Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 4

HS 22
Dr. R. Käppeli

Problem 4.1 Operator overloading and function templates

This exercise focuses on the implementation of a new type describing the $\mathbb{Z}_2$ group, and on the implementation of a generic power function working with all standard numeric types, as well as our new $\mathbb{Z}_2$.

Definition The finite group $\mathbb{Z}_2$ has the following properties:

- The group's element are: $\mathbb{Z}_2 = \{+, -\}$, where $+$ is the identity element.
- The group operation $\cdot$ is defined through:

$$+ \cdot + = - \cdot - = +$$
$$- \cdot + = + \cdot - = -$$

- The representation of a group element $g(\mu)$, $\mu \in \mathbb{Z}_2$ on integer, real or complex numbers is given by

$$g(\mu) = \begin{cases} +1, & \text{for } \mu = + \\ -1, & \text{for } \mu = - \end{cases}$$

a) Implementation of $\mathbb{Z}_2$ To represent the $\mathbb{Z}_2$ group in C++, we will use the enumeration type:

```
enum Z2 { Plus, Minus };
```

The group operation will be implemented by overloading the * operator, i.e. we assign the correct meaning to the expression

```
Z2 p = Plus, m = Minus;
Z2 r = p*m;
```

Every time we make use of the operator *, the C++ compiler is looking for the function `operator*` with the correct types to be invoked[1]. In our case, we need to define:

```
Z2 operator*(Z2 a, Z2 b);
```

Furthermore, we want to be able to print our result in a nice form, i.e. using an expression such as `std::cout << r << std::endl;`. We will therefore overload

```
ostream& operator<<(ostream& os, Z2 a);
```

in such a way that *Plus* is printed for $+$ and *Minus* for $-$.
To implement the action of a group element on a number, we will implement the following function templates (note that from the point of view of C++, `a*b` is not necessarily the same as `b*a`):

```
template<class T> T operator*(T a, Z2 b);
template<class T> T operator*(Z2 a, T b);
```

[1] The same is also valid for all operators: $+, -, (), [], <<$, etc.

b) Implementation of generic power function We also want to implement a power function template which only relies on the multiplication, so that it can also be used on our $\mathbb{Z}_2$ group:

```
template<class T> T mypow(T a, unsigned int n);
```

Hint: In order to be generic, we provide a function template `identity_element`, which returns *one* for all numeric types. How can you overload such a function to provide the identity element of the $\mathbb{Z}_2$ group?

c) Concepts / named requirements Document the concepts/named requirements (including the semantics/meaning of the operators you use) of your function templates.

Problem 4.2 Simpson integration with function objects

In a previous exercise we wrote a static library to perform the Simpson integration of $\sin(x)$ in the region $x \in [0, \pi]$, where the integrand was passed as a function pointer. In this exercise, implement the Simpson integration of $\exp(-\lambda x)$ in the region $x \in [0, 1]$ by the use of a function object.

1. Rewrite the `simpson` function so that it works with function objects.
   Hint: Replace the function pointer by a template parameter.

2. Introduce a template parameter for the type of the integration boundaries `a` and `b`.
   What is its concept?
   What happens if you call your function like `simpson(0, 1, 128, func_obj)`?

3. Document the concepts/named requirements of your function.

Keep in mind (and make sure you know why) that you need to provide the definition and not just the declaration in the header when templating the Simpson integration (i.e. you do not need the simpson.cpp file anymore and you will not build a library as in exercise 2.2).

Problem 4.3 Class Header for Animal and Genome

Read the paper by Penna [T.J.P. Penna, J. Stat. Phys. 78, 1629 (1995)][2]. 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?
Having understood the paper and thought about its structure, think of an animal and a genome as objects and write the class headers[3] for them.

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

[3]Only the definition of the class and declaration of members.