month = month;
    }
    void setYear(int year) {
        this->year = year;
    }
    int getYear() {
        return this->year;
    }
    void setMonth(int month) {
        this->month = month;
    }
    int getMonth() {
        return this->month;
    }
    void setDay(int day) {
        this->day = day;
    }
    int getDay() {
        return this->day;
    }
    void setSeparator(char separator) {
        this->separator = separator;
    }

    /* 输出函数，请使用 setfill( ‘0’ )和 setw(2)，需要包含<iomanip>头文件 */
    void printFullYear() {
        cout << this->year << this->separator << setfill('0') << setw(2) << this->month
<< this->separator << setfill('0') << setw(2) << this->day << endl;
    }       // 以 YYYY-MM-DD 的形式打印，2011-01-08
    void printStandardYear() {
        cout << this->year % 100 << this->separator << setfill('0') << setw(2) <<
this->month << this->separator << setfill('0') << setw(2) << this->day << endl;
    } // 以 YY-MM-DD 的形式打印，比如 11-01-08
    /*  计算函数  */
    // 计算当前日期与参数日期之间相差几个整年，仅考虑参数日期比当前日期晚的情况
    int fullYearsTo(Date& date) {
        int wholeYear = date.year - this->year;
        if (this->month < date.month) {
            //cout << wholeYear;
        }
        else
        {
            wholeYear--;
            //cout << wholeYear;
        }
        return wholeYear;
        cout << " 满" << wholeYear << "岁了" << endl;
    };
    /* 计算当前日期与参数日期之间相差多少天(考虑闰年)，如果参数日期在当前日期之前，返回
负数。 */
    int daysTo(Date& date) {
        int run = 0;
        int days = 0;
        if (date.year < this->year)
        {
            for (int i = date.year; i <= this->year; i++)
            {
                if (i % 400 == 0 || i % 4 == 0 && i % 100 != 0) {
                    run++;
                }
            }
            days = wholeMonth[month] - date.day;
            for (int i = date.month + 1; i < 13; i++)
            {
                days += wholeMonth[i];
            }
            //cout << days<< endl;
            days += (this->year - date.year - 1) * 365 + run;
            for (int i = 1; i < this->month; i++)
            {
                days += wholeMonth[i];
            }
            days += this->day;
            //cout << "-" << days << endl;
            return -days;
        }
        else
        {
            for (int i = date.year; i >= this->year; i--)
            {
                if (i % 400 == 0 || i % 4 == 0 && i % 100 != 0) {
                    run++;
                }
            }
            //int wholeMonth[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };
            days = wholeMonth[this->month] - this->day;
            for (int i = this->month + 1; i < 13; i++)
            {
                days += wholeMonth[i];
            }
            days += (date.year - this->year - 1) * 365 + run;
            for (int i = 1; i < date.month; i++)
            {
                days += wholeMonth[i];
            }
            days += date.day;
            //cout << "-" << days << endl;
            return days;
        }

    }
    int operator-(Date& date) {
        return this->daysTo(date);
    }
    int getDayOfYear() {
        int days = 0;
        for (int i = 0; i < this->month; i++)
        {
            days += wholeMonth[i];
        }
        if (month > 2)
        {
            if (isLeapyear(this->year))
            {
                days++;
            }
        }
        return days;
    }  //计算当前日期是本年的第几天
    int getLeftDaysYear() {
        if (isLeapyear(this->year))
        {
            return 366 - this->getDayOfYear();
        }
        return 365 - this->getDayOfYear();
    }//计算当前日期距本年结束还有几天，不包括当前日期这一天
    bool isLeapyear(int year1) {
        if (year1 % 400 == 0 || year1 % 4 == 0 && year1 % 100 != 0) {
            return true;
        }
        return false;
    }//断参数年是否是闰年。
    bool operator>(const Date otherDate) {
        if (this->year > otherDate.year)
        {
            return true;
        }
        else if (this->month > otherDate.month) {
            return true;
        }
        else if (this->day > otherDate.day) {
            return true;
        }
        else
        {
            return false;
        }
    }
    bool operator<(const Date otherDate2) {
        if (this->year < otherDate2.year)
        {
            return true;
        }
```

```
        else if (this->month < otherDate2.month) {
            return true;
        }
        else if (this->day < otherDate2.day) {
            return true;
        }
        else
        {
            return false;
        }
    }
private:
    int year;
    int month;
    int day;
    int yearString;
    char separator = '-';  // 日期分隔符;
    static int wholeMonth[13];
};
int Date::wholeMonth[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };
class Employee {
public:
    //构造函数，使用“成员初始化器”初始化数据成员
    Employee(string firstName, string lastName, Date& birthDate, Date& hireDate) {
        this->firstName = firstName;
        this->lastName = lastName;
        this->birthDate = birthDate;
        this->hireDate = hireDate;
    }
    //打印员工的信息。调用 Date 类的 print 函数，打印员工的生日和雇佣日期。
    void print() {
        cout << this->firstName << ' ' << this->lastName << "'s birthDate and HireDate:
" << endl;
        this->birthDate.printFullYear();
        this->hireDate.printFullYear();
    }
    //计算员工在参数指定的日期时，满多少岁。请使用 Date 类的 fullYearsTo 函数
    int getAge(Date& date) {
        return this->birthDate.fullYearsTo(date);
    }
    //计算该员工在参数指定的日期时，工作满了多少年。
    int getYearsWorked(Date& date) {
        return this->hireDate.fullYearsTo(date);
    }
```

```cpp
    //计算该员工在参数指定的日期时，工作了多少天。使用 Date 类的 daysTo 函数。
    int getDaysWorked(Date& date) {
        //return -(date - this->hireDate);
        return (this->hireDate.daysTo(date));
    }
    static const Employee& getMostFaith(const Employee employees[], int n) {
        Employee max = employees[0];
        Date today(2021, 5, 19);
        for (int i = 0; i < n; i++)
        {
            if (max.hireDate > employees[i].hireDate) {
                max = employees[i];
            }
        }
        cout <<"工作了：" <<max.getDaysWorked(today) <<"天"<< endl;
        max.print();
        return max;
    }
    //~Employee();   //析构函数
private:
    string firstName;
    string lastName;
    Date birthDate;    //内嵌对象，出生日期
    Date hireDate;     //内嵌对象，雇用日期
};
void main() {
    Date birth(1969, 8, 11);
    Date hire(1998, 4, 1);
    Date today(2010, 4, 30);
    Employee manager("Bob", "Blue", birth, hire);
    /*2.1 测两组数据
        第一组 t1 1969 8 11; t2 2010 4 15 结果 14857
        第二组 t1 1969 8 11; t2 1949 10 1 结果 - 7254

        2.2 及 2.3 测一组数据 名字，birth 自己随便设置
        today 2021 5 19
        以下为 hire
        e1 1998 4 1
        e2 1999 8 15
        e3 2010 4 30
        e4 2010 4 1
        e5 2021 12 31*/
    Date t1(1969, 8, 11);
    Date t2(2010, 4, 15);
    Date t3(1949, 10, 1);
    cout << t1 - t2 << endl;
    cout << t1 - t3 << endl;
    Date e1Hire(1998, 4, 1);
    Date e2Hire(1999, 8, 15);
    Date e3Hire(2010, 4, 30);
    Date e4Hire(2010, 4, 1);
    Date e5Hire(2021, 12, 31);
    Employee e1("Bob", "Blue", birth, e1Hire);
    Employee e2("Bob", "Blue", birth, e2Hire);
    Employee e3("Bob", "Blue", birth, e3Hire);
    Employee e4("Bob", "Blue", birth, e4Hire);
    Employee e5("Bob", "Blue", birth, e5Hire);
    Employee testArray[5] = { e1,e2,e3,e4,e5 };
    Employee::getMostFaith(testArray, 5);
    //answer.print();
}
```

截图：



```
14857
-7254
工作了：8449天
Bob Blue's birthDate and HireDate:
1969-08-11
1998-04-01

D:\大一下\c++\课上实验\实验8\Operater\Debug\Operater.exe (进程 12500)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```