Computer Systems (SS 2011) Exercise 2: April 18, 2011

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Wolfgang.Schreiner@risc.jku.at

March 25, 2011

The exercise is to be submitted by the denoted deadline via the submission interface of the Moodle course as a single file in zip (`.zip`) or tarred gzip (`.tgz`) format which contains the following files:

- A PDF file `Exercise`*`Number-MatNr`*`.pdf` (where *Number* is the number of the exercise and *MatNr* is your "Matrikelnummer") which consists of the following parts:
  1. A decent cover page with the title of the course, the number of the exercise, and the author of the solution (identified by name, Matrikelnummer and email address).
  2. For every source file, a listing in a *fixed width font*, e.g. `Courier`, (such that indentations are appropriately preserved) and an appropriate *font size* such that source code lines do not break.
  3. A description of all tests performed (copies of program inputs and program outputs) explicitly highlighting, if some test produces an unexpected result.
  4. Any additional explanation you would like to give. In particular, if your solution has unwanted problems or bugs, please document these explicitly (you will get more credit for such solutions).
- Each source file of your solution (no object files or executables).

Please obey the coding style recommendations posted on the course site.

Exercise 2: Polygons Extended

Write a class `Polygon` with the following (minimum) public interface:

```
class Polygon
{
public:
  // as in Exercise 1
  Polygon();
  ~Polygon();
  void add(double x, double y);
  void random(int n, int x, int y, int w, int h, int seed = 0);
  bool read(const char* filename);

  // draws the polygon (ignoring any intersections)
  void draw(unsigned int color1 = 0, unsigned int color2 = 0);

  // set polygon to initial state
  void clear();

  // copy constructor and copy assignment operator
  Polygon(const Polygon poly&);
  Polygon& operator=(const Polygon& poly);

  // move polygon by vector (x, y)
  void move(double x, double y);

  // rotate polygon around point (x, y) by angle a
  void rotate(double x, double y, double a);

  // expand/shrink polygon around point (x, y) by factor f
  void expand(double x, double y, double f);

  // get pointer to bounding rectangle of polygon
  Polygon* boundingRectangle();

  // get pointer to convex hull of polygon
  Polygon* convexHull();
};
```

The basis of this exercise is the class `Poly` (and the associated classes) of Exercise 1[1] but only include that code that is relevant for this exercise. The new functions to be implemented are as follows:

1. The function `clear()` sets the polygon to the initial state (deallocating all allocated memory appropriately).

[1]If you have not solved that exercise, you may ask a colleague for a solution.

2. The copy constructor and the copy assignment operator overwrite this polygon by the data of a polygon *poly* such that this polygon becomes a copy of *poly*. Please note that

   a) the memory allocated by this polygon has to be deallocated before the polygon is overwritten (no memory leaks may arise),

   b) after the call, the two polygons have no shared representation (all data have to be copied).

3. The function `move(`x`,`y`)` shifts every point of the polygon by the vector $\langle x, y\rangle$.

4. The function `rotate(`x`,`y`,`a`)` rotates every point of the polygon around point $\langle x, y\rangle$ by angle a in radians.

5. The function `expand(`x`,`y`,`f`)` multiplies the distance of point $\langle x, y\rangle$ to every point of the polygon by factor f ($f > 1$ expands the polygon, $f < 1$ shrinks it).

6. The function `boundingRectangle()` returns the smallest rectangle that encloses the polygon.

7. The function `convexHull()` returns the convex hull of the points of the polygon. For this purpose, you may apply the "gift wrapping algorithm" explained on the corresponding Wikipedia page[2].

Write a program for testing the functions of this class using (among others) the polygons of Exercise 1. Deliver the source code, the textual input/output of the test program (if any) and screenshots of the graphical output.

[2] `http://en.wikipedia.org/wiki/Gift_wrapping_algorithm`