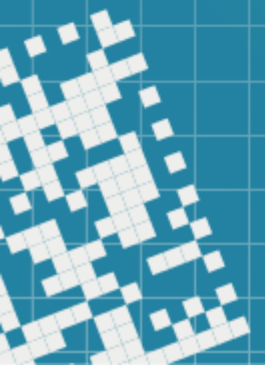
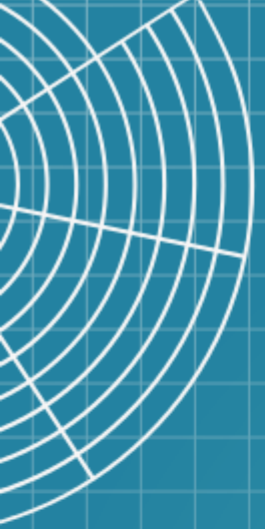
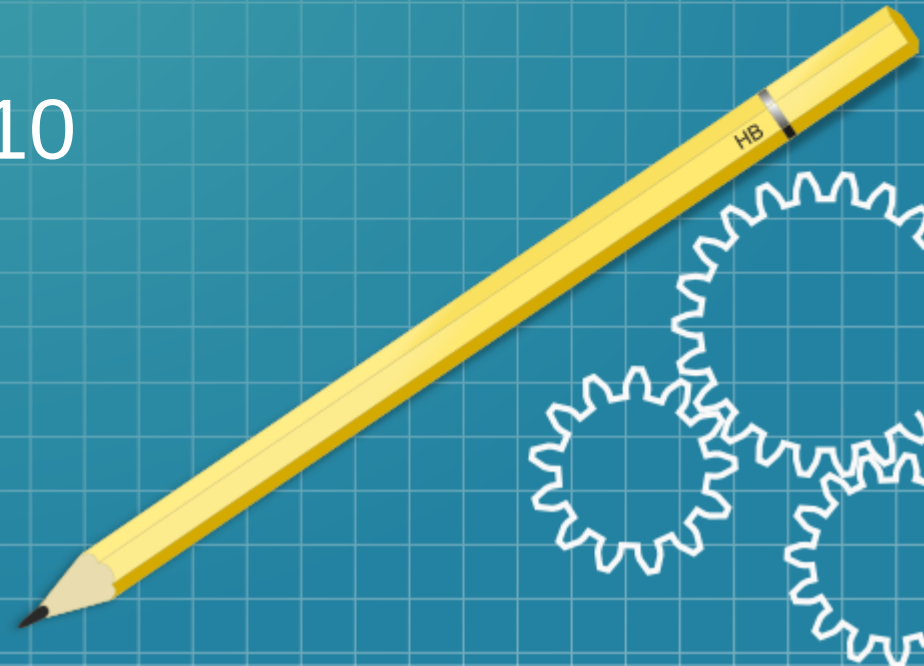


Operator overloading

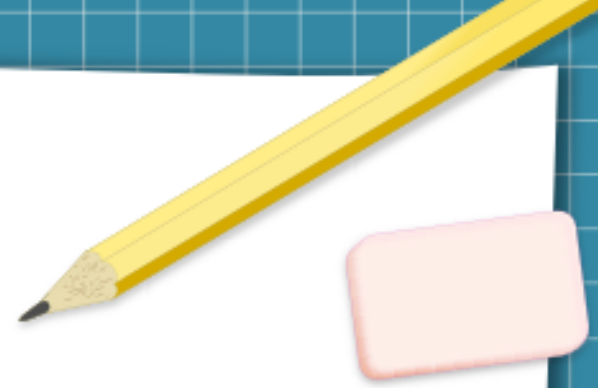
Class string

(part 1)

Chapter 10

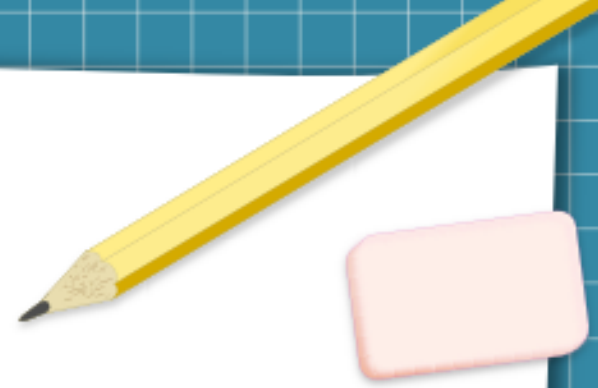


Today we will



- Talk about *operator overloading*
- work with strings using overloaded operators
- define `Date` class

Operator Overloading



Recall a statement:

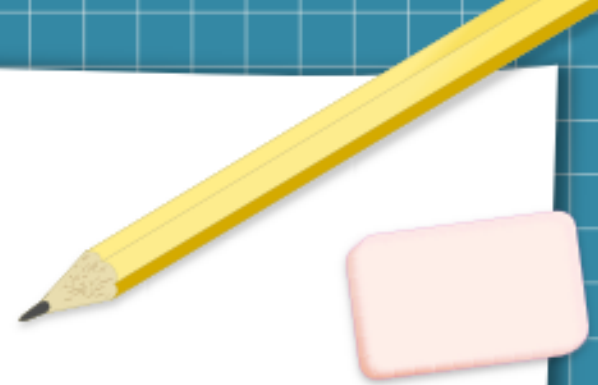
```
cout << "Hello, world!" ;
```

Object `cout` is an instance of class `ostream`, *standard output stream*.

operator `<<` is overloaded to work with such objects (as *stream insertion* operator).

Also, operator `<<` can be used for the *bitwise left-shift*.

Operator Overloading



Recall a statement:

```
cout << "Hello, world!" ;
```

Object `cout` is an instance of class `ostream`, *standard output stream*.

operator `<<` is overloaded to work with such objects (as *stream insertion* operator).

Also, operator `<<` can be used for the *bitwise left-shift*.

Similarly, operator `>>` is also overloaded, it can work as

- *stream extraction* operator (`cin >> a;`), and
- *bitwise left-shift* operator

Operator Overloading



See the work with strings in [stringsWork.cpp](#)

Operator Overloading

We can overload *most* operators to be used with class objects, the compiler will generate the appropriate code based on the *types* of the operands.

Of course we can use *explicit function calls* (recall `empty()`), but *operator notation* is often more natural.

Let's build `class Date`: each object of the class will be representing a date (month-day-year). We will need the following methods:

- display the date
- increment a date
- add a number of days to the date
- check if the year is a leap year
- ... anything else that come during the development

Static members of a class

We discussed *static variables*:
they keep their values and are not destroyed when they go
out of scope.




Static members of a class

Static member variables are shared by all the instances of the class, they do not belong to a particular object.

- we can access them through the class objects or through the class name.
- since they are not part of the individual class objects (they are treated similarly to global variables, and get initialized when the program starts), we must explicitly define the static member outside of the class, in the global scope.
- static member definition is not subject to access controls: we can define and initialize the value even if it's declared as private (or protected) in the class.
- If the class is defined in a `myClass.h` file, the static member definition is usually placed in the associated code file for the class (e.g. `myClass.cpp`).
- If the class is defined in a `.cpp` file, the static member definition is usually placed directly underneath the class.

Static members of a class



```
class myClass {  
public:  
    myClass(int a = 10) : x{ a } { }  
    static int y; // static member variable  
private:  
    int x;  
};  
  
int myClass::y{ 0 };  
  
int main() {  
    myClass obj1{ 23 }, obj2;  
  
    obj1.y = 90;  
    obj2.y = -23;  
    cout << myClass::y << endl;  
}
```

see [staticMembersOfClass.cpp](#)

Static members of a class

Static member functions/methods are shared also by all the instances of the class and do not belong to a particular object.

- we can access them through the class objects or through the class name directly (`myClass::`)
- since they are not part of the individual class objects, they have no `this` pointer
- they can directly access other static members (variables or functions), but
- they cannot access non-static members

see `staticMembersOfClass_2.cpp`

Operator Overloading

Let's start working on the `class Date`: each object of the class will be representing a date (month-day-year). We will need the following methods:

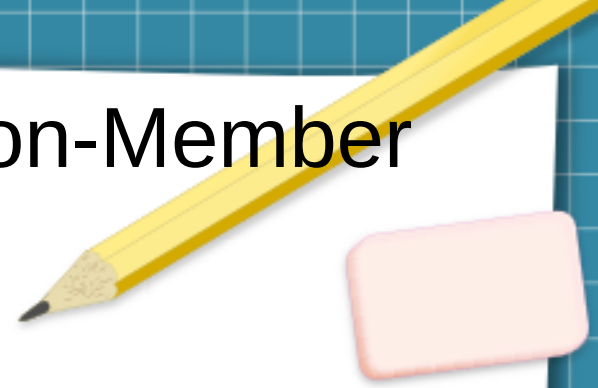
- display the date
- increment a date
- add a number of days to the date
- check if the year is a leap year
- ... anything else that come during the development

see file `Date.h`

It follows up with `Date.cpp`

- we haven't finished working on it.

Increment/Decrement – Member/Non-Member



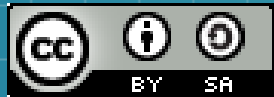
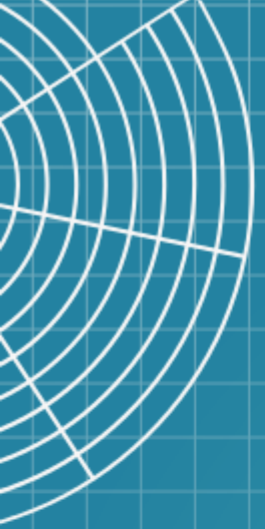
HW assignment

1) will be posted shortly

Suggested exercises

(not for grade, but the questions related to these will appear on a quiz or a test):

1) Chapter 10, Summary and all Self-Review Exercises



This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Unported License.
It makes use of the works of Mateus Machado Luna.

