

Programming in C/C++

Exercises 6

1. Implement a class "MyTime" for simple time management. It shall have to following features:

- The constructor shall accept up to 3 int parameters in the order: hour, minute, second. If a parameter is not given 0 is assumed. If second or minute are larger than 59 it should be adjusted to the interval [0;59] and the next time element (i.e. minute or hour) should be increased accordingly. Example: "0,0,75" will result in "0,1,15" internally. There is no restriction for the hour. You do not need to check for negative values.
- The methods getHour(), getMinute(), getSecond() return the hour, minute and second of a MyTime object as int.
- If a MyTime object is converted to an int it shall return the number of seconds since 0:0:0.
- It shall be possible to add two MyTime objects using "+" and to add a MyTime object and an int in any order using "+".
- It shall be possible to add a MyTime object or an int to another MyTime object using "+=".
- When adding MyTime objects, seconds are added to seconds, minutes to minutes and hours to hours. When adding an int and a MyTime object, the int is interpreted as seconds. You can assume that only positive ints are added (but do not use "unsigned" as parameter type). The resulting time shall always be in correct bounds as defined for the constructor.
- It shall be possible to print a MyTime object using <<, e.g. "std::cout << time;".

Use the "const" modifier wherever possible. You can add private methods to your class if needed.

Submit your solution as "mytime.cpp" and "mytime.hpp", since we test it with our own test program. Not respecting the naming of files and classes results in deduction of points!

Add a small report to answer the following question: Why is it not necessary to overload comparison operators to compare two time objects?

(code 30 pts, comments 5 pts, report 5 pts)

2. Implement a class "SecArr" for secure access to an int array. The array is external, i.e. it should NOT be stored in the class. The class shall have the following features:

- The constructor accepts pointers to the first and the last element of the array. If the end pointer is before the start pointer it prints a warning and sets the end pointer to the start pointer. Example:

```
int a[3] = {1,2,3};
SecArr sptr(a, &a[2]);
```
- The class provides access to the current array element using the * operator, e.g., "*sarr = 3".
- Moving the current element with ++ and -- (post- and pre-form) shall be possible. If the current position would be moved out of the bounds defined in the constructor it prints a warning and ignores the command.

- Access to an element relative to the current position shall be possible with [], e.g. "sarr[1] = 2" sets the element after the current position. If the designated element would be out of the bounds defined in the constructor it prints a warning and returns the last element of the array.

Submit your solution as "secarr.cpp" and "secarr.hpp", since we test it with our own test program. Not respecting the naming of files and classes results in deduction of points!

Add a small report to answer the following question: Why is it better to use pre-increment/pre-decrement instead of post-increment/post-decrement operators?

(code 45 pts, comments 10 pts, report 5 pts)

3. Bonus task. Consider the following program:

```
#include <iostream>
int main(void) {
    std::cout << "Hello world!\n";
}
```

Modify it to produce the output

```
---start---
Hello world!
---end---
```

without changing the main() function, using preprocessor statements, redefining classes, etc.. Use operator overloading only! (code 10 pts)