Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Theoretische Informatik
Dr. B. Gärtner, Prof. Dr. J. Hromkovič

7. Dezember 2010

Informatik für Mathematiker und Physiker Serie 11 HS 10

URL: `http://www.ti.inf.ethz.ch/ew/courses/Info1_10/`

Skript-Aufgabe 123 (4 Punkte)

Write programs that produce turtle graphics drawings for the following Lindenmayer systems (Σ, P, s).

a) $\Sigma = \{F, +, -\}$, $s = F + F + F + F$ and P given by

$$F \mapsto FF + F + F + F + F + F - F.$$

b) $\Sigma = \{X, Y, +, -\}$, $s = Y$, and P given by

$$\begin{aligned} X &\mapsto Y + X + Y \\ Y &\mapsto X - Y - X. \end{aligned}$$

For the drawing, use rotation angle $\alpha = 60$ degrees and interpret *both* X and Y as "move one step forward".

c) Like b), but with the productions

$$\begin{aligned} X &\mapsto X + Y + +Y - X - -XX - Y + \\ Y &\mapsto -X + YY + +Y + X - -X - Y. \end{aligned}$$

Skript-Aufgabe 129 (4 Punkte)

Define a type `Tribool` for three-valued logic; in three-valued logic, we have the truth values *true*, *false*, and *unknown*.

For the type `Tribool`, implement the logical operators

```
// POST: returns x AND y
Tribool operator&& (Tribool x, Tribool y);

// POST: returns x OR y
Tribool operator|| (Tribool x, Tribool y);
```

where AND ($\wedge$) and OR ($\vee$) are defined according to the following two tables.

$\wedge$	*false*	*unknown*	*true*
false	*false*	*false*	*false*
unknown	*false*	*unknown*	*unknown*
true	*false*	*unknown*	*true*

$\vee$	*false*	*unknown*	*true*
false	*false*	*unknown*	*true*
unknown	*unknown*	*unknown*	*true*
true	*true*	*true*	*true*

Test your type by writing a program that outputs these truth tables in some format of your choice.

Skript-Aufgabe 142 (8 Punkte)

The C++ standard library also contains a type for computing with *complex numbers*. A complex number where both the real and the imaginary part are doubles has type `std::complex<double>` (you need to `#include <complex>` in order to get this type). In order to get a a complex number with real part r and imaginary part i, you can use the expression

```
std::complex<double>(r,i); // r and i are of type double
```

Otherwise, complex numbers work as expected. All the standard operators (arithmetic, relational) and mathematical functions (`std::sqrt`, `std::abs`, `std::pow`...) are available. The operators also work in mixed expressions where one operand is of type `std::complex<double>` and the other one of type `double`. Of course, you can also input and output complex numbers.

Here is the actual exercise. Implement the following function for solving quadratic equations over the complex numbers:

```
// POST: return value is the number of distinct complex solutions
//       of the quadratic equation ax^2 + bx + c = 0. If there
//       are infinitely many solutions (a=b=c=0), the return
//       value is -1. Otherwise, the return value is a number n
//       from {0,1,2}, and the solutions are written to s1,..,sn
int solve_quadratic_equation (std::complex<double> a,
                              std::complex<double> b,
                              std::complex<double> c,
                              std::complex<double>& s1,
                              std::complex<double>& s2);
```

Test your function in a program for at least the triples (a, b, c) from the set

$$\{(0,0,0),(0,0,2),(0,2,2),(2,2,2),(1,2,1),(i,1,1)\}.$$

Die **Aufgaben 127** aus den Vorlesungsunterlagen ist die **Challenge Aufgabe** und gibt 8 Punkte, wenn sie vollständig gelöst wird.

Abgabe: Bis 14. Dezember 2010, 15.15 Uhr.