Department of Mathematics
Dr. Ulf Lorenz
Christian Brandenburg

TECHNISCHE
UNIVERSITÄT
DARMSTADT

winter term 2008/2009
26/01/2009

Introduction to Mathematical Software

6th Exercise Sheet

Exercise 1 (Visualization of Statistical Samples)

(a) Obtain the files `window.h`, `window.cc`, and `stat.cc` from the course webpage (section *Source Code*).

(b) The files `window.h` and `window.cc` contain the definition of the class `window`, an intentionally very simple C++ class for drawing graphical windows under the X11-Window system (this is the graphical user interface you are using on Linux computers).
Fortunately, in order to use the `window` class, you do not need to know about its inner workings. The only thing that is interesting for you is its interface, i.e. the functions it provides (this is a programming paradigm called *design by contract*).
Here is a short description of the functions that `window` provides:

`int SetPoint(int x, int y);` draw a black point in the window at position (x, y). Note that the window's coordinate system has the point $(0, 0)$ in the upper left corner with the x-axis running from left to right and the y-axis running from top to bottom.

`int SetBar(int x, int y);` draw a black line from the point $(x, 0)$ to the point (x, y), i.e. a vertical bar.

`int ErasePoint(int x, int y);` delete the point at position (x, y)

`int EraseBar(int x, int y);` delete the bar from $(x, 0)$ to (x, y)

`int ClearScreen( void );` delete all elements in the window

`int win_setup(char *disp, char* program, int width, int height, int xpos, int ypos)` creates a window with width `width` and height `height` and places it on the screen such that its upper left corner is at position (`xpos`, `ypos`).

These functions are declared in the class definition of `window` in the file `window.h`. The `public` access specifier before these functions notes that these are the class's member functions that can be called from outside the class.

(c) The file `stat.cc` contains, apart from the `main` function of your program, the declaration of the class `Sample`, which visualizes samples of a discrete random variable. The size of the sample is `MAX_ELEMENTS`, the values of the random variable range from 0 to `MAX_NUMBER`.

The class `Sample` is derived from the class `window`, which is denoted by `: public window` after the class name. This means that it inherits the members of `window` and a user of `Sample` can access all public members of `window`, and in particular can call all public member functions, as if they were members of `Sample`. Apart from that, `Sample` defines its own function and data members:

`void plot_sample ();` plot the sample in the window

`void plot_arithmetic_mean ();` compute the arithmetic mean and plot it

`void plot_sample_standard_deviation ();` compute the sample standard deviation and plot it

`void init_sample(int size);` create a sample of size `size`

`int values [MAX_ELEMENTS];` array containing the values of the sample

`int NumPoints;` size of the array `values`

`double mean;` arithmetic mean of the sample

`double s;` standard deviation of the sample

The four functions are `public`, and hence make up the interface of the `Sample` class. The four variables are `private` and therefore are invisible from outside the class, making up the implementation details of the class, which are irrelevant to users.

(d) Before changing the source file, compile the program first to see if everything is ok by typing

```
g++ -g window.cc stat.cc -lX11 -o stat
```

on the command line. What this command does is the following:

`g++` calls the GNU Compiler Collection's C++ compiler

`-g` tells the compiler to add debugging information

`window.cc stat.cc` tells the compiler to translate these source files to object code and link them

`-lX11` tells `g++` to link the program file against the X11-Window libraries to make available functions for creating windows and drawing objects

`-o stat` tells `g++` to name the resulting executable `stat`

Run the program by typing `./stat`. A blank window should pop up. Make sure that the window and the console do not overlap, because the `window` class has no redraw capability. Pressing `enter` once will show a random sample in the window. Pressing `enter` three more times ends the program. If the program behaves this way, you can proceed to the next step.

(e) The *arithmetic mean* $\bar{X}$ of a sample with elements X_i is given by the formula

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Implement the member function `void Sample::plot_arithmetic_mean ()` that computes the mean of the sample and plots it into the window.

Hints:

- the value of $\bar{X}$ is stored in the member variable `double mean`.
- use the `SetPoint` function to draw a line corresponding to the mean.
- to draw the mean, you need *type casting*. The reason is that `SetPoint` requires two arguments of type `int`. However, the type of `mean` is `double`, so passing it to `SetPoint` results in an error. The solution to this problem is to cast `mean` to type `int` with the syntax `(int) mean` and pass this variable to `SetPoint`.

(f) The *sample standard deviation* is defined as

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$$

Implement the member function `void Sample::plot_sample_standard_deviation ()`. Compute the sample standard deviation similarly to the computation of the arithmetic mean. Draw the sample standard deviation from the arithmetic mean, i.e. plot the lines $\bar{X} \mp s$.

Hints:

- for computing the square root, use the function `sqrt` in the `math.h` header file.

(g) test your program with different values of `MAX_ELEMENTS`, e.g. 30, 50, 100, 150, 250, 500. What happens?

Exercise 2 (Classes and Pointers)

(a) Obtain the file `pointer.cc` from the Course Webpage.

(b) Open the file in a text editor and try to understand what happens.

(c) Compile and run the program. Is the result what you expected?

(d) To learn more about pointers, see `http://www.cplusplus.com/doc/tutorial/pointers.html`.

Exercise 3 (Three Term Linear Recurrence)

The three term linear recurrence relation

$$x_{n+1} = Ax_n + Bx_{n-1}$$

has the general solution $x_n = C_1(\rho_1)^n + C_2(\rho_2)^n$, where ρ_1 and ρ_2 are the roots of the quadratic equation

$$\rho^2 - A\rho - B = 0$$

with constants C_1 and C_2 that can be determined from the initial data x_0 and x_1.

- Show (using pen and paper) that $x_n = C_1 + C_2(-B)^n$ if $-1 \neq B = 1 - A$.
- Show that $C_1 = c, C_2 = 0$ in the case that $x_0 = x_1 = c$.
- Write a program that, for integers `i1`, `i2`, `i3`, sets type `double` variables `A = i1+1`, `B=-i1`, `x[0] = x[1] = (double)i2 / (double)i3` and calculates the values `x[2]` ... `x[25]` using the three term linear recurrence relation. Test your program with the following values:

case	`i1`	`i2`	`i3`
1	10000001	1	2
2	10000001	1	3
3	10000001	1	4
4	100000001	1	2
5	100000001	1	3
6	100000001	1	4

  What happens?
- Now, change the type of `A, B, x[]` to `float` and test your program with the same cases. What happens? Can you explain the result?