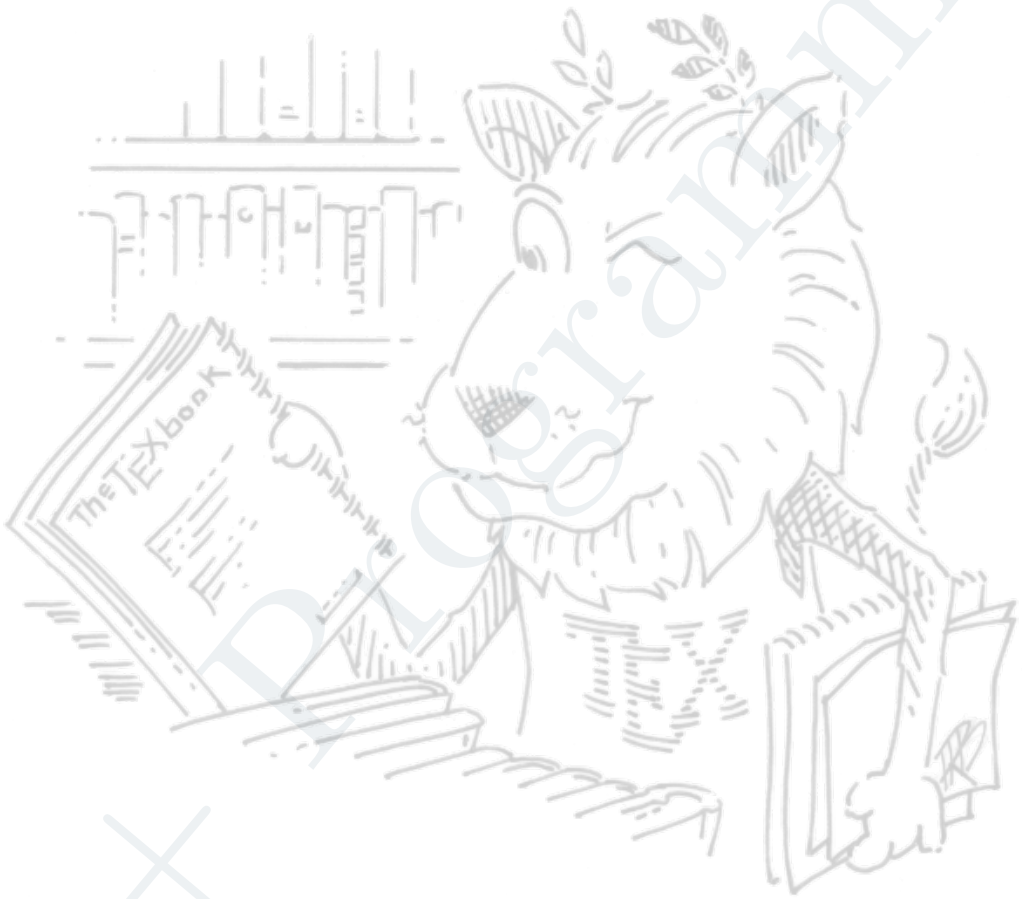
《C++ 程序设计》OJ2（参考答案）

VCG

2023 年 5 月 27 日



1 清单类

【描述】

已知日期类的定义如下：

```
class Date  //日期类
{
private:
    int year;    //年
    int month;   //月
    int day;     //日
public:
    Date(int year = 2000, int month = 1, int day = 1);
    void show(); //以“年-月-日”格式输出年月日
    ~Date();
};
```

```
class Item  //清单类;含有日期类的对象
{
private:
    std::string name;
    Date birthday;
public:
    Item();
    Item(const std::string & name, int year, int month, int day);
    Item(const std::string & name, const Date & date);
    void show();//显示姓名和出生日期
};
int main()
{
    int choice;
    while (std::cin >> choice)
    {
        int y, m, d;
        std::string name;
        if (choice == 0)
        {
            Item stu0;
            stu0.show();
        }
        else if (choice == 1)
        {
            std::cin >> name >> y >> m >> d;
            Item stu1(name, y, m, d);
            stu1.show();
        }
        else if (choice == 2)
        {
            std::cin >> name >> y >> m >> d;
            Date m_date(y, m, d);
```

```
            Item stu2(name, m_date);
            stu2.show();
        }
        else if (choice == -1)
            return 0;
    }
    return 0;
}
```

要求实现以上两个类，并在主函数中进行测试。【注意，上述代码默认存在，提交代码时不能包括以下代码】

要求输入为多组数据：

(1) 输入为 0 的时候，直接使用 Item 类的无参构造函数（即第 1 个构造函数）实例化对象，并输出：Name: NULL, Birthday: 0-0-0；

(2) 当输入为 1 时，继续输入姓名和年月日，使用 Item 类的含有 4 个参数的构造函数（即第 2 个构造函数）实例化对象，并进行输出，详见输出样例 2；

(3) 当输入为 2 时，继续输入姓名和年月日，使用 Item 类的含有 2 个参数的构造函数（即第 3 个构造函数）实例化对象，并进行输出，详见输出样例 3；

(4) 当输入为-1 时，退出程序

【输入描述】

输入为 0，则不继续输入；输入为-1 则退出程序。输入为 1 或 2 则继续输入姓名和年月日。

【输出描述】

见测试样例

【输入样例】

```
0
```

【输出样例】

```
Name: NULL, Birthday: 0-0-0
```

【输入样例 2】

```
1
Jack
1998
10
1
```

【输出样例 2】

```
Name: Jack, Birthday: 1998-10-1
```

【输入样例 3】

```
2
Mary
2000
1
1
```

【输出样例 3】

```
Name: Mary, Birthday: 2000-1-1
```

【提示】

冒号和逗号后各有一个空格

main.cpp

```cpp
#include<iostream>
#include<string>

class Date  //日期类
{
private:
    int year;   //年
    int month;  //月
    int day;    //日
public:
    Date(int year = 2000, int month = 1, int day = 1);
    void show(); //以“年-月-日”格式输出年月日
    ~Date();
};

Date::Date(int year, int month, int day)
{
    this->year = year;
    this->month = month;
    this->day = day;
}

void Date::show()
{
    std::cout << year << "-" << month << "-" << day << std::endl;
}

Date::~Date()
{}

class Item  //清单类
{
private:
    std::string name;
    Date birthday;
public:
    Item();
    Item(const std::string& name, int year, int month, int day);
    Item(const std::string& name, const Date& date);
    void show();//显示姓名和出生日期
};

Item::Item() :birthday(0, 0, 0)
{
    name = "NULL";
}

Item::Item(const std::string& name, int year, int month, int day)
```

```
49      : birthday(year, month, day)
50  {
51      this->name = name;
52  }
53
54  Item::Item(const std::string& name, const Date& date)
55      : birthday(date)
56  {
57      this->name = name;
58  }
59
60  void Item::show()
61  {
62      std::cout << "Name: " << name << ", Birthday: ";
63      birthday.show();
64  }
65
66  int main()
67  {
68      int choice;
69      while (std::cin >> choice)
70      {
71          int y, m, d;
72          std::string name;
73          if (choice == 0)
74          {
75              Item stu0;
76              stu0.show();
77          }
78          else if (choice == 1)
79          {
80              std::cin >> name >> y >> m >> d;
81              Item stu1(name, y, m, d);
82              stu1.show();
83          }
84          else if (choice == 2)
85          {
86              std::cin >> name >> y >> m >> d;
87              Date m_date(y, m, d);
88              Item stu2(name, m_date);
89              stu2.show();
90          }
91          else if (choice == -1)
92              return 0;
93      }
94      return 0;
95  }
```

2 圆周长

【描述】

编写一个圆形类 Circle，实现半径的输入、周长的计算和输出。使用如下 main 函数对程序进行测试

```
int main()
{
    double r;
    cin >> r;
    Circle ci(a);
    cout << ci.Perimeter();
    return 0;
}
```

【输入描述】

半径

【输出描述】

周长（小数点后两位）

【输入样例】

```
3.21484
```

【输出样例】

```
20.20
```

【提示】

其中 PI=acos(-1.0)
圆周率的取值需要比较精确，以保证计算结果的精度
控制精度可能需要以下代码

```
#include<iostream>
#include<iomanip>

cout<<setiosflags(ios::fixed)<<setprecision(2);
```

main.cpp

```cpp
#include<iostream>
#include<iomanip>
#include <cmath>

double PI = acos(-1.0);
class Circle
{
private:
    double r;
public:
    Circle(double r=0.0){this->r=r;};
    double Perimeter(){return 2*PI*r;}
};

int main()
{
    double r;
    std::cin >> r;
    Circle ci(r);
    std::cout << std::setiosflags(std::ios::fixed) << std::setprecision(2);
    std::cout << ci.Perimeter();
    return 0;
}
```

3 点和线

【描述】

点和线是平面几何的基础。请实现点 (Point2) 以及线段 (Line2) 类，并计算线段长度 double Line2::Length(); 完成以上类，并通过以下测试

```
int main()
{
    Point2 pt1(1.0, 1.0);
    Point2 pt2(3.0, 4.0);
    Line2 line(pt1, pt2);
    cout << line.Length() << endl;

    double x1,y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    Line2 nLine(Point2(x1, y1), Point2(x2, y2));
    cout << nLine.Length()<< endl;

    return 0;
}
```

【输入描述】

两个点的 x,y 坐标

【输出描述】

线段长度

【输入样例】

```
1.0 3.0 2.0 6.0
```

【输出样例】

```
3.60555
3.16228
```

【提示】

main.cpp

```cpp
#include <iostream>
#include <cmath>

class Point2
{
private:
    double x, y;
public:
    Point2(){}
    Point2(double xx, double yy): x(xx), y(yy){}
    double X()const {return x;}
    double Y()const {return y;}
};

class Line2
{
private:
    Point2 pt1, pt2;
public:
    Line2(){}
    Line2(const Point2& pt11, const Point2& pt22): pt1(pt11), pt2(pt22){}
    double Length(){
        double dx = pt1.X()-pt2.X();
        double dy = pt1.Y()-pt2.Y();
        double dis2 = dx*dx+dy*dy;
        double dis = sqrt(dis2);
        return dis;
    }
};


int main()
{
    Point2 pt1(1.0, 1.0);
    Point2 pt2(3.0, 4.0);
    Line2 line(pt1, pt2);
    std::cout << line.Length() << std::endl;

    double x1,y1, x2, y2;
    std::cin >> x1 >> y1 >> x2 >> y2;
    Line2 nLine(Point2(x1, y1), Point2(x2, y2));
    std::cout << nLine.Length()<< std::endl;

    return 0;
}
```

4 Input and Output

【描述】

设计一个保存你个人信息的类，包含姓名和年龄。并使用以下代码测试

```
int main()
{
    string name;
    int year;
    cin >> name >> year;
    PersonInfo info(name, year);
    cout << "I am " << info.Name() << ", " << info.Age() << " years old.\n";
    return 0;
}
```

【输入描述】

姓名
年龄

【输出描述】

见测试样例

【输入样例】

```
Master
999
```

【输出样例】

```
I am Master, 999 years old.
```

【提示】

1. 使用类
2. ”,” 后面有一个空格，”.” 结束，没有其它格式输出

main.cpp

```cpp
#include<iostream>
#include<string>

class PersonInfo
{
private:
    std::string name;
    int age;
public:
    PersonInfo(const std::string& name = "", int age = 0)
    {
        this->name = name;
        this->age = age;
    }
    std::string Name()const {return name;}
    int Age()const {return age;}
};

int main()
{
    std::string name;
    int year;
    std::cin >> name >> year;
    PersonInfo info(name, year);
    std::cout << "I am " << info.Name() << ", " << info.Age() << " years old.\n";
    return 0;
}
```

5 正方形面积与周长

【描述】

编写一个正方形类 Square，实现边长的输入、面积和周长的计算和输出。使用如下 main 函数对程序进行测试

```
int main()
{
    double a;
    cin >> a;
    Square sq(a);

    cout << "Area: "<< sq.Area() << "\nPerimeter: "<< sq.Perimeter();
    return 0;
}
```

【输入描述】

边长

【输出描述】

面积
周长

【输入样例】

```
10.2
```

【输出样例】

```
Area: 104.04
Perimeter: 40.8
```

【提示】

main.cpp

```cpp
#include<iostream>

class Square
{
private:
    double a;
public:
    Square(){a=0.0;};
    Square(double a){this->a=a;};
    double Area(){return a*a;}
    double Perimeter(){return 4*a;}
};

int main()
{
    double a;
    std::cin >> a;
    Square sq(a);

    std::cout << "Area: "<< sq.Area() << "\nPerimeter: "<< sq.Perimeter();
    return 0;
}
```

6 计算机

【描述】

计算机包含 CPU 和硬盘。请设计 Computer、CPU、Disk 类。并满足以下测试

```
int main()
{
    std::string cpuType, diskType;
    double  frequency, mount;
    std::cin >> cpuType >> frequency >> diskType >> mount;

    CPU cpu(cpuType, frequency);
    Disk disk(diskType, mount);
    Computer computer(cpu, disk);
    computer.Print();

    return 0;
}
```

【输入描述】

CPU 类型 CPU 主频
Disk 类型 Disk 容量

【输出描述】

见测试样例

【输入样例】

```
i7 2.9
ST 2
```

【输出样例】

```
The computer has a CPU and a disk.
CPU type: i7, CPU frequency: 2.9 GHz
disk type: ST, disk mount: 2 T
```

【提示】

main.cpp

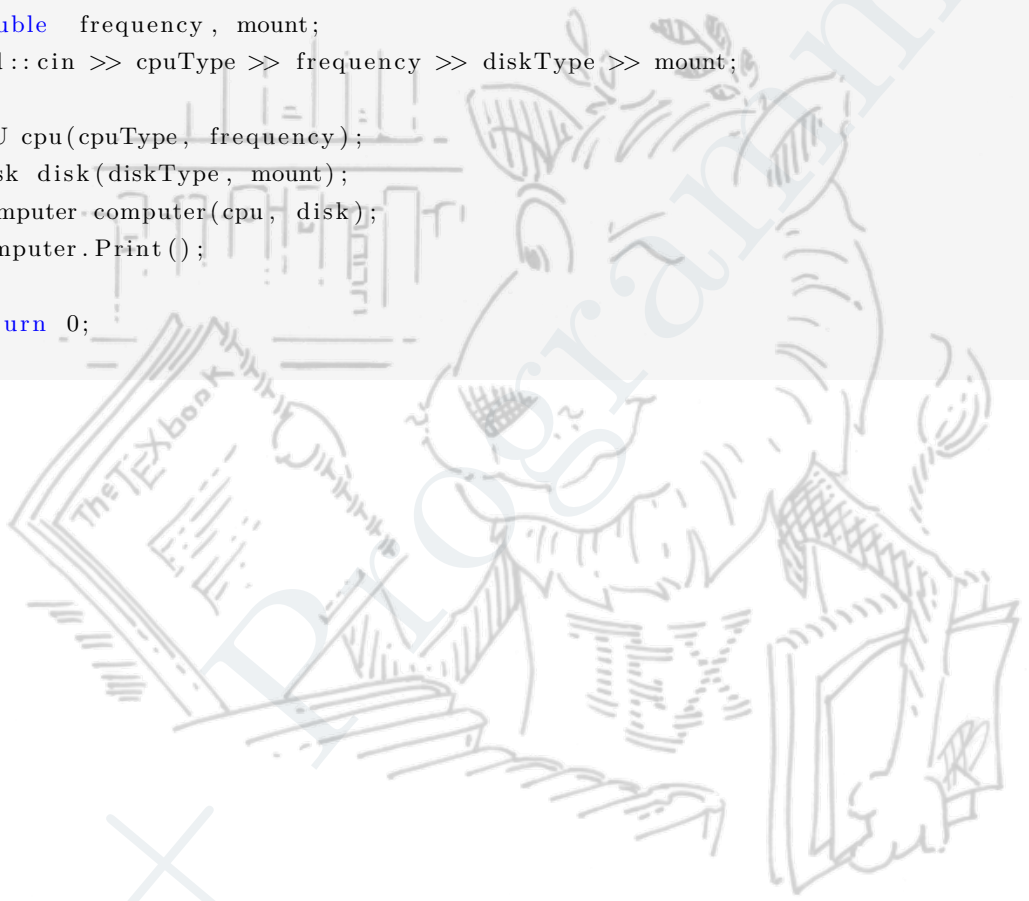
```cpp
#include<iostream>
#include<string>

class CPU
{
private:
    std::string type;
    double frequency;
public:
    CPU(const std::string& type ="", const double frequency = 0)
    {
        this->type = type;
        this->frequency = frequency;
    }
    void Print()const
    {
        std::cout << "CPU type: " << type << ", CPU frequency: " << frequency << "
            GHz" << std::endl;
    }
};

class Disk
{
private:
    std::string type;
    double mount;
public:
    Disk(const std::string& type="", const double mount = 0)
    {
        this->type = type;
        this->mount = mount;
    }
    void Print()const
    {
        std::cout << "disk type: " << type << ", disk mount: " << mount << " T" <<
            std::endl;
    }
};

class Computer
{
private:
    CPU cpu;
    Disk disk;
public:
    Computer(const CPU& cpu, const Disk& disk)
    {
        this->cpu = cpu;
```

```
47          this->disk = disk;
48      }
49      void Print()const
50      {
51          std::cout << "The computer has a CPU and a disk.\n";
52          cpu.Print();
53          disk.Print();
54      }
55  };
56
57  int main()
58  {
59      std::string cpuType, diskType;
60      double  frequency, mount;
61      std::cin >> cpuType >> frequency >> diskType >> mount;
62
63      CPU cpu(cpuType, frequency);
64      Disk disk(diskType, mount);
65      Computer computer(cpu, disk);
66      computer.Print();
67
68      return 0;
69  }
```



7 N 维向量

【描述】

向量类 VectorN 的数据成员描述如下

```
class VectorN{
private:
    int      dim;          //向量维数
    double   *values;      //具体数值
};
```

double VectorN::Mag() 为向量的模，为各分量平方和之后再开根号。使得测试函数

```
int main()
{
    int num;
    cin >> num;
    double *dtemp = new double[num];
    for(int i=0; i<num; i++)
        cin >> dtemp[i];
    VectorN vn2(num, dtemp);
    delete[] dtemp;
    cout << vn2.Mag() << endl;

    return 0;
}
```

【输入描述】

小数的个数
小数

【输出描述】

见测试样例

【输入样例】

```
4
1.0 2.0 3.0 4.0
```

【输出样例】

```
5.47723
```

【提示】

main.cpp

```
#include<iostream>
#include<cmath>

class VectorN
{
private:
    int dim;
    double *values;
public:
    VectorN()
    {
        dim = 0;
        values = 0;
    }

    VectorN(int d, const double *v)
    {
        dim = d;
        values = new double[dim];
        for(int i=0; i<dim; i++)
        {
            values[i] = v[i];
        }
    }

    ~VectorN()
    {
        if(values)delete[] values;
    }

    double Mag()const
    {
        double sum = 0.0;
        for(int i=0; i<dim; i++)
        {
            sum += values[i] * values[i];
        }
        return double(sqrt(sum));
    }
};

int main()
{
    int num;
    std::cin >> num;
    double *dtemp = new double[num];
    for(int i=0; i<num; i++)
        std::cin >> dtemp[i];
```

```
49      VectorN vn2(num, dtemp);
50      delete[] dtemp;
51      std::cout << vn2.Mag() << std::endl;
52      return 0;
53  }
```

8 日期类

【描述】

给定日期类的声明

```
#include<iostream>

class Date
{
public:
    Date(int d = 0, int m = 0, int y = 0); //构造函数
    int get_day() const; // 返回day
    int get_month() const; //返回month
    int get_year() const; // 返回year
    static void set_default(int, int, int);
    static int get_default_day(); //返回缺省day
    static int get_default_month(); //返回缺省month
    static int get_default_year(); //返回缺省year
    Date & add_year(int n); //加n年
    Date & add_month(int n); //加n月，考虑超过12月，考虑闰年
    Date & add_day(int n); //加n天，考虑进位月和年，考虑闰年
private:
    int day, month, year;
    static Date default_date; //初始化为 1901年1月1日
};
```

实现这个类并用以下的 main 函数测试它。

```
int main()
{
    std::string type;
    while (std::cin >> type)
    {
        if (type == "Date")
        {
            int day, mon, year;
            std::cin >> day >> mon >> year;
            Date mydate(day, mon, year);

            int addday, addmon, addyear;
            std::cin >> addday >> addmon >> addyear;
            mydate.add_day(addday).add_month(addmon).add_year(addyear);
            std::cout << mydate.get_day() << " " <<
                mydate.get_month() << " " << mydate.get_year() << std::endl;
        }
        else if (type == "defaultDate")
        {
            std::cout << Date::get_default_day() << " " <<
                Date::get_default_month() << " " << Date::get_default_year() << std
                    ::endl;
        }
```

```
23          else if (type == "setdefaultDate")
24          {
25              int day, mon, year;
26              std::cin >> day >> mon >> year;
27              Date::set_default(day, mon, year);
28              std::cout << Date::get_default_day() << " " << Date::get_default_month
                    ()
29                  << " " << Date::get_default_year() << std::endl;
30          }
31      }
32      return 0;
33  }
```

【输入描述】

多组输入，每一组输入以一个 type 来判断输入类型
如果是'Date'，输入当前设置的日期以及要加的天数;
如果是'defaultDate'，输出默认日期;
如果是'setdefaultDate'，输入要设置的默认日期。

【输出描述】

输出计算后得到的日期。

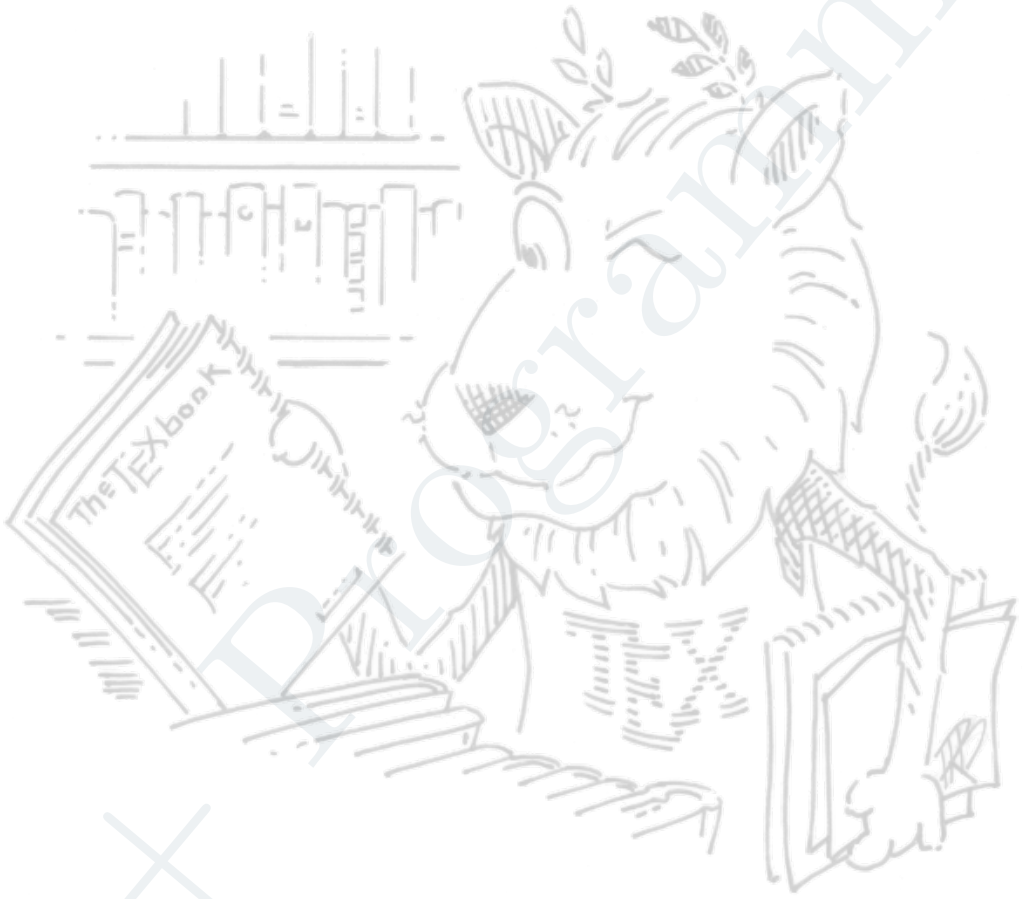
【输入样例】

```
Date 18 3 1981 2 1 1
defaultDate
setdefaultDate 1 1 1900
Date 20 4 1982 2000 50 30
Date 20 1 1990 500 50 14
Date 20 1 1990 0 0 0
Date 10 3 2016 50 20 10
setdefaultDate 24 7 1993
```

【输出样例】

```
20 4 1982
1 1 1901
1 1 1900
11 12 2021
4 8 2009
20 1 1990
29 12 2027
24 7 1993
```

【提示】



main.cpp

```cpp
#include<iostream>
#include<string>

class Date
{
public:
    Date(int d = 0, int m = 0, int y = 0); //构造函数
    int get_day() const; // 返回day
    int get_month() const; //返回month
    int get_year() const; // 返回year
    static void set_default(int, int, int);
    static int get_default_day(); //返回缺省day
    static int get_default_month(); //返回缺省month
    static int get_default_year(); //返回缺省year
    Date& add_year(int n); //加n年
    Date& add_month(int n); //加n月，考虑超过12月，考虑闰年
    Date& add_day(int n); //加n天，考虑进位月和年，考虑闰年
private:
    int day, month, year;
    static Date default_date; //初始化为 1901年1月1日
};

Date Date::default_date(1, 1, 1901);

int Date::get_default_day()
{
    return default_date.day;
}

int Date::get_default_month()
{
    return default_date.month;
}

int Date::get_default_year()
{
    return default_date.year;
}

void Date::set_default(int day = 1, int month = 1, int year = 1901)
{
    default_date.day = day;
    default_date.month = month;
    default_date.year = year;
}

int judge_leap(int m)
{
```

```
49      if (m % 400 == 0 || (m % 100 != 0 && m % 4 == 0))
50          return 1;
51      else return 0;
52  }
53
54  int judge_month(int n,int m)
55  {
56      if (n == 1 || n == 3 || n == 5 || n == 7 || n == 8 || n == 10 || n == 12)
57          return 31;
58      if (n == 4 || n == 6 || n == 9 || n == 11)
59          return 30;
60      if (n == 2)
61      {
62          if (judge_leap(m))
63              return 29;
64          else return 28;
65      }
66  }
67
68  Date& Date::add_day(int n)
69  {
70      int days_of_the_month = judge_month(this->month, this->year);
71      if (this->day + n <= days_of_the_month)
72      {
73          this->day += n;
74          return *this;
75      }
76      else
77      {
78          while (this->day + n> days_of_the_month)
79          {
80              this->month++;
81              n -= (days_of_the_month - this->day);
82              this->day = 0;
83              if (this->month > 12)
84              {
85                  this->year++;
86                  this->month -= 12;
87              }
88              days_of_the_month = judge_month(this->month, this->year);
89          }
90          this->day += n;
91          return *this;
92      }
93  }
94
95  Date& Date::add_year(int n)
96  {
97      this->year += n;
98      return *this;
```

```
99  }
100
101 Date& Date::add_month(int n)
102 {
103     if (this->month + n <= 12)
104     {
105         this->month += n;
106         return *this;
107     }
108     else
109     {
110         this->month = this->month + n - 12;
111         this->year++;
112         while (1)
113         {
114             if (this->month > 12)
115             {
116                 this->year++;
117                 this->month -= 12;
118             }
119             else break;
120         }
121         return *this;
122     }
123 }
124
125 Date::Date(int d, int m, int y)
126 {
127     this->day = d;
128     this->month = m;
129     this->year = y;
130 }
131
132 int Date::get_day() const
133 {
134     return this->day;
135 }
136
137 int Date::get_month() const
138 {
139     return this->month;
140 }
141
142 int Date::get_year() const
143 {
144     return this->year;
145 }
146
147 int main()
148 {
```

```
149     std::string type;
150     while (std::cin >> type)
151     {
152         if (type == "Date")
153         {
154             int day, mon, year;
155             std::cin >> day >> mon >> year;
156             Date mydate(day, mon, year);
157 
158             int addday, addmon, addyear;
159             std::cin >> addday >> addmon >> addyear;
160             mydate.add_day(addday).add_month(addmon).add_year(addyear);
161             std::cout << mydate.get_day() << " " <<
162                 mydate.get_month() << " " << mydate.get_year() << std::endl;
163         }
164         else if (type == "defaultDate")
165         {
166             std::cout << Date::get_default_day() << " " <<
167                 Date::get_default_month() << " " << Date::get_default_year() << std
                    ::endl;
168         }
169         else if (type == "setdefaultDate")
170         {
171             int day, mon, year;
172             std::cin >> day >> mon >> year;
173             Date::set_default(day, mon, year);
174             std::cout << Date::get_default_day() << " " << Date::get_default_month
                    ()
175                 << " " << Date::get_default_year() << std::endl;
176         }
177     }
178 
179     return 0;
180 }
```