Buğra Gedik's Courses
@ Bilkent University

cs315:fall2016:project_part1

Design a Figure Drawing Language

You are asked to design a figure drawing language. The language should provide imperative features to enable creation of drawings. It should support modularity at the level of functions. It should have extensibility features to create new shapes, which would ease the creation of complex drawings.

Here are some requirements:

1. (4pts) The language should support configuration of the following
    - (2pts) an entry point (the function that does the drawing)
    - (2pts) the width, height, and the default stroke width of the figure
2. (6pts) In addition to drawing with explicit sizes, drawing independent of scale should be supported. As an example of the latter, one should be able to draw 10 ovals next to each other horizontally, without specifying absolute values for the oval width and height.
3. (5pts) The language should be dynamically typed.
    - (2pts) It should support the following primitive types: float, integer, string, and boolean.
    - (2pts) Basic arithmetic and logical operations should be supported on these types.
    - (1pts) Strings should support concatenation via the + operator and conversion from other types.
4. (3pts) The language should also support Location, Size, and Color types.
    - (1pts) Location should provide access to two members: x and y.
    - (1pts) Size should provide access to two members: width and height.
    - (1pts) Color should provide a predefined set of constants, as well as an RGB constructor.
5. (18pts) The language should provide a number of built-in shapes.
    - (3pts) These shapes should include at least the following: Line, Rectangle, and Oval.
    - (15pts) There should be a common syntax for instantiating shapes.
        - (5pts) This syntax should support parameters, including optional ones.
        - (5pts) A shape instance should be scale free, that is it should not reference a location or a size.
        - (5pts) You can assume that any given shape can be drawn inside a bounding box. Thus, that bounding box defines its location and size, which is specified later when the shape is drawn.
            - For instance, a Line should support specifying a direction (NE (north east), NW, SE, SW), arrows (start and end), arrow sizes (relative to line stroke width), etc. However, an Oval should not specify any parameters. A Rectangle can specify a parameter about rounded corners.
6. (14pts)The language should support basic drawing statements.
    - (10pts) These statements should support parameterization, which can be used to specify
        - (2pts) location, (2pts) size, (2pts) stroke width (relative), (2pts) fill state, and (2pts) fill color.
        - You can assume that some parameters are mandatory, some optional.
    - (4pts) Similar to drawing shapes, drawing strings should be supported as well.
7. (10pts) The language should support loops that can ease repetitive tasks.
8. (10pts) The language should support conditionals (if and if/else) as well.
9. (10pts) The language should support modularity at the level of functions.
10. (20pts) The language should support extension of shapes.
    - (10pts) This extension should be performed via defining composite shapes
    - (5pts) These shapes should be parameterizable (including optional and mandatory parameters).

- (5pts) Once a shape is defined, it should be possible to draw it as usual using the shape drawing functions.

We ask you to do the following:

1. (50%) Design a language (give it a name) to meet the requirements described above.
   - This would be graded based on the points indicated as part of the requirements, using your tutorial, report, and samples.
2. (15%) Write a tutorial for the language you have designed.
3. (15%) Write a report describing how you addressed each requirement. For each requirement that has points associated with it, briefly describe language features you have used to cover the requirement. You can cross-reference the tutorial for this purpose.
4. (5%) Write 3 sample programs, each drawing a non-trivial figure.
5. (15%) Write a lexer for your language, using Lex.

Logics

Once you are done, put your deliverable under a directory named group<GroupNo>_proj1 and make an archive from that directory. It is important that your code should compile without any issues using the `make` command. Once complete, it should output an executable called 'lexer' that can be used to lex your sample input. For example, the following Unix commands could be used:

```
mkdir group<GroupNo>_proj1
cd group<GroupNo>_proj1
    mkdir code      # contains Makefile, lex, and C code
    mkdir samples   # contains the program samples
    mkdir documents # contains the tutorial and report
    (edit and test your files)
    ...
cd ..
tar -cvzf group<GroupNo>_proj1.tar.gz group<GroupNo>_proj1
```

Then upload this newly generated file (named group<GroupNo>_proj1.tar.gz) into the course Moodle.

Reports in formats other than .pdf and .txt are not accepted.

cs315/fall2016/project_part1.txt · Last modified: 2016/10/29 15:04 by bgedik