Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
29.05.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1. Write a class `customer` for a bank customer. The class contains the name of the customer as `string`, the current balance as `double` and a pin code as `int`. Implement `set` and `get` methods for the member variables as well as the following class methods

- `void printBalance()`
  prints the current balance on the screen.
- `bool checkPIN()`
  reads in a PIN code and checks whether it is correct or not.
- `void drawMoney()`
  checks a given PIN code, reads in the amount the customer wants to draw, and prints the new balance on the screen. The account must not be overdrawn. If necessary, print a warning on the screen.

How did you test your implementation? Save your source code as `customer.{hpp,cpp}` into the directory `serie09`.

Aufgabe 9.2. 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. How did you test your implementation? Save your source code as `university.{hpp,cpp}` into the directory `serie09`.

Aufgabe 9.3. For the HR-department of the University it can be tedious to add and delete students one by one in their data. Therefore, overload the methods `graduate` and `newStudent` from the class `University` from Exercise 9.2, so that the number of graduating and beginning students can be a parameter of the methods. Moreover, write constructors which initialize your object with meaningful data. If the object is not initialized directly, then set `numStudents = 0`, `city = noWhere`, `name = noName`. Write a `plot`-routine to print the data of your object on screen. How did you test your implementation? Save your source code as `University.{hpp,cpp}` into the directory `serie09`.

Aufgabe 9.4. Extend the class `Triangle` from the lecture (slides 213ff) by two further methods:

- the method `getPerimeter()`, which computes and returns the perimeter of a triangle;
- the method `isEquilateral()`, which checks whether a triangle is equilateral.

Test your implementation in a suitable way!

Aufgabe 9.5. Write a class `Stopwatch` to measure a routines execution time. The stopwatch has the following two methods. If the first method is called, then the time measurement starts. If the method is called again, the time measurement stops. The second method is used to reset the time to zero. To realize this situation, implement the methods `pushButtonStartStop` and `pushButtonReset`. Implement another method `print` 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`. Use the following code snippet to test your implementation

```
Stopwatch S;
double sum = 0.0;
S.pushButtonStartStop();
for(int j=0; j<100*1000*1000; ++j)
  sum += 1./j;
S.pushButtonStartStop();
S.print();
```

What is computed here? Save your source code as `Stopwatch.{hpp,cpp}` into the directory `serie09`.

Hint: Use the data-type `clock_t` and the function `clock()` from the library `time.h`. The elapsed time in seconds between two calls of `clock()` can be obtained via

```
clock_t t1, t2;
double secs;
t1 = clock();
/* ... do some work ... */
t2 = clock();
secs = (double) (t2-t1) / CLOCKS_PER_SEC;
```

It makes sense to use a variable `isRunning` of type `bool`. If the method `pushButtonStartStop` is called, then this variable is either set to `true` or `false`.

Bonus: Adapt the code snippet from above in order to compute $\sum_{j=1}^{N} j^3$ for $N = 10^8$ using two different methods: Once naively using the power function `pow(j,3)` from the math library, once clever without using the math library. Measure the execution time of the different strategies. What do you observe?

Aufgabe 9.6. Write a structure `Matrix` to save quadratic $n \times n$ `double` matrices. Distinguish between fully-populated matrices (type `'F'`), lower triangle matrices (type `'L'`) and upper triangle matrices (type `'U'`). A lower triangular matrix L and an upper triangular matrix U have the following polulation structure:

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & u_{33} & \dots & u_{3n} \\ & & & \ddots & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix} \qquad L = \begin{pmatrix} \ell_{11} & & & & \mathbf{0} \\ \ell_{21} & \ell_{22} & & & \\ \ell_{31} & \ell_{32} & \ell_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{pmatrix}$$

We thus have $u_{jk} = 0$, if $j > k$ and $\ell_{jk} = 0$, if $j < k$. A fully populated matrix should by stored in Fortran-style, i.e., columnwise in a dynamical vector with $n \cdot n$ entries. Triangular matrices should be stored in a vector with $\sum_{j=1}^{n} j = n(n+1)/2$ entries. Implement the following functionalities:

- Default constructor, which allocates a 0×0 matrix of the type `'F'`
- Constructor, which gets the type and the dimension as an input parameter
- Destructor
- `get` and `set`-methods for the matrix entries, the type and the dimension

The `get` and `set`-methods for the matrix entries depend on the type of the matrix. How did you test your implementation? Save your source code as `matrix.{hpp,cpp}` into the directory `serie09`.

Aufgabe 9.7. Extend the class `Matrix` from Exercise 9.6 by

- a method `scanMatrix(char typ, int n)` to read the type and the matrix $A \in \mathbb{R}^{n \times n}$ depending on the type from the keyboard,
- a method `printMatrix()`, which prints the matrix to the screen,

- a method `columnsumnorm()`, which computes and returns the column sum norm

$$\|A\| = \max_{k=0,\ldots,n-1} \sum_{j=0}^{n-1} |a_{jk}|,$$

- a method `rowsumnorm()`, which computes and returns the row sum norm

$$\|A\| = \max_{j=0,\ldots,n-1} \sum_{k=0}^{n-1} |a_{jk}|.$$

Note that for lower resp. upper triangular matrizes the methods can access only coefficients a_{jk} resp. a_{kj} with $0 \leq k \leq j \leq n-1$. How did you test your implementation? Save your source code as `matrix2.{hpp,cpp}` into the directory `serie09`.

Aufgabe 9.8. According to the lecture, the members of a 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;
}
```