Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
19.05.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1. $f'(x)$ can also be approximated by

- the central differential quotient

$$\Phi(h) := \frac{f(x+h) - f(x-h)}{2h} \quad \text{für } h > 0$$

- and by the one-sided differential quotient

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{für } h > 0$$

Write a function `diff` which uses the central differential quotient and a function `diff2` which uses the one-sided differential quotient. The function f should be an input parameter of `diff` und `diff2`! Which other input parameters do these two functions need? Test your function with $f(x) = \exp(x)$ at $x = 1$. Compare the runtime of both functions. Save your source code as `diffcomp.c` into the directory `serie09`.

Aufgabe 9.2. The command `cin` reads a text input only until the first space. Write a function `myFullName`, which reads your given and your surname from the keyboard and stores them both to strings. Then, put these two strings together in one string and print that string on screen. Save your source code as `MyFullName.cpp` into the directory `serie09`. Test your code on a suitable example. Do you know any other ways to read a longer keyboard input?

Aufgabe 9.3. Write a class `Name` which contains two members, `firstName` and `surname` of type `string`. Implement the set-method `setName` that has one string variable as input parameter, and splits the input in first name and surname automatically. Note that the input can contain multiple first names. Furthermore, write a method `printName` which prints out the whole name on the monitor. In case of multiple first names, the output should be shortened as follows: The name `Max Maxi Mustermann` should be printed out as `Max M. Mustermann`. Save your source code as `name.cpp` into the directory `serie09`.

Aufgabe 9.4. Extend the class `Fraction` from the lecture by the `public` method `void reduce()` that determines the reduced form of the fraction `numerator/denominator`. Use the *euclidean division algorithm*. Moreover, implement the method `setValue(string value)` that converts an arbitrary number, given as a string, into a fraction. For the implementation you can proceed as follows: First, find the decimal-point in the string and count the number of positions after the decimal-point. Then, erase the decimal-point from the string. The string now represents a natural number and can be converted into an `int` variable by use of the function `atoi`. This number is used for the numerator. Then, the denominator is set to 10^p, where $p \in \mathbb{N}$ is the number of positions after the decimal-point. Then, call the method `reduce()`. Finally, overload the method `setValue` in an appropiate way, so that `setValue(n)` for n of type `int` makes sense. Save your source code as `fraction.cpp` into the directory `serie09`. Test your code on a suitable example.
Hint: The method `find` of the class `string` allows you to find a specific character in the string, e.g., `int pos = value.find('.')` returns the position of the decimal-point in the string `value`. The call `value.erase(pos,k)`, erases `k` characters after the position `pos` in the string `value`. The function `atoi` from the standard library `cstdlib` converts a given string (in C-style) to an `int` variable. To get the string as `char *`, you can use the method `c_str()` of class `string`.

Aufgabe 9.5. Write a class `University`. This class should contain the members `numStudents`, `city`, and `name` as well as the methods `graduate`, and `newStudent`. If the method `graduate` is called, the number of students gets decreased by one, whereas if `newStudent` is called, the number of students increases by one. All data-members should be declared as `private`! Therefore, you have to implement `get` and `set` methods. Save your source code as `University.cpp` into the directory `serie09`.

Aufgabe 9.6. Write a class `Deposit` with members `accountNumber`, `assets`, and `ratePerCent`. Moreover, implement `set` and `get` methods for the members `accountNumber`, `assets`. To change the assets, write a method `drawMoney` and `placeOnDeposit`. Note that with this deposit you are not allowed to draw more money than is given, i.e., the member `assets` must be positive. The rate per cent as well as the account number must also be positive. Finally, implement the method `calculateAssets`. Save your source code as `Deposit.cpp` into the directory `serie09`.

Aufgabe 9.7. Write a class `Stopwatch` that simulates a stopwatch. The stopwatch consists of two buttons: If the first button is pressed, then the time measurement starts. If the button is pressed again, then the time measurement stops. The second button is used to reset the time to zero. To realize this situation, implement the methods `pushButtonStartStop` (first button) and `pushButtonReset`. Implement another method that prints out the time formatted in the style `hh:mm:ss.xx`, e.g., if the measured time is two minutes, then the output should be `00:02:00.00`. Save your source code as `Stopwatch.cpp` into the directory `serie09`.
Hint: Use the data-type `clock_t` and the function `clock()` from the library `time.h`. It makes sense to use a variable `isRunning` of type `bool`. If the first button is pressed, then this variable is either set to `true` or `false`.

Aufgabe 9.8. According to the lecture Members of the class can only be accessed indirectly via `set`- and `get`-methods. What is the output of the following C++ program? Why is this possible? Explain why this is a bad programming style.

```
#include <iostream>
using std::cout;
using std::endl;

class Test{

private:
  int N;

public:
  void setN(int N_in) { N = N_in; };
  int getN(){ return N; };
  int* getptrN(){ return &N; };

};


int main(){

  Test A;
  A.setN(5);
  int* ptr = A.getptrN();
  cout << A.getN() << endl;
  *ptr = 10;
  cout << ptr << endl;
  cout << A.getN() << endl;

  return 0;
}
```