ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 3

HS 22
Dr. R. Käppeli

Problem 3.1 The Simpson Collaboration

Work together in a team on the Simpson integration library. The idea is to learn to collaborate on a project.

1. team up with a partner, preferably one with a machine running a different OS
2. create a joint project on GitLab
3. one of you commits the main file which calls the integration routine and the other the header and source files for the Simpson integration
4. jointly write a CMake file which builds the library and links it to the main executable
5. confirm that the whole project, building and executions, works on both machines

Problem 3.2 Machine epsilon – revisited

Rewrite the program computing machine epsilon (Problem 1.4), but this time using templates. Compare your results against the standard numeric limits library (in the `<limits>` header) for types `float`, `double` and `long double`.

Problem 3.3 Penna model – part 1

Read the paper by Penna [T.J.P. Penna, J. Stat. Phys. 78, 1629 (1995)][1]. Think over the structure for a Penna model simulation. Summarize the stated concepts. What are the features which all individuals have in common? Which features are different? How would you represent an individual in your code? Do not write any code yet.

[1] The paper is available at https://doi.org/10.1007/BF02180147. You have to be inside the ETH network in order to download it.

Problem 3.4 Trafficlight

What is the output of the following code? Can you guess it without running it?

```
#include <iostream>

class Trafficlight {
public:
    enum light { green, orange, red };

    Trafficlight(light l = red) : id_(counter_++), state_(l) {
        std::cout << "Constructing " << id_ << std::endl;
    }
    Trafficlight(const Trafficlight& rhs) : id_(counter_++), state_(rhs.state_) {
        std::cout << "Copy constructing " << id_ << " from " << rhs.id_ << std::endl;
    }
    Trafficlight& operator=(const Trafficlight& rhs) {
        std::cout << "Assigning " << rhs.id_ << " to " << id_ << std::endl;
        if (this != &rhs) { state_ = rhs.state_; }
        return *this;
    }
    ~Trafficlight() { std::cout << "Destructing " << id_ << std::endl; }

    void print_state() const;

private:
    static int counter_;
    const int id_; // Unique ID
    light state_;
};

int Trafficlight::counter_ = 0;

void Trafficlight::print_state() const {
    switch (state_) {
    case Trafficlight::red:
        std::cout << "red" << std::endl;
        break;
    case Trafficlight::orange:
        std::cout << "orange" << std::endl;
        break;
    case Trafficlight::green:
        std::cout << "green" << std::endl;
        break;
    default:
        std::cout << "broken?!" << std::endl;
        break;
    }
}

int main() {
    Trafficlight a(Trafficlight::green);

    Trafficlight* b_ptr = new Trafficlight(Trafficlight::orange);

    Trafficlight c(a);

    Trafficlight d = *b_ptr;

    Trafficlight e;
    e = a;

    Trafficlight& f = a;

    a.print_state();
    b_ptr->print_state();
    c.print_state();
    d.print_state();
    e.print_state();
    f.print_state();

    delete b_ptr;

    return 0;
}
```