Institut für Informatik — TUM
Scientific **C**omputing in **C**omputer **S**cience
R. Wittmann

WT 2015

Tutorial (Advanced Programming) Worksheet 8:

In this worksheet we will introduce some specific properties of C++ with respect to Object Oriented Programming (OOP). First, we will start to wrap a piece of allocated memory into a simple object. Second, we extend this object by some useful operators to make it more convenient to use. Download the stub implementation `vector_access.cpp` from moodle and modify it according to the following assignments.

Assignment 1: Access rules for information

Recall the different available storage types from the previous worksheet. Define a class `OurVector` which shall manage heap allocated memory with the following properties:

- When the object gets constructed it takes two arguments. The first argument defines the size of our `double` array, which we want to allocate on the heap. The second argument defines the initial value for all array elements.
- When the object gets destructed, it shall free the memory associated with the array.
- The size parameter shall not be directly accessible by other classes.
- Provide a safe way to access the contents of the internal array in a read only way.

Declare your internal variables using the special keywords from the lecture to control the access from inside as well as outside of `OurVector`. Further, initialize your internal variables using member initialization lists where appropriate. Then, test your code by creating a single instance using:

```
OurVector first(5,2.0)
```

Assignment 2: Controlled access from non-derived classes

Now, we define a second class `VectorOperations` which will provide some operations on `OurVector` instances. The first operation `add` takes three arguments:

- The first two input arguments shall be references to constant `OurVector` objects.
- The third argument is a reference to a non constant `OurVector` object.
- The two input vectors shall be added together and the result shall be stored in the output argument.

The second operation `print` is defined in the following way:

- It takes a single argument which is a reference to a constant `OurVector` object.
- It prints out all values in the internal array of the provided `OurVector` object.

These operations shall now be implemented in a way that they do not need an instance of `VectorOperations`. Further, you are not allowed to change properties of your variables in `OurVector` and you are not allowed to add new methods to class `OurVector`. Your existing code from Assignment 1 should still work after you have changed the `OurVector` class.
Test your code using the existing instance `first` from Assignment 1 as your input to the first two arguments of the `add` method. For the corresponding output argument use a new instance of `OurVector` of size 5, which shall be initialized with 0.0. Then, use the `print` operation to check if the result is 4.0 as expected.

Assignment 3: Implicit / Explicit Operators

Uncomment the lines 16 to 29 in the stub implementation from the moodle and run the program with your own class implementations. You will notice that the program will crash in line 19 and line 25.

- Explain this behaviour and describe the default implementation of the compiler for these operations.
- Extend your class implementation of `OurVector` by explicit operations to restore the intended behaviour.