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

WT 2015

Tutorial
(Advanced Programming)
Worksheet 9:

Assignment 1: Inheritance and virtual methods

In this assignment, we will apply our knowledge about inheritance and virtual methods to change the design of an array management class. Please download `inheritance.cpp` from moodle and work through the following sub-tasks.

1. Allow only those classes which are derived from `FixedVector` to directly access the internal array structure. Then, implement the missing functionality in the `SequenceVector` constructor.

2. Look very carefully at the output of sub-task 2. What should the output be like and what is going wrong? Further, think about potential problems when dealing with dynamically allocated data in subclasses. Fix this issue by only changing the implementation of `AbstractVector`.

3. What would you expect from sub-task 3 and why do you get something else instead? Explain this behavior and repair the code accordingly. You are only allowed to change `FixedVector` and `AbstractVector`.

4. In sub-task 4 we want to copy the content of one vector to another vector. However, something goes horribly wrong. Study the origin of this problem and fix it. The debugging output should help you with that.

Assignment 2: VTable Implementation

Similar to one of the previous exercises where we analyzed the call stack of a function, we will use the same methods to gain a deeper understanding of the C++ VTable implementation. While the basic concepts are very similar on each platform, the actual addresses, offsets or sizes are highly dependent on your target system and platform. Use the file `find_vtable.cpp` provided on moodle as a starting point for your analysis.

1. Your first task in this reverse engineering process is to look at the object sizes of our predefined class types. What is your expectation and do you notice anything special? (e.g. what influences the size of a class?)

2. Though the classes have no visible members, they still require some specific amount of storage. Assume that there is an invisible member variable which has the same size as the whole class object. Use this size to pick the corresponding primitive datatype and print the content of this hidden

member variable. (HINT: use the address of the class object for the hidden memory variable). For classes without virtual methods, this hidden variable is only a placeholder. Hence, we can ignore them. But what about the other classes?

3. Now, focus on the classes with virtual methods. Compute the differences between the hidden member variable values of a base class and its subclasses. The hidden variable defines the address of our VTable and the computed difference is a rough approximation of its size. Assume that the VTable only contains pointers to functions. Implement the function print_hidden_table() which shall print the contents of a given table with a certain size. Then, use this function to print the VTable contents of all classes with virtual methods.

Homework Assignment 3: Call-by value / reference

Look at the following code snippet and answer the following questions:

- What is legitimate C++ code and what is not? Explain why.
- If the respective code part is correct give the corresponding output.

```
#include <iostream>
class Base { public:
    void a() { std::cout << "a" << std::endl; }
    virtual void b() { std::cout << "b" << std::endl; }
};

class Derived : public Base { public:
    void a() { std::cout << "ad" << std::endl; }
    void b() { std::cout << "bd" << std::endl; }
};

void func1(Base base) { base.a(); base.b(); }
void func2(Base& base) { base.a(); base.b(); }
void func3(Derived *der) { der->a(); der->b(); }

int main(int argc, char **argv) {
    Base c1; Derived c2;
    func1(c1); func1(c2);
    func2(c1); func2(c2);
    func3(&c1); func3(&c2);
}
```