```
 1:
 2: #ifndef __INTERP_H__
 3: #define __INTERP_H__
 4:
 5: #include <iostream>
 6: #include <map>
 7:
 8: using namespace std;
 9:
10: #include "object.h"
11: #include "trace.h"
12:
13: typedef map <string, object *> objectmap;
14:
15: class interpreter {
16:    public:
17:       typedef list<string> parameters;
18:       interpreter (const string &, ostream &, objectmap &);
19:       ~interpreter ();
20:       void interpret (parameters &);
21:    private:
22:       interpreter (); // Disable
23:       interpreter (const interpreter &); // Disable
24:       interpreter &operator= (const interpreter &); // Disable
25:
26:       // Data fields.
27:       typedef void (interpreter::*interpreterfn) (parameters &);
28:       typedef object *(interpreter::*factoryfn) (parameters &);
29:       static map <string, interpreterfn> interpmap;
30:       static map <string, factoryfn> factorymap;
31:       ostream &outfile;
32:       int pagenr;
33:       objectmap objmap;
34:       string infilename;
35:       double page_xoffset;
36:       double page_yoffset;
37:
38:       // Service functions.
39:       void do_define (parameters &);
40:       void do_draw (parameters &);
41:       void do_newpage (parameters &);
42:       void prolog ();
43:       void startpage ();
44:       void endpage ();
45:       void epilog ();
46:
47:       // Factory functions.
48:       object *make_object (parameters &);
49:       object *make_text (parameters &);
50:       object *make_ellipse (parameters &);
51:       object *make_circle (parameters &);
52:       object *make_polygon (parameters &);
53:       object *make_rectangle (parameters &);
54:       object *make_square (parameters &);
55:       object *make_line (parameters &);
56: };
57:
58: RCSH(__interp_h__,
59: "$Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $")
60:
61: #endif
62:
```

```
 1:
 2: #ifndef __NUMBERS_H__
 3: #define __NUMBERS_H__
 4:
 5: #include <iostream>
 6: #include <utility>
 7:
 8: using namespace std;
 9:
10: #include "trace.h"
11:
12: const double PTS_PER_INCH = 72;
13:
14: class degrees {
15:    friend ostream &operator<< (ostream &, const degrees &);
16:    public:
17:       explicit degrees (double init): angle(init) {}
18:       operator double() {return angle; }
19:    private:
20:       degrees (); // Disable.
21:       double angle;
22: };
23:
24: class points {
25:    friend ostream &operator<< (ostream &, const points &);
26:    public:
27:       explicit points (double init): pointvalue(init) {}
28:       operator double() {return pointvalue; }
29:    private:
30:       points (); // Disable.
31:       double pointvalue;
32: };
33:
34: class inches {
35:    friend ostream &operator<< (ostream &, const inches &);
36:    public:
37:       explicit inches (double init): pointvalue(init * PTS_PER_INCH) {}
38:       operator double() {return pointvalue; }
39:    private:
40:       inches (); // Disable.
41:       double pointvalue;
42: };
43:
44: typedef pair <inches, inches> xycoords;
45:
46: ostream &operator<< (ostream &, const xycoords &);
47:
48: RCSH(__numbers_h__,
49: "$Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $")
50:
51: #endif
52:
```

```
 1:
 2: #ifndef __OBJECT_H__
 3: #define __OBJECT_H__
 4:
 5: #include <iomanip>
 6: #include <iostream>
 7: #include <list>
 8: #include <utility>
 9:
10: using namespace std;
11:
12: #include "numbers.h"
13:
14: //
15: // Objects constitute a single-inheritance hierarchy, summarized
16: // here, with the superclass listed first, and subclasses indented
17: // under their immediate superclass.
18: //
19: // object
20: //    test
21: //    shape
22: //       ellipse
23: //          circle
24: //       polygon
25: //          rectangle
26: //             square
27: //          line
28: //
29:
30: typedef list<xycoords> coordlist;
31:
32: //
33: // Abstract base class for all shapes in this system.
34: //
35:
36: class object {
37:    public:
38:       virtual ~object ();
39:       virtual void draw (ostream &, const xycoords &,
40:                          const degrees &angle) = 0;
41:    protected:
42:       object () {}
43: };
44:
45: //
46: // Class for printing text.
47: //
48:
49: class text: public object {
50:    public:
51:       text (const string &fontname, const points &fontsize,
52:             const string &textdata);
53:       virtual void draw (ostream &, const xycoords &,
54:                          const degrees &angle);
55:    protected:
56:       string fontname;
57:       points fontsize;
58:       string textdata;
59: };
60:
61: //
62: // Shape of a geometric object.
63: //
64:
```

```
 65: class shape: public object {
 66:    protected:
 67:       shape (const points &thick): thick(thick) {}
 68:       points thick;
 69:    private:
 70:       shape (); // Disable.
 71: };
 72:
 73: //
 74: // Classes for ellipse and circle.
 75: //
 76:
 77: class ellipse: public shape {
 78:    public:
 79:       ellipse (const inches &height, const inches &width,
 80:                const points &thick);
 81:       virtual void draw (ostream &, const xycoords &,
 82:                          const degrees &angle);
 83:    protected:
 84:       inches height;
 85:       inches width;
 86: };
 87:
 88: class circle: public ellipse {
 89:    public:
 90:       circle (const inches &diameter, const points &thick);
 91: };
 92:
 93: //
 94: // Class polygon.
 95: //
 96:
 97: class polygon: public shape {
 98:    public:
 99:       polygon (const coordlist &coords, const points &thick);
100:       virtual void draw (ostream &, const xycoords &,
101:                          const degrees &angle);
102:    protected:
103:       const coordlist coordinates;
104: };
105:
106: //
107: // Classes rectangle, square, and line..
108: //
109:
110: class rectangle: public polygon {
111:    public:
112:       rectangle (const inches &height, const inches &width,
113:                  const points &thick);
114:    private:
115:       static coordlist make_list (
116:              const inches &height, const inches &width);
117: };
118:
119: class square: public rectangle {
120:    public:
121:       square (const inches &width, const points &thick);
122: };
123:
124: class line: public polygon {
125:    public:
126:       line (const inches &length, const points &thick);
127:    private:
128:       static coordlist make_list (const inches &length);
```

```
129: };
130:
131: RCSH(__object_h__,
132: "$Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $")
133:
134: #endif
135:
```

```
 1:
 2: #ifndef __TRACE_H__
 3: #define __TRACE_H__
 4:
 5: #include <vector>
 6:
 7: using namespace std;
 8:
 9: //
10: // traceflags -
11: //    static class for maintaining global trace flags, each indicated
12: //    by a single character.
13: // setflags -
14: //    Takes a string argument, and sets a flag for each char in the
15: //    string.  As a special case, '@', sets all flags.
16: // getflag -
17: //    Used by the TRACE macro to check to see if a flag has been set.
18: //    Not to be called by user code.
19: //
20:
21: class traceflags {
22:    private:
23:       static vector<char> flags;
24:    public:
25:       static void setflags (const string &optflags);
26:       static bool getflag (char flag);
27: };
28:
29: //
30: // TRACE -
31: //    Macro which expands into trace code.  First argument is a
32: //    trace flag char, second argument is output code that can
33: //    be sandwiched between <<.  Beware of operator precedence.
34: //    Example:
35: //       TRACE ('u', "foo = " << foo);
36: //    will print two words and a newline if flag 'u' is  on.
37: //    Traces are preceded by filename, line number, and function.
38: //
39:
40: #define TRACE(FLAG,CODE) { \
41:            if (traceflags::getflag (FLAG)) { \
42:               cerr << __FILE__ << ":" << __LINE__ << ":" \
43:                    << __func__ << ": "; \
44:               cerr << CODE << endl; \
45:            } \
46:         }
47:
48: //
49: // RCSH, RCSC -
50: //    Macros which allow RCS Id information to transfer to object
51: //    files and executable binaries.
52: //
53:
54: #define RCSH(NAME,ID) \
55: static const char __RCS_##NAME[] = "\0" ID;
56: #define RCSC(NAME,ID) \
57: static const char __RCS_C_##NAME[] = "\0" ID \
58: "\0$Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $";
59:
60: RCSH(__trace_h__,
61: "$Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $")
62:
63: #endif
64:
```

```
 1: //
 2: // util -
 3: //    A utility class to provide various services not conveniently
 4: //    included in other modules.
 5: //
 6:
 7: #ifndef __UTIL_H__
 8: #define __UTIL_H__
 9:
10: #include <iostream>
11: #include <list>
12: #include <string>
13:
14: #ifdef __GNUC__
15: #include <stdexcept>
16: #endif
17:
18: using namespace std;
19:
20: #include "trace.h"
21:
22: //
23: // sys_info -
24: //    Keep track of execname and exit status.  Must be initialized
25: //    as the first thing done inside main.  Main should call:
26: //       sys_info::set_execname (argv[0]);
27: //    before anything else.
28: //
29:
30: class sys_info {
31:    public:
32:       static const string &get_execname ();
33:       static void set_exit_status (int status);
34:       static int get_exit_status ();
35:    private:
36:       friend int main (int argc, char **argv);
37:       static void set_execname (const string &argv0);
38:       static string *execname;
39:       static int exit_status;
40: };
41:
42: //
43: // datestring -
44: //    Return the current date, as printed by date(1).
45: //
46:
47: const string datestring ();
48:
49: //
50: // split -
51: //    Split a string into a list<string>..  Any sequence
52: //    of chars in the delimiter string is used as a separator.  To
53: //    Split a pathname, use "/".  To split a shell command, use " ".
54: //
55:
56: list<string> split (const string &line, const string &delimiter);
57:
58: //
59: // complain -
60: //    Used for starting error messages.  Sets the exit status to
61: //    EXIT_FAILURE, writes the program name to cerr, and then
62: //    returns the cerr ostream.  Example:
63: //       complain() << filename << ": some problem" << endl;
64: //
```

```
 65:
 66: ostream &complain();
 67:
 68: //
 69: // syscall_error -
 70: //    Complain about a failed system call.  Argument is the name
 71: //    of the object causing trouble.  The extern errno must contain
 72: //    the reason for the problem.
 73: //
 74:
 75: void syscall_error (const string &);
 76:
 77: //
 78: // operator<< (list) -
 79: //    An overloaded template operator which allows lists to be
 80: //    printed out as a single operator, each element separated from
 81: //    the next with spaces.  The item_t must have an output operator
 82: //    defined for it.
 83: //
 84:
 85: template <typename item_t>
 86: ostream &operator<< (ostream &out, const list<item_t> &vec);
 87:
 88: //
 89: // string to_string (thing) -
 90: //    Convert anything into a string if it has an ostream<< operator.
 91: //
 92:
 93: template <typename type>
 94: string to_string (const type &);
 95:
 96: //
 97: // thing from_string (cons string &) -
 98: //    Scan a string for something if it has an istream>> operator.
 99: //
100:
101: template <typename type>
102: type from_string (const string &);
103:
104: //
105: // Put the RCS Id string in the object file.
106: //
107:
108: RCSH(__util_h__,
109: "$Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $")
110:
111: #endif
112:
```

```
 1:
 2: //
 3: // Unlike Sun CC, Gnu g++ is not properly able to instantiate all
 4: // templates.  To fix this one problem in an ad hoc way, we
 5: // explicitly declare the missing instatiations.  A sample error
 6: // message in typical C++ unreadable template style follows.
 7: //
 8: // Note that his is a link time, not a compile time, error.  The
 9: // error messages have been edited with the addition of newlines to
10: // avoid random line wrap.  This is not needed with SUNWspro CC.
11: //
12: // No header file is needed, because this module is not needed by
13: // the compiler, only the Gnu linker.
14: //
15: //
16: // Undefined                     first referenced
17: //  symbol                           in file
18: // __cxa_get_exception_ptr             main.o
19: //
20: // std::basic_ostream<char, std::char_traits<char> >& operator<<
21: // <std::pair<inches, inches> >
22: // (std::basic_ostream<char, std::char_traits<char> >&,
23: // std::list<std::pair<inches, inches>,
24: // std::allocator<std::pair<inches, inches> > > const&)object.o
25: //
26: // double from_string<double>(std::basic_string<char,
27: // std::char_traits<char>, std::allocator<char> > const&)interp.o
28: //
29: // ld: fatal: Symbol referencing errors. No output written to draw
30: // collect2: ld returned 1 exit status
31: //
32:
33: #include "numbers.h"
34:
35: template ostream &operator<< (ostream &, const list<xycoords> &);
36: template double from_string <double> (const string &);
37:
38: RCSH(__templates_h__,
39: "$Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $")
40:
```

```
 1: // $Id: main.cc,v 1.2 2010-02-04 19:35:22-08 - - $
 2:
 3: #include <fstream>
 4: #include <iostream>
 5:
 6: using namespace std;
 7:
 8: #include "interp.h"
 9: #include "trace.h"
10: #include "util.h"
11:
12: //
13: // Parse a file.  Read lines from input file, parse each line,
14: // and interpret the command.
15: //
16:
17: void parsefile (const string &infilename,
18:                       istream &infile, ostream &outfile) {
19:    objectmap objmap;
20:    interpreter interp (infilename, outfile, objmap);
21:    for (int linenr = 1;; ++linenr) {
22:       try {
23:          string line;
24:          getline (infile, line);
25:          if (infile.eof()) break;
26:          if (line.size() == 0) continue;
27:          for (;;) {
28:             TRACE ('m', line);
29:             int last = line.size() - 1;
30:             if (line[last] != '\\') break;
31:             line[last] = ' ';
32:             string contin;
33:             getline (infile, contin);
34:             if (infile.eof()) break;
35:             line += contin;
36:          }
37:          list<string> words = split (line, " \t");
38:          if (words.size() == 0 || words.front()[0] == '#') continue;
39:          TRACE ('m', words);
40:          interp.interpret (words);
41:       }catch (runtime_error error) {
42:          complain() << infilename << ":" << linenr << ": "
43:                     << error.what() << endl;
44:       }
45:    }
46:    TRACE ('m', infilename << " EOF");
47: }
48:
49: //
50: // Strip off the dirname portion and the suffix and tack on .ps.
51: //
52: string get_outfilename (const string &infilename) {
53:       string suffix = ".dr";
54:       int slashpos = infilename.find_last_of ('/') + 1;
55:       string outname = infilename.substr (slashpos);
56:       int baselen = outname.size();
57:       int suffixlen = suffix.size();
58:       int difflen = baselen - suffixlen;
59:       if (baselen > suffixlen && outname.substr (difflen) == suffix) {
60:          outname = outname.substr (0, difflen);
61:       }
62:       return outname + ".ps";
63: }
64:
```

```
 65: //
 66: // Scan the option -D and check for operands.
 67: //
 68:
 69: void scan_options (int argc, char **argv) {
 70:    opterr = 0;
 71:    for (;;) {
 72:       int option = getopt (argc, argv, "@:");
 73:       if (option == EOF) break;
 74:       switch (option) {
 75:          case '@':
 76:             traceflags::setflags (optarg);
 77:             break;
 78:          default:
 79:             complain() << "-" << (char) optopt << ": invalid option"
 80:                        << endl;
 81:             break;
 82:       }
 83:    }
 84: }
 85:
 86: //
 87: // Main function.  Iterate over files if given, use cin if not.
 88: //
 89: int main (int argc, char **argv) {
 90:    sys_info::set_execname (argv[0]);
 91:    scan_options (argc, argv);
 92:    if (optind == argc) {
 93:       parsefile ("-", cin, cout);
 94:    }else {
 95:       for (int argi = optind; argi < argc; ++argi) {
 96:          const string infilename = argv[argi];
 97:          ifstream infile (infilename.c_str());
 98:          if (infile.fail()) {
 99:             syscall_error (infilename);
100:             continue;
101:          }
102:          const string outfilename = get_outfilename (infilename);
103:          ofstream outfile (outfilename.c_str());
104:          if (outfile.fail()) {
105:             syscall_error (outfilename);
106:             infile.close();
107:             continue;
108:          }
109:          TRACE ('m', infilename << " => " << outfilename);
110:          parsefile (infilename, infile, outfile);
111:          infile.close ();
112:          outfile.close ();
113:       }
114:    }
115:    return sys_info::get_exit_status ();
116: }
```

```
 1:
 2: #include <list>
 3: #include <map>
 4: #include <string>
 5:
 6: using namespace std;
 7:
 8: #include "interp.h"
 9: #include "object.h"
10: #include "util.h"
11:
12: interpreter::interpreter(const string &filename, ostream &outfile,
13:                          objectmap &objmap):
14:    outfile(outfile), pagenr(1), objmap(objmap), infilename(filename),
15:    page_xoffset (inches (.25)), page_yoffset (inches (.25)) {
16:    if (interpmap.size() == 0) {
17:       interpmap["define" ] = &interpreter::do_define ;
18:       interpmap["draw"   ] = &interpreter::do_draw   ;
19:       interpmap["newpage"] = &interpreter::do_newpage;
20:    }
21:    if (factorymap.size() == 0) {
22:       factorymap["text"     ] = &interpreter::make_text     ;
23:       factorymap["ellipse"  ] = &interpreter::make_ellipse  ;
24:       factorymap["circle"   ] = &interpreter::make_circle   ;
25:       factorymap["polygon"  ] = &interpreter::make_polygon  ;
26:       factorymap["rectangle"] = &interpreter::make_rectangle;
27:       factorymap["square"   ] = &interpreter::make_square   ;
28:       factorymap["line"     ] = &interpreter::make_line     ;
29:    }
30:    prolog ();
31:    startpage ();
32: }
33:
34: interpreter::~interpreter () {
35:    endpage ();
36:    epilog ();
37: }
38:
39: map <string, interpreter::interpreterfn> interpreter::interpmap;
40: map <string, interpreter::factoryfn> interpreter::factorymap;
41:
42: string shift (list<string> &words) {
43:    if (words.size() == 0) throw runtime_error ("syntax error");
44:    string front = words.front();
45:    words.pop_front();
46:    return front;
47: }
48:
49: void interpreter::interpret (parameters &params) {
50:    TRACE ('i', params);
51:    string command = shift (params);
52:    interpreterfn function = interpmap[command];
53:    if (function == NULL) throw runtime_error ("syntax error");
54:    (this->*function) (params);
55: }
56:
57: void interpreter::do_define (parameters &params) {
58:    TRACE ('i', params);
59:    string name = shift (params);
60:    objmap[name] = make_object (params);
61: }
62:
63: void interpreter::do_draw (parameters &params) {
64:    TRACE ('i', params);
```

```
 65:    string name = shift (params);
 66:    object *thing = objmap[name];
 67:    if (thing == NULL) throw runtime_error (name + ": no such object");
 68:    degrees angle = degrees (0);
 69:    if (params.size() == 3) {
 70:       angle = degrees (from_string<double> (params.back()));
 71:       params.pop_back();
 72:    }
 73:    if (params.size() != 2) throw runtime_error ("syntax error");
 74:    xycoords coords (inches (from_string<double> (params.front())),
 75:                     inches (from_string<double> (params.back())));
 76:    thing->draw (outfile, coords, angle);
 77: }
 78:
 79: void interpreter::do_newpage (parameters &params) {
 80:    if (params.size() != 0) throw runtime_error ("syntax error");
 81:    endpage ();
 82:    ++pagenr;
 83:    startpage ();
 84: }
 85:
 86: void interpreter::prolog () {
 87:    outfile << "%!PS-Adobe-3.0" << endl;
 88:    outfile << "%%Creator: " << sys_info::get_execname () << endl;
 89:    outfile << "%%CreationDate: " << datestring() << endl;
 90:    outfile << "%%PageOrder: Ascend" << endl;
 91:    outfile << "%%Orientation: Portrait" << endl;
 92:    outfile << "%%SourceFile: " << infilename << endl;
 93:    outfile << "%%EndComments" << endl;
 94: }
 95:
 96: void interpreter::startpage () {
 97:    outfile << endl;
 98:    outfile << "%%Page: " << pagenr << " " << pagenr << endl;
 99:    outfile << page_xoffset << " " << page_yoffset
100:            << " translate" << endl;
101:    outfile << "/Courier findfont 10 scalefont setfont" << endl;
102:    outfile << "0 0 moveto (" << infilename << ":"
103:            << pagenr << ") show" << endl;
104:
105: }
106:
107: void interpreter::endpage () {
108:    outfile << endl;
109:    outfile << "showpage" << endl;
110:    outfile << "grestoreall" << endl;
111: }
112:
113: void interpreter::epilog () {
114:    outfile << endl;
115:    outfile << "%%Trailer" << endl;
116:    outfile << "%%Pages: " << pagenr << endl;
117:    outfile << "%%EOF" << endl;
118:
119: }
120:
121: object *interpreter::make_object (parameters &command) {
122:    TRACE ('f', command);
123:    string type = shift (command);
124:    factoryfn func = factorymap[type];
125:    if (func == NULL) throw runtime_error (type + ": no such object");
126:    return (this->*func) (command);
127: }
128:
```

```
129: object *interpreter::make_text (parameters &command) {
130:    TRACE ('f', command);
131:    return new text ("", points(0), string());
132: }
133:
134: object *interpreter::make_ellipse (parameters &command) {
135:    TRACE ('f', command);
136:    return new ellipse (inches(0), inches(0), points(0));
137: }
138:
139: object *interpreter::make_circle (parameters &command) {
140:    TRACE ('f', command);
141:    return new circle (inches(0), points(0));
142: }
143:
144: object *interpreter::make_polygon (parameters &command) {
145:    TRACE ('f', command);
146:    return new polygon (coordlist(), points(0));
147: }
148:
149: object *interpreter::make_rectangle (parameters &command) {
150:    TRACE ('f', command);
151:    return new rectangle (inches(0), inches(0), points(0));
152: }
153:
154: object *interpreter::make_square (parameters &command) {
155:    TRACE ('f', command);
156:    return new square (inches(0), points(0));
157: }
158:
159: object *interpreter::make_line (parameters &command) {
160:    TRACE ('f', command);
161:    return new line (inches(0), points(0));
162: }
163:
164: RCSC(__interp_cc__,
165: "$Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $")
166:
```

```
 1:
 2: #include <cstdlib>
 3:
 4: using namespace std;
 5:
 6: #include "numbers.h"
 7: #include "util.h"
 8:
 9: ostream &operator<< (ostream &out, const degrees &that) {
10:    out << that.angle << "deg";
11:    return out;
12: }
13:
14: ostream &operator<< (ostream &out, const points &that) {
15:    out << that.pointvalue << "pt";
16:    return out;
17: }
18:
19: ostream &operator<< (ostream &out, const inches &that) {
20:    out << that.pointvalue / PTS_PER_INCH << "in";
21:    return out;
22: }
23:
24: ostream &operator<< (ostream &out, const xycoords &coords) {
25:    out << "(" << coords.first << "," << coords.second << ")";
26:    return out;
27: }
28:
29: RCSC(__numbers_cc__,
30: "$Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $")
```

```
 1:
 2: #include <typeinfo>
 3:
 4: using namespace std;
 5:
 6: #include "object.h"
 7: #include "util.h"
 8:
 9: #define WHOAMI \
10:         "[" << typeid(*this).name() << "@" << (void *) this << "]"
11:
12: #define CTRACE(ARGS) \
13:         TRACE ('c', WHOAMI << " " << ARGS)
14:
15: #define DTRACE(ARGS) \
16:         TRACE ('d', WHOAMI << " coords=" << coords \
17:                << " angle=" << angle << endl << ARGS);
18:
19: object::~object () {
20:    CTRACE ("delete");
21: }
22:
23: text::text (const string &font, const points &size, const string &data):
24:       fontname(font), fontsize(size), textdata(data) {
25:    CTRACE ("font=" << fontname << " size=" << fontsize
26:            << " \"" << textdata << "\"")
27: }
28:
29: ellipse::ellipse (const inches &initheight, const inches &initwidth,
30:                   const points &initthick):
31:       shape(initthick), height(initheight), width(initwidth) {
32:    CTRACE ("height=" << height << " width=" << width
33:            << " thick=" << thick);
34: }
35:
36: circle::circle (const inches &diameter, const points &thick):
37:       ellipse (diameter, diameter, thick) {
38: }
39:
40: polygon::polygon (const coordlist &coords, const points &initthick):
41:       shape(initthick), coordinates(coords) {
42:    CTRACE ( "thick=" << thick << " coords=" << endl
43:             << coordinates);
44: }
45:
46: rectangle::rectangle (const inches &height, const inches &width,
47:                       const points &initthick):
48:       polygon (make_list (height, width), initthick) {
49: }
50:
51: square::square (const inches &width, const points &thick):
52:       rectangle (width, width, thick) {
53: }
54:
55: line::line (const inches &length, const points &initthick):
56:       polygon (make_list (length), initthick) {
57:
58: }
59:
60: void text::draw (ostream &out, const xycoords &coords,
61:                 const degrees &angle) {
62:    DTRACE ("font=" << fontname << " size=" << fontsize
63:            << " \"" << textdata << "\"")
64: }
```

```
65:
66: void ellipse::draw (ostream &out, const xycoords &coords,
67:                 const degrees &angle) {
68:    DTRACE ("height=" << height << " width=" << width
69:            << " thick=" << thick);
70: }
71:
72: void polygon::draw (ostream &out, const xycoords &coords,
73:                 const degrees &angle) {
74:    DTRACE ( "thick=" << thick << " coords=" << endl
75:             << coordinates);
76: }
77:
78: coordlist rectangle::make_list (
79:             const inches &height, const inches &width) {
80:    coordlist coordlist;
81:    return coordlist;
82: }
83:
84: coordlist line::make_list (const inches &length) {
85:    coordlist coordlist;
86:    return coordlist;
87: }
88:
89:
90: RCSC(__object_cc__,
91: "$Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $")
```

```
 1:
 2: #include <climits>
 3: #include <iostream>
 4: #include <limits>
 5: #include <vector>
 6:
 7: using namespace std;
 8:
 9: #include "trace.h"
10:
11: //
12: // ** BUG IN STL ** BUG IN STL **
13: // We should use vector<bool> instead of vector<char>,
14: // but vector<bool> has a bug:
15: // http://forums.sun.com/thread.jspa?threadID=5277939
16: // Static linking works, but doubles the size of the executable
17: // image.
18: // ** BUG IN STL ** BUG IN STL **
19: //
20:
21: typedef vector<char> boolvec;
22: boolvec traceflags::flags (UCHAR_MAX + 1, false);
23: const boolvec trueflags (UCHAR_MAX + 1, true);
24:
25: void traceflags::setflags (const string &optflags) {
26:    string::const_iterator itor = optflags.begin();
27:    string::const_iterator end = optflags.end();
28:    for (; itor != end; ++itor) {
29:       if (*itor == '@') {
30:          flags = trueflags;
31:       }else {
32:          flags[*itor] = true;
33:       }
34:    }
35:    // Note that TRACE can trace setflags.
36:    TRACE ('t',  "optflags = " << optflags);
37: }
38:
39: //
40: // getflag -
41: //    Check to see if a certain flag is on.
42: //
43:
44: bool traceflags::getflag (char flag) {
45:    // Bug alert:
46:    // Don't TRACE this function or the stack will blow up.
47:    bool result = flags[flag];
48:    return result;
49: }
50:
51: RCSC(__trace_cc__,
52: "$Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $")
53:
```

```
 1:
 2: #include <cerrno>
 3: #include <cstdlib>
 4: #include <cstring>
 5: #include <ctime>
 6: #include <sstream>
 7: #include <stdexcept>
 8: #include <string>
 9: #include <typeinfo>
10:
11: using namespace std;
12:
13: #include "util.h"
14:
15: int sys_info::exit_status = EXIT_SUCCESS;
16: string *sys_info::execname = NULL; // Must be initialized from main().
17:
18: void sys_info_error (const string &condition) {
19:    throw logic_error ("main() has " + condition
20:                  + " called sys_info::set_execname()");
21: }
22:
23: void sys_info::set_execname (const string &argv0) {
24:    if (execname != NULL) sys_info_error ("already");
25:    int slashpos = argv0.find_last_of ('/') + 1;
26:    execname = new string (argv0.substr (slashpos));
27:    cout << boolalpha;
28:    cerr << boolalpha;
29:    TRACE ('u', "execname = " << execname);
30: }
31:
32: const string &sys_info::get_execname () {
33:    if (execname == NULL) sys_info_error ("not yet");
34:    return *execname;
35: }
36:
37: void sys_info::set_exit_status (int status) {
38:    if (execname == NULL) sys_info_error ("not yet");
39:    exit_status = status;
40: }
41:
42: int sys_info::get_exit_status () {
43:    if (execname == NULL) sys_info_error ("not yet");
44:    return exit_status;
45: }
46:
47: const string datestring () {
48:    time_t clock = time (NULL);
49:    struct tm *tm_ptr = localtime (&clock);
50:    char timebuf[128];
51:    size_t bufsize = strftime (timebuf, sizeof timebuf,
52:           "%a %b %e %H:%M:%S %Z %Y", tm_ptr);
53:    return timebuf;
54: }
55:
56: list<string> split (const string &line, const string &delimiters) {
57:    list<string> words;
58:    int end = 0;
59:    // Loop over the string, splitting out words, and for each word
60:    // thus found, append it to the output list<string>.
61:    for (;;) {
62:       int start = line.find_first_not_of (delimiters, end);
63:       if (start == string::npos) break;
64:       end = line.find_first_of (delimiters, start);
```

```
 65:       words.push_back (line.substr (start, end - start));
 66:    }
 67:    TRACE ('u', words);
 68:    return words;
 69: }
 70:
 71: ostream &complain() {
 72:    sys_info::set_exit_status (EXIT_FAILURE);
 73:    cerr << sys_info::get_execname () << ": ";
 74:    return cerr;
 75: }
 76:
 77: void syscall_error (const string &object) {
 78:    complain() << object << ": " << strerror (errno) << endl;
 79: }
 80:
 81: template <typename item_t>
 82: ostream &operator<< (ostream &out, const list<item_t> &vec) {
 83:    typename list<item_t>::const_iterator itor = vec.begin();
 84:    typename list<item_t>::const_iterator end = vec.end();
 85:    // If the list is empty, do nothing.
 86:    if (itor != end) {
 87:       // Print out the first element without a space.
 88:       out << *itor++;
 89:       // Print out the rest of the elements each preceded by a space.
 90:       while (itor != end) out << " " << *itor++;
 91:    }
 92:    return out;
 93: }
 94:
 95: template <typename type>
 96: string to_string (const type &that) {
 97:    ostringstream stream;
 98:    stream << that;
 99:    return stream.str ();
100: }
101:
102: template <typename type>
103: type from_string (const string &that) {
104:    stringstream stream;
105:    stream << that;
106:    type result;
107:    if ( !(stream >> result   // Can we read type from string?
108:        && stream >> std::ws  // Flush trailing white space.
109:        && stream.eof ())     // Must now be at end of stream.
110:    ) {
111:       throw domain_error (string (typeid (type).name ())
112:             + " from_string (" + that + ")");
113:    }
114:    return result;
115: }
116:
117: #include "templates.h"
118:
119: RCSC(__util_cc__,
120: "$Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $")
121:
```

```
# $Id: Makefile,v 1.8 2010-02-05 14:34:31-08 - - $

MKFILE      = Makefile
DEPSFILE    = ${MKFILE}.deps
NOINCL      = ci clean spotless
NEEDINCL    = ${filter ${NOINCL}, ${MAKECMDGOALS}}
GMAKE       = ${MAKE} --no-print-directory
UNAME      ?= ${shell uname -s}

ifeq (${UNAME},SunOS)
COMPILECCC  = CC -g -features=extensions
MAKEDEPSCCC = CC -xM1
endif
ifeq (${UNAME},Linux)
COMPILECCC  = g++ -g
MAKEDEPSCCC = g++ -MM
endif

CCHEADER    = interp.h numbers.h object.h trace.h util.h
CCSOURCE    = main.cc ${CCHEADER:.h=.cc}
EXECBIN     = draw
OBJECTS     = ${CCSOURCE:.cc=.o}
OTHERS      = ${MKFILE} README
ALLSOURCES  = ${CCHEADER} templates.h ${CCSOURCE} ${OTHERS}
LISTFILES   = ${ALLSOURCES} ${DEPSFILE} Idents ../data/draw.perl

LISTING     = ../asg3-draw.code.ps
CLASS       = cmps109-wm.w09
PROJECT     = asg3

all : ${EXECBIN}
        - checksource ${ALLSOURCES}

${EXECBIN} : ${OBJECTS}
        ${COMPILECCC} -o $@ ${OBJECTS}

%.o : %.cc
        ${COMPILECCC} -c $<

ci : ${ALLSOURCES}
        @ - checksource ${ALLSOURCES}
        cid + ${ALLSOURCES}

lis : ${ALLSOURCES}
        ${GMAKE} idents >Idents
        mkpspdf ${LISTING} ${LISTFILES}
        - rm Idents

clean :
        - rm ${OBJECTS} ${DEPSFILE} core ${EXECBIN}.errs

spotless : clean
        - rm ${EXECBIN}

submit : ${ALLSOURCES}
        - checksource ${ALLSOURCES}
        submit ${CLASS} ${PROJECT} ${ALLSOURCES}
        testsubmit ${CLASS} ${PROJECT} ${ALLSOURCES}

deps : ${CCSOURCE} ${CCHEADER}
        @ echo "# ${DEPSFILE} created ‘LC_TIME=C date‘" >${DEPSFILE}
        ${MAKEDEPSCCC} ${CCSOURCE} | sort | uniq >>${DEPSFILE}

${DEPSFILE} :
```

```
65:         @ touch ${DEPSFILE}
66:         ${GMAKE} deps
67:
68: idents : ${ALLSOURCES} ${OBJECTS} ${EXECBIN}
69:         ldd ${EXECBIN}
70:         ident ${ALLSOURCES} ${OBJECTS} ${EXECBIN}
71:
72: again :
73:         ${GMAKE} spotless deps ci all lis
74:
75: ifeq (${NEEDINCL}, )
76: include ${DEPSFILE}
77: endif
78:
```

```
1: $Id: README,v 1.1 2010-01-29 18:07:32-08 - - $
```

```
  1: # Makefile.deps created Mon Feb  8 13:45:19 PST 2010
  2: interp.o : interp.cc
  3: interp.o : interp.h
  4: interp.o : numbers.h
  5: interp.o : object.h
  6: interp.o : trace.h
  7: interp.o : util.h
  8: main.o : interp.h
  9: main.o : main.cc
 10: main.o : numbers.h
 11: main.o : object.h
 12: main.o : trace.h
 13: main.o : util.h
 14: numbers.o : numbers.cc
 15: numbers.o : numbers.h
 16: numbers.o : trace.h
 17: numbers.o : util.h
 18: object.o : numbers.h
 19: object.o : object.cc
 20: object.o : object.h
 21: object.o : trace.h
 22: object.o : util.h
 23: trace.o : trace.cc
 24: trace.o : trace.h
 25: util.o : numbers.h
 26: util.o : templates.h
 27: util.o : trace.h
 28: util.o : util.cc
 29: util.o : util.h
```

```
    1: ldd draw
    2:         libCstd.so.1 =>  /usr/lib/libCstd.so.1
    3:         libCrun.so.1 =>  /usr/lib/libCrun.so.1
    4:         libm.so.2 =>     /lib/libm.so.2
    5:         libc.so.1 =>     /lib/libc.so.1
    6: ident interp.h numbers.h object.h trace.h util.h templates.h main.cc interp.cc n
umbers.cc object.cc trace.cc util.cc Makefile README main.o interp.o numbers.o object.o
 trace.o util.o draw
    7: interp.h:
    8:      $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
    9:
   10: numbers.h:
   11:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
   12:
   13: object.h:
   14:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
   15:
   16: trace.h:
   17:      $Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $
   18:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
   19:
   20: util.h:
   21:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
   22:
   23: templates.h:
   24:      $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $
   25:
   26: main.cc:
   27:      $Id: main.cc,v 1.2 2010-02-04 19:35:22-08 - - $
   28:
   29: interp.cc:
   30:      $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $
   31:
   32: numbers.cc:
   33:      $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $
   34:
   35: object.cc:
   36:      $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $
   37:
   38: trace.cc:
   39:      $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $
   40:
   41: util.cc:
   42:      $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $
   43:
   44: Makefile:
   45:      $Id: Makefile,v 1.8 2010-02-05 14:34:31-08 - - $
   46:
   47: README:
   48:      $Id: README,v 1.1 2010-01-29 18:07:32-08 - - $
   49:
   50: main.o:
   51:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
   52:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
   53:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
   54:      $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
   55:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
   56:
   57: interp.o:
   58:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
   59:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
   60:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
   61:      $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
   62:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
```

```
  63:      $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $
  64:      $Compiled: interp.cc Feb  8 2010 13:45:21 $
  65:
  66: numbers.o:
  67:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
  68:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
  69:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
  70:      $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $
  71:      $Compiled: numbers.cc Feb  8 2010 13:45:22 $
  72:
  73: object.o:
  74:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
  75:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
  76:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
  77:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
  78:      $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $
  79:      $Compiled: object.cc Feb  8 2010 13:45:22 $
  80:
  81: trace.o:
  82:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
  83:      $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $
  84:      $Compiled: trace.cc Feb  8 2010 13:45:23 $
  85:
  86: util.o:
  87:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
  88:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
  89:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
  90:      $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $
  91:      $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $
  92:      $Compiled: util.cc Feb  8 2010 13:45:23 $
  93:
  94: draw:
  95:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
  96:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
  97:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
  98:      $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
  99:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
 100:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
 101:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
 102:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
 103:      $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
 104:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
 105:      $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $
 106:      $Compiled: interp.cc Feb  8 2010 13:45:21 $
 107:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
 108:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
 109:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
 110:      $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $
 111:      $Compiled: numbers.cc Feb  8 2010 13:45:22 $
 112:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
 113:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
 114:      $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
 115:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
 116:      $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $
 117:      $Compiled: object.cc Feb  8 2010 13:45:22 $
 118:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
 119:      $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $
 120:      $Compiled: trace.cc Feb  8 2010 13:45:23 $
 121:      $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
 122:      $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
 123:      $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
 124:      $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $
 125:      $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $
 126:      $Compiled: util.cc Feb  8 2010 13:45:23 $
```

```
 1: #!/usr/bin/perl
 2: # $Id: draw.perl,v 1.2 2010-01-29 18:15:03-08 - - $
 3:
 4: use strict;
 5: use warnings;
 6: use POSIX qw (strftime);
 7: use Data::Dumper;
 8:
 9: my $script = $0;
10: my $date = strftime "%a %b %e %H:%M:%S %Z %Y", localtime;
11: my $debug = 0;
12:
13: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
14: my $exit_status = 0;
15: END{ exit $exit_status; }
16: sub note(@){ print STDERR "$0: @_"; };
17: $SIG{'__WARN__'} = sub{ note @_;; $exit_status = 1; };
18: $SIG{'__DIE__'} = sub{ warn @_; exit; };
19:
20: sub startpage ($) {
21:    my ($state) = @_;
22:    my $outfile = $state->{OUTFILE};
23:    print $outfile
24:       "%%Page: $state->{PAGENR} $state->{PAGENR}\n",
25:       "18 18 translate\n",
26:       "/Courier findfont 10 scalefont setfont\n",
27:       "0 0 moveto ($state->{INFILENAME}:$state->{PAGENR}) show\n";
28: }
29:
30: sub endpage ($) {
31:    my ($state) = @_;
32:    my $outfile = $state->{OUTFILE};
33:    print $outfile
34:       "showpage\n",
35:       "grestoreall\n";
36: }
37:
38: sub prolog ($) {
39:    my ($state) = @_;
40:    my $outfile = $state->{OUTFILE};
41:    print $outfile
42:       "%!PS-Adobe-3.0\n",
43:       "%%Creator: $script\n",
44:       "%%CreationDate: $date\n",
45:       "%%PageOrder: Ascend\n",
46:       "%%Orientation: Portrait\n",
47:       "%%SourceFile: $state->{INFILENAME}\n",
48:       "%%EndComments\n";
49:    $state->{PAGENR} = 1;
50:    startpage $state;
51: }
52:
53: sub epilog ($) {
54:    my ($state) = @_;
55:    my $outfile = $state->{OUTFILE};
56:    endpage $state;
57:    print $outfile
58:       "%%Trailer\n",
59:       "%%Pages: $state->{PAGENR}\n",
60:       "%%EOF\n",
61: }
62:
63: sub error ($;$) {
64:    my ($state, $message) = @_;
```

```
 65:    $message = "syntax error" unless $message;
 66:    warn "$state->{INFILENAME}: $.: $message\n";
 67: }
 68:
 69: sub numeric ($) {
 70:    my ($number) = @_;
 71:    return $number =~ m/^[+-]?(\d+\.?\d*|\.\d+)([Ee][+-]?\d+)?$/;
 72: }
 73:
 74: sub numbers ($$) {
 75:    my ($count, $numbers) = @_;
 76:    return 0 if $count != @$numbers and $count > 0;
 77:    return ! grep {! numeric $_} @$numbers;
 78: }
 79:
 80: sub make_text ($$) {
 81:    my ($state, $words) = @_;
 82:    my $size = (numeric $words->[0]) ? (shift @$words) : 12;
 83:    my $font = shift @$words;
 84:    do {error $state; return} unless @$words >= 1;
 85:    my $text = join ' ', @$words;
 86:    $text =~ s/[(\\)]/\\$&/g;
 87:    my $outfile = $state->{OUTFILE};
 88:    return sub {
 89:       my ($place) = @_;
 90:       my ($x0, $y0, $angle) = @$place;
 91:       map {$_ *= 72} $x0, $y0;
 92:       $angle = 0 unless $angle;
 93:       print $outfile
 94:             "gsave\n",
 95:             "   /$font findfont\n",
 96:             "   $size scalefont setfont\n",
 97:             "   $x0 $y0 translate\n",
 98:             "   $angle rotate\n",
 99:             "   0 0 moveto\n",
100:             "   ($text)\n",
101:             "   show\n",
102:             "grestore\n";
103:    }
104: }
105:
106: sub make_ellipse ($$) {
107:    my ($state, $words) = @_;
108:    push @$words, 2 unless @$words == 3;
109:    do {error $state; return} unless numbers 3, $words;
110:    my ($height, $width, $thick) = @$words;
111:    map {$_ *= 72} $height, $width;
112:    do {error $state, "syntax error height"; return} if $height == 0;
113:    my ($xscale, $yscale, $radius);
114:    if ($height < $width) {
115:       $xscale = 1;
116:       $yscale = $height / $width;
117:       $radius = $width / 2;
118:    }else {
119:       $xscale = $width / $height;
120:       $yscale = 1;
121:       $radius = $height / 2;
122:    }
123:    my $outfile = $state->{OUTFILE};
124:    return sub {
125:       my ($place) = @_;
126:       my ($x0, $y0, $angle) = @$place;
127:       map {$_ *= 72} $x0, $y0, $width;
128:       $angle = 0 unless $angle;
```

```
129:       print $outfile
130:             "gsave\n",
131:             "   newpath\n",
132:             "   /save matrix currentmatrix def\n",
133:             "   $x0 $y0 translate\n",
134:             "   $angle rotate\n",
135:             "   $xscale $yscale scale\n",
136:             "   0 0 $radius 0 360 arc\n",
137:             "   save setmatrix\n",
138:             "   $thick setlinewidth\n",
139:             "   stroke\n",
140:             "grestore\n";
141:    }
142: }
143:
144: sub make_circle ($$) {
145:    my ($state, $words) = @_;
146:    unshift @$words, $words->[0];
147:    return make_ellipse $state, $words;
148: }
149:
150: sub make_polygon ($$) {
151:    my ($state, $words) = @_;
152:    do {error $state; return} unless @$words >= 2 and numbers 0, $words;
153:    my $thick = (@$words % 2 == 0) ? 2 : pop @$words;
154:    my $outfile = $state->{OUTFILE};
155:    return sub {
156:       my ($place) = @_;
157:       my ($x0, $y0, $angle) = @$place;
158:       map {$_ *= 72} $x0, $y0;
159:       $angle = 0 unless $angle;
160:       print $outfile
161:             "gsave\n",
162:             "   newpath\n",
163:             "   $x0 $y0 translate\n",
164:             "   $angle rotate\n",
165:             "   0 0 moveto\n";
166:       for (my $gon = 0; $gon < @$words; $gon += 2) {
167:          my ($xrel, $yrel) = @{$words}[$gon, $gon + 1];
168:          map {$_ *= 72} $xrel, $yrel;
169:          print $outfile
170:             "   $xrel $yrel rlineto\n";
171:       }
172:       print $outfile
173:             "   closepath\n",
174:             "   $thick setlinewidth\n",
175:             "   stroke\n",
176:             "grestore\n";
177:    }
178: }
179:
180: sub make_rectangle ($$) {
181:    my ($state, $words) = @_;
182:    push @$words, 2 unless @$words == 3;
183:    do {error $state; return} unless numbers 3, $words;
184:    my ($hght, $wid, $thick) = @$words;
185:    return make_polygon $state, [0, $hght, $wid, 0, 0, - $hght, $thick];
186: }
187:
188: sub make_square ($$) {
189:    my ($state, $words) = @_;
190:    unshift @$words, $words->[0];
191:    return make_rectangle $state, $words;
192: }
```

```
193:
194: sub make_line ($$) {
195:    my ($state, $words) = @_;
196:    push @$words, 2 unless @$words == 2;
197:    do {error $state; return} unless numbers 2, $words;
198:    my ($length, $thick) = @$words;
199:    $thick = 2 unless $thick;
200:    return make_polygon $state, [$words->[0], 0, $thick];
201: }
202:
203: my %objects = (
204:    "text" => \&make_text,
205:    "ellipse" => \&make_ellipse,
206:    "circle" => \&make_circle,
207:    "polygon" => \&make_polygon,
208:    "rectangle" => \&make_rectangle,
209:    "square" => \&make_square,
210:    "line" => \&make_line,
211: );
212:
213: sub do_define ($$) {
214:    my ($state, $words) = @_;
215:    do {error $state; return} unless @$words > 2;
216:    my ($name) = shift @$words;
217:    my ($class) = shift @$words;
218:    my ($maker) = $objects{$class};
219:    do {error $state, "no such shape"; return} unless $maker;
220:    my $object = $maker->($state, $words);
221:    $state->{SYMTAB}->{$name} = $object if $object;
222: }
223:
224: sub do_draw ($$) {
225:    my ($state, $words) = @_;
226:    my $object = $state->{SYMTAB}->{shift @$words};
227:    do {error $state, "no such object"; return} unless $object;
228:    push @$words, 0 if @$words == 2;
229:    do {error $state; return} unless numbers 3, $words;
230:    $object->($words);
231: }
232:
233: sub do_newpage ($$) {
234:    my ($state, $words) = @_;
235:    do {error $state; return} unless @$words == 0;
236:    endpage $state;
237:    $state->{PAGENR}++;
238:    startpage $state;
239: }
240:
241: my %commands = (
242:    "define" => \&do_define,
243:    "draw" => \&do_draw,
244:    "newpage" => \&do_newpage,
245: );
246:
247: sub parsefile ($) {
248:    my ($state) = @_;
249:    prolog $state;
250:    my $infile = $state->{INFILE};
251:    my $outfile = $state->{OUTFILE};
252:    while (defined (my $line = <$infile>)) {
253:       for (;;) {
254:          chomp $line;
255:          last unless $line =~ s/\\$//;
256:          my $contin = <$infile>;
```

```
257:          $line .= $contin;
258:       }
259:       chomp $line;
260:       print $outfile "\n",
261:             "%%Command[$.]: $line\n";
262:       next if $line =~ m/^\s*(#|$)/;
263:       my @words = split " ", $line;
264:       my $command = $commands{shift @words};
265:       do {error $state; next} unless defined $command;
266:       $command->($state, \@words);
267:    }
268:    epilog $state;
269: }
270:
271: sub main () {
272:    @ARGV and $ARGV[0] =~ m/^-D/ and $debug = shift;
273:    unless (@ARGV) {
274:       parsefile {INFILENAME => "-",
275:                  INFILE => *STDIN,
276:                  OUTFILE => *STDOUT};
277:    }else {
278:       for my $infilename (@ARGV) {
279:          my $outfilename = $infilename;
280:          $outfilename =~ s|^.*/([^/]+)/*$|$1|;
281:          $outfilename =~ s/(\.dr)?$/.ps/;
282:          open my $infile, "<$infilename"
283:               or warn "$infilename: $!\n" and next;
284:          open my $outfile, ">$outfilename"
285:               or warn "$outfilename: $!\n" and next;
286:          print "$0: $infilename => $outfilename\n";
287:          parsefile {INFILENAME => $infilename,
288:                     INFILE => $infile,
289:                     OUTFILE => $outfile};
290:          close $infile;
291:          close $outfile;
292:       }
293:    }
294: }
295:
296: main;
297:
```