Computing 2 - Labs

Lab 3: `Date` Class

Create a class called `Date` that includes three pieces of information as data members—a month (type `int`), a day (type `int`) and a year (type `int`). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this exercise, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1–12; if it isn't, print out an error message (for example: `cerr << "error: month must be 1-12";`) and then set the month data member to 1. Also make sure that the year is a positive number. If not, return an error message and set it to 1900. Provide a *set* and a *get* function for each data member. Provide a destructor that prints out a message saying that the object was deleted and what date it had, e.g., like this:

```
cout << "Destructor runs on Date object " << getYear() << '/' << getMonth() << '/'
<< getDay() << "\n" << endl;
```

Provide a member function `displayDate` that displays the day, month and year separated by dots (.), for example 5.6.1981 for June 5, 1981. Write a test program that demonstrates class `Date`'s capabilities.

d m y

Here is the UML diagram.

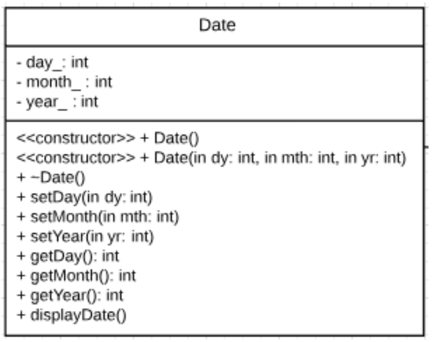
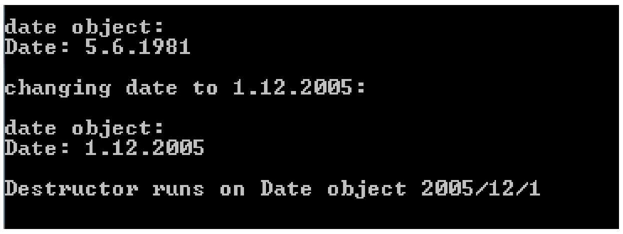


Figure 1: UML diagram for class `Date`.

Write a header file called date.h. Add the proper preprocessor directives to the header file. Implement all methods in the file date.cpp.

A driver program is provided (see main.cpp). The output of the program should, for example, look like this:



```
date object:
Date: 5.6.1981

changing date to 1.12.2005:

date object:
Date: 1.12.2005

Destructor runs on Date object 2005/12/1
```

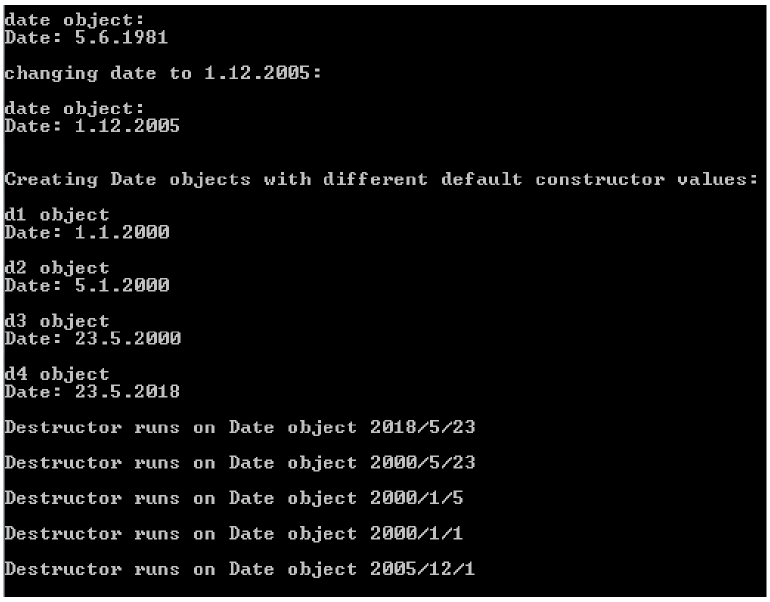
Figure 2: Output of the driver program provided.

If you have time left, set up a constructor with default values (`mth = 1, dy=1, yr = 2000`). Then follow the example in class and create four objects initialized as

1. `1.1.2000`
2. `dy.1.2000,`
3. `dy.mth.2000`
4. `dy.mth.yr`

default value must be define at the function definition.

The variables `dy, mth` and `yr`, are day, month, and year, respectively. They can be specified in the driver program. The output of the modified main function should look like:



```
date object:
Date: 5.6.1981

changing date to 1.12.2005:

date object:
Date: 1.12.2005


Creating Date objects with different default constructor values:

d1 object
Date: 1.1.2000

d2 object
Date: 5.1.2000

d3 object
Date: 23.5.2000

d4 object
Date: 23.5.2018

Destructor runs on Date object 2018/5/23

Destructor runs on Date object 2000/5/23

Destructor runs on Date object 2000/1/5

Destructor runs on Date object 2000/1/1

Destructor runs on Date object 2005/12/1
```

Figure 3: Output of the driver program with additional code for testing a constructor with default arguments.

Additional Problem

If you have still some time, you may want to work on a class `Person` displayed in the UML diagram below. Currently, it has two member variables to store the date of birth (`dob_`) and date of death (`dod_`), respectively. The UML diagram shows that it also has a member variable of class `Date`. This is called composition. It is a “life and death” relationship, because if the parent, here an object of class `Person`, is deleted, all child objects will also be deleted. They have the same life-cycle. Implement the header file person.h and the source file person.cpp.

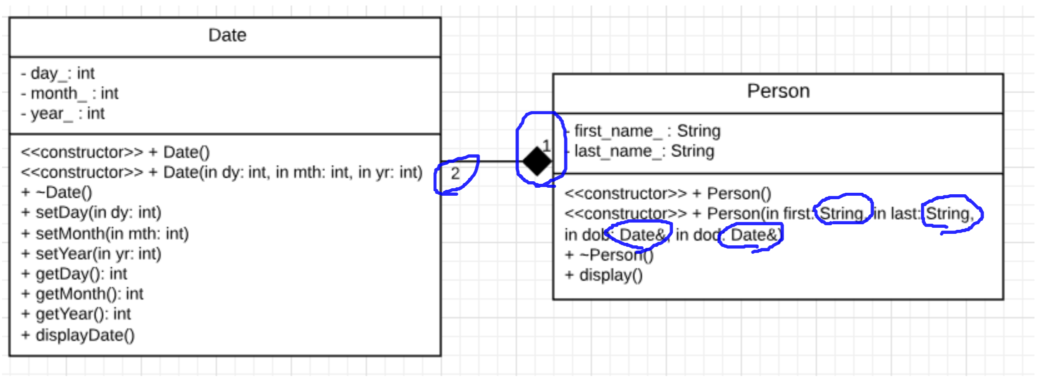


Figure 4: This UML diagram shows that the class `Person` is composed with two data members of class `Date`. From `Date`, only one `Person` is seen. Due to the composition relationship, `Date` objects must not belong to any other `Person` object at the same time. Note that we do not list the `Date` data members explicitly in the UML class diagram for class `Person`. It is implicitly understood that they belong there. You can see this, e.g., in the second constructor declaration.