```

#ifndef __INTERP_H__
#define __INTERP_H__

#include <iostream>
#include <map>

using namespace std;

#include "object.h"
#include "trace.h"

typedef map <string, object *> objectmap;

class interpreter {
   public:
      typedef list<string> parameters;
      interpreter (const string &, ostream &, objectmap &);
      ~interpreter ();
      void interpret (parameters &);
   private:
      interpreter (); // Disable
      interpreter (const interpreter &); // Disable
      interpreter &operator= (const interpreter &); // Disable

      // Data fields.
      typedef void (interpreter::*interpreterfn) (parameters &);
      typedef object *(interpreter::*factoryfn) (parameters &);
      static map <string, interpreterfn> interpmap;
      static map <string, factoryfn> factorymap;
      ostream &outfile;
      int pagenr;
      objectmap objmap;
      string infilename;
      double page_xoffset;
      double page_yoffset;

      // Service functions.
      void do_define (parameters &);
      void do_draw (parameters &);
      void do_newpage (parameters &);
      void prolog ();
      void startpage ();
      void endpage ();
      void epilog ();

      // Factory functions.
      object *make_object (parameters &);
      object *make_text (parameters &);
      object *make_ellipse (parameters &);
      object *make_circle (parameters &);
      object *make_polygon (parameters &);
      object *make_rectangle (parameters &);
      object *make_square (parameters &);
      object *make_line (parameters &);
};

RCSH(__interp_h__,
"$Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $")

#endif

```

```

#ifndef __NUMBERS_H__
#define __NUMBERS_H__

#include <iostream>
#include <utility>

using namespace std;

#include "trace.h"

const double PTS_PER_INCH = 72;

class degrees {
   friend ostream &operator<< (ostream &, const degrees &);
   public:
      explicit degrees (double init): angle(init) {}
      operator double() {return angle; }
   private:
      degrees (); // Disable.
      double angle;
};

class points {
   friend ostream &operator<< (ostream &, const points &);
   public:
      explicit points (double init): pointvalue(init) {}
      operator double() {return pointvalue; }
   private:
      points (); // Disable.
      double pointvalue;
};

class inches {
   friend ostream &operator<< (ostream &, const inches &);
   public:
      explicit inches (double init): pointvalue(init * PTS_PER_INCH) {}
      operator double() {return pointvalue; }
   private:
      inches (); // Disable.
      double pointvalue;
};

typedef pair <inches, inches> xycoords;

ostream &operator<< (ostream &, const xycoords &);

RCSH(__numbers_h__,
"$Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $")

#endif

```

```

#ifndef __OBJECT_H__
#define __OBJECT_H__

#include <iomanip>
#include <iostream>
#include <list>
#include <utility>

using namespace std;

#include "numbers.h"

//
// Objects constitute a single-inheritance hierarchy, summarized
// here, with the superclass listed first, and subclasses indented
// under their immediate superclass.
//
// object
//    test
//    shape
//       ellipse
//          circle
//       polygon
//          rectangle
//             square
//          line
//

typedef list<xycoords> coordlist;

//
// Abstract base class for all shapes in this system.
//

class object {
   public:
      virtual ˜object ();
      virtual void draw (ostream &, const xycoords &,
                         const degrees &angle) = 0;
   protected:
      object () {}
};

//
// Class for printing text.
//

class text: public object {
   public:
      text (const string &fontname, const points &fontsize,
            const string &textdata);
      virtual void draw (ostream &, const xycoords &,
                         const degrees &angle);
   protected:
      string fontname;
      points fontsize;
      string textdata;
};

//
// Shape of a geometric object.
//

```

```
class shape: public object {
   protected:
      shape (const points &thick): thick(thick) {}
      points thick;
   private:
      shape (); // Disable.
};

//
// Classes for ellipse and circle.
//

class ellipse: public shape {
   public:
      ellipse (const inches &height, const inches &width,
               const points &thick);
      virtual void draw (ostream &, const xycoords &,
                         const degrees &angle);
   protected:
      inches height;
      inches width;
};

class circle: public ellipse {
   public:
      circle (const inches &diameter, const points &thick);
};

//
// Class polygon.
//

class polygon: public shape {
   public:
      polygon (const coordlist &coords, const points &thick);
      virtual void draw (ostream &, const xycoords &,
                         const degrees &angle);
   protected:
      const coordlist coordinates;
};

//
// Classes rectangle, square, and line..
//

class rectangle: public polygon {
   public:
      rectangle (const inches &height, const inches &width,
                 const points &thick);
   private:
      static coordlist make_list (
             const inches &height, const inches &width);
};

class square: public rectangle {
   public:
      square (const inches &width, const points &thick);
};

class line: public polygon {
   public:
      line (const inches &length, const points &thick);
   private:
      static coordlist make_list (const inches &length);
};

RCSH(__object_h__,
"$Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $")

#endif

```

```

#ifndef __TRACE_H__
#define __TRACE_H__

#include <vector>

using namespace std;

//
// traceflags -
//    static class for maintaining global trace flags, each indicated
//    by a single character.
// setflags -
//    Takes a string argument, and sets a flag for each char in the
//    string.  As a special case, '@', sets all flags.
// getflag -
//    Used by the TRACE macro to check to see if a flag has been set.
//    Not to be called by user code.
//

class traceflags {
   private:
      static vector<char> flags;
   public:
      static void setflags (const string &optflags);
      static bool getflag (char flag);
};

//
// TRACE -
//    Macro which expands into trace code.  First argument is a
//    trace flag char, second argument is output code that can
//    be sandwiched between <<.  Beware of operator precedence.
//    Example:
//       TRACE ('u', "foo = " << foo);
//    will print two words and a newline if flag 'u' is  on.
//    Traces are preceded by filename, line number, and function.
//

#define TRACE(FLAG,CODE) { \
           if (traceflags::getflag (FLAG)) { \
              cerr << __FILE__ << ":" << __LINE__ << ":" \
                   << __func__ << ": "; \
              cerr << CODE << endl; \
           } \
        }

//
// RCSH, RCSC -
//    Macros which allow RCS Id information to transfer to object
//    files and executable binaries.
//

#define RCSH(NAME,ID) \
static const char __RCS_##NAME[] = "\0" ID;
#define RCSC(NAME,ID) \
static const char __RCS_C_##NAME[] = "\0" ID \
"\0$Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $";

RCSH(__trace_h__,
"$Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $")

#endif

```

```
//
// util -
//    A utility class to provide various services not conveniently
//    included in other modules.
//

#ifndef __UTIL_H__
#define __UTIL_H__

#include <iostream>
#include <list>
#include <string>

#ifdef __GNUC__
#include <stdexcept>
#endif

using namespace std;

#include "trace.h"

//
// sys_info -
//    Keep track of execname and exit status.  Must be initialized
//    as the first thing done inside main.  Main should call:
//       sys_info::set_execname (argv[0]);
//    before anything else.
//

class sys_info {
   public:
      static const string &get_execname ();
      static void set_exit_status (int status);
      static int get_exit_status ();
   private:
      friend int main (int argc, char **argv);
      static void set_execname (const string &argv0);
      static string *execname;
      static int exit_status;
};

//
// datestring -
//    Return the current date, as printed by date(1).
//

const string datestring ();

//
// split -
//    Split a string into a list<string>..  Any sequence
//    of chars in the delimiter string is used as a separator.  To
//    Split a pathname, use "/".  To split a shell command, use " ".
//

list<string> split (const string &line, const string &delimiter);

//
// complain -
//    Used for starting error messages.  Sets the exit status to
//    EXIT_FAILURE, writes the program name to cerr, and then
//    returns the cerr ostream.  Example:
//       complain() << filename << ": some problem" << endl;
//
```

```

ostream &complain();

//
// syscall_error -
//    Complain about a failed system call.  Argument is the name
//    of the object causing trouble.  The extern errno must contain
//    the reason for the problem.
//

void syscall_error (const string &);

//
// operator<< (list) -
//    An overloaded template operator which allows lists to be
//    printed out as a single operator, each element separated from
//    the next with spaces.  The item_t must have an output operator
//    defined for it.
//

template <typename item_t>
ostream &operator<< (ostream &out, const list<item_t> &vec);

//
// string to_string (thing) -
//    Convert anything into a string if it has an ostream<< operator.
//

template <typename type>
string to_string (const type &);

//
// thing from_string (cons string &) -
//    Scan a string for something if it has an istream>> operator.
//

template <typename type>
type from_string (const string &);

//
// Put the RCS Id string in the object file.
//

RCSH(__util_h__,
"$Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $")

#endif

```

```

//
// Unlike Sun CC, Gnu g++ is not properly able to instantiate all
// templates.  To fix this one problem in an ad hoc way, we
// explicitly declare the missing instatiations.  A sample error
// message in typical C++ unreadable template style follows.
//
// Note that his is a link time, not a compile time, error.  The
// error messages have been edited with the addition of newlines to
// avoid random line wrap.  This is not needed with SUNWspro CC.
//
// No header file is needed, because this module is not needed by
// the compiler, only the Gnu linker.
//
//
// Undefined                    first referenced
//  symbol                          in file
// __cxa_get_exception_ptr             main.o
//
// std::basic_ostream<char, std::char_traits<char> >& operator<<
// <std::pair<inches, inches> >
// (std::basic_ostream<char, std::char_traits<char> >&,
// std::list<std::pair<inches, inches>,
// std::allocator<std::pair<inches, inches> > > const&)object.o
//
// double from_string<double>(std::basic_string<char,
// std::char_traits<char>, std::allocator<char> > const&)interp.o
//
// ld: fatal: Symbol referencing errors. No output written to draw
// collect2: ld returned 1 exit status
//

#include "numbers.h"

template ostream &operator<< (ostream &, const list<xycoords> &);
template double from_string <double> (const string &);

RCSH(__templates_h__,
"$Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $")

```

```
// $Id: main.cc,v 1.2 2010-02-04 19:35:22-08 - - $

#include <fstream>
#include <iostream>

using namespace std;

#include "interp.h"
#include "trace.h"
#include "util.h"

//
// Parse a file.  Read lines from input file, parse each line,
// and interpret the command.
//

void parsefile (const string &infilename,
                     istream &infile, ostream &outfile) {
   objectmap objmap;
   interpreter interp (infilename, outfile, objmap);
   for (int linenr = 1;; ++linenr) {
      try {
         string line;
         getline (infile, line);
         if (infile.eof()) break;
         if (line.size() == 0) continue;
         for (;;) {
            TRACE ('m', line);
            int last = line.size() - 1;
            if (line[last] != '\\') break;
            line[last] = ' ';
            string contin;
            getline (infile, contin);
            if (infile.eof()) break;
            line += contin;
         }
         list<string> words = split (line, " \t");
         if (words.size() == 0 || words.front()[0] == '#') continue;
         TRACE ('m', words);
         interp.interpret (words);
      }catch (runtime_error error) {
         complain() << infilename << ":" << linenr << ": "
                    << error.what() << endl;
      }
   }
   TRACE ('m', infilename << " EOF");
}

//
// Strip off the dirname portion and the suffix and tack on .ps.
//
string get_outfilename (const string &infilename) {
      string suffix = ".dr";
      int slashpos = infilename.find_last_of ('/') + 1;
      string outname = infilename.substr (slashpos);
      int baselen = outname.size();
      int suffixlen = suffix.size();
      int difflen = baselen - suffixlen;
      if (baselen > suffixlen && outname.substr (difflen) == suffix) {
         outname = outname.substr (0, difflen);
      }
      return outname + ".ps";
}

```

```
//
// Scan the option -D and check for operands.
//

void scan_options (int argc, char **argv) {
   opterr = 0;
   for (;;) {
      int option = getopt (argc, argv, "@:");
      if (option == EOF) break;
      switch (option) {
         case '@':
            traceflags::setflags (optarg);
            break;
         default:
            complain() << "-" << (char) optopt << ": invalid option"
                       << endl;
            break;
      }
   }
}

//
// Main function.  Iterate over files if given, use cin if not.
//
int main (int argc, char **argv) {
   sys_info::set_execname (argv[0]);
   scan_options (argc, argv);
   if (optind == argc) {
      parsefile ("-", cin, cout);
   }else {
      for (int argi = optind; argi < argc; ++argi) {
         const string infilename = argv[argi];
         ifstream infile (infilename.c_str());
         if (infile.fail()) {
            syscall_error (infilename);
            continue;
         }
         const string outfilename = get_outfilename (infilename);
         ofstream outfile (outfilename.c_str());
         if (outfile.fail()) {
            syscall_error (outfilename);
            infile.close();
            continue;
         }
         TRACE ('m', infilename << " => " << outfilename);
         parsefile (infilename, infile, outfile);
         infile.close ();
         outfile.close ();
      }
   }
   return sys_info::get_exit_status ();
}
```

```

#include <list>
#include <map>
#include <string>

using namespace std;

#include "interp.h"
#include "object.h"
#include "util.h"

interpreter::interpreter(const string &filename, ostream &outfile,
                         objectmap &objmap):
   outfile(outfile), pagenr(1), objmap(objmap), infilename(filename),
   page_xoffset (inches (.25)), page_yoffset (inches (.25)) {
   if (interpmap.size() == 0) {
      interpmap["define" ] = &interpreter::do_define ;
      interpmap["draw"   ] = &interpreter::do_draw   ;
      interpmap["newpage"] = &interpreter::do_newpage;
   }
   if (factorymap.size() == 0) {
      factorymap["text"     ] = &interpreter::make_text     ;
      factorymap["ellipse"  ] = &interpreter::make_ellipse  ;
      factorymap["circle"   ] = &interpreter::make_circle   ;
      factorymap["polygon"  ] = &interpreter::make_polygon  ;
      factorymap["rectangle"] = &interpreter::make_rectangle;
      factorymap["square"   ] = &interpreter::make_square   ;
      factorymap["line"     ] = &interpreter::make_line     ;
   }
   prolog ();
   startpage ();
}

interpreter::~interpreter () {
   endpage ();
   epilog ();
}

map <string, interpreter::interpreterfn> interpreter::interpmap;
map <string, interpreter::factoryfn> interpreter::factorymap;

string shift (list<string> &words) {
   if (words.size() == 0) throw runtime_error ("syntax error");
   string front = words.front();
   words.pop_front();
   return front;
}

void interpreter::interpret (parameters &params) {
   TRACE ('i', params);
   string command = shift (params);
   interpreterfn function = interpmap[command];
   if (function == NULL) throw runtime_error ("syntax error");
   (this->*function) (params);
}

void interpreter::do_define (parameters &params) {
   TRACE ('i', params);
   string name = shift (params);
   objmap[name] = make_object (params);
}

void interpreter::do_draw (parameters &params) {
   TRACE ('i', params);
   string name = shift (params);
   object *thing = objmap[name];
   if (thing == NULL) throw runtime_error (name + ": no such object");
   degrees angle = degrees (0);
   if (params.size() == 3) {
      angle = degrees (from_string<double> (params.back()));
      params.pop_back();
   }
   if (params.size() != 2) throw runtime_error ("syntax error");
   xycoords coords (inches (from_string<double> (params.front())),
                    inches (from_string<double> (params.back())));
   thing->draw (outfile, coords, angle);
}

void interpreter::do_newpage (parameters &params) {
   if (params.size() != 0) throw runtime_error ("syntax error");
   endpage ();
   ++pagenr;
   startpage ();
}

void interpreter::prolog () {
   outfile << "%!PS-Adobe-3.0" << endl;
   outfile << "%%Creator: " << sys_info::get_execname () << endl;
   outfile << "%%CreationDate: " << datestring() << endl;
   outfile << "%%PageOrder: Ascend" << endl;
   outfile << "%%Orientation: Portrait" << endl;
   outfile << "%%SourceFile: " << infilename << endl;
   outfile << "%%EndComments" << endl;
}

void interpreter::startpage () {
   outfile << endl;
   outfile << "%%Page: " << pagenr << " " << pagenr << endl;
   outfile << page_xoffset << " " << page_yoffset
           << " translate" << endl;
   outfile << "/Courier findfont 10 scalefont setfont" << endl;
   outfile << "0 0 moveto (" << infilename << ":"
           << pagenr << ") show" << endl;

}

void interpreter::endpage () {
   outfile << endl;
   outfile << "showpage" << endl;
   outfile << "grestoreall" << endl;
}

void interpreter::epilog () {
   outfile << endl;
   outfile << "%%Trailer" << endl;
   outfile << "%%Pages: " << pagenr << endl;
   outfile << "%%EOF" << endl;

}

object *interpreter::make_object (parameters &command) {
   TRACE ('f', command);
   string type = shift (command);
   factoryfn func = factorymap[type];
   if (func == NULL) throw runtime_error (type + ": no such object");
   return (this->*func) (command);
}

```

```
object *interpreter::make_text (parameters &command) {
   TRACE ('f', command);
   return new text ("", points(0), string());
}

object *interpreter::make_ellipse (parameters &command) {
   TRACE ('f', command);
   return new ellipse (inches(0), inches(0), points(0));
}

object *interpreter::make_circle (parameters &command) {
   TRACE ('f', command);
   return new circle (inches(0), points(0));
}

object *interpreter::make_polygon (parameters &command) {
   TRACE ('f', command);
   return new polygon (coordlist(), points(0));
}

object *interpreter::make_rectangle (parameters &command) {
   TRACE ('f', command);
   return new rectangle (inches(0), inches(0), points(0));
}

object *interpreter::make_square (parameters &command) {
   TRACE ('f', command);
   return new square (inches(0), points(0));
}

object *interpreter::make_line (parameters &command) {
   TRACE ('f', command);
   return new line (inches(0), points(0));
}

RCSC(__interp_cc__,
"$Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $")

```

```

#include <cstdlib>

using namespace std;

#include "numbers.h"
#include "util.h"

ostream &operator<< (ostream &out, const degrees &that) {
   out << that.angle << "deg";
   return out;
}

ostream &operator<< (ostream &out, const points &that) {
   out << that.pointvalue << "pt";
   return out;
}

ostream &operator<< (ostream &out, const inches &that) {
   out << that.pointvalue / PTS_PER_INCH << "in";
   return out;
}

ostream &operator<< (ostream &out, const xycoords &coords) {
   out << "(" << coords.first << "," << coords.second << ")";
   return out;
}

RCSC(__numbers_cc__,
"$Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $")
```

```

#include <typeinfo>

using namespace std;

#include "object.h"
#include "util.h"

#define WHOAMI \
        "[" << typeid(*this).name() << "@" << (void *) this << "]"

#define CTRACE(ARGS) \
        TRACE ('c', WHOAMI << " " << ARGS)

#define DTRACE(ARGS) \
        TRACE ('d', WHOAMI << " coords=" << coords \
               << " angle=" << angle << endl << ARGS);

object::~object () {
   CTRACE ("delete");
}

text::text (const string &font, const points &size, const string &data):
      fontname(font), fontsize(size), textdata(data) {
   CTRACE ("font=" << fontname << " size=" << fontsize
           << " \"" << textdata << "\"")
}

ellipse::ellipse (const inches &initheight, const inches &initwidth,
                  const points &initthick):
      shape(initthick), height(initheight), width(initwidth) {
   CTRACE ("height=" << height << " width=" << width
           << " thick=" << thick);
}

circle::circle (const inches &diameter, const points &thick):
      ellipse (diameter, diameter, thick) {
}

polygon::polygon (const coordlist &coords, const points &initthick):
      shape(initthick), coordinates(coords) {
   CTRACE ( "thick=" << thick << " coords=" << endl
            << coordinates);
}

rectangle::rectangle (const inches &height, const inches &width,
                      const points &initthick):
      polygon (make_list (height, width), initthick) {
}

square::square (const inches &width, const points &thick):
      rectangle (width, width, thick) {
}

line::line (const inches &length, const points &initthick):
      polygon (make_list (length), initthick) {

}

void text::draw (ostream &out, const xycoords &coords,
                 const degrees &angle) {
   DTRACE ("font=" << fontname << " size=" << fontsize
           << " \"" << textdata << "\"")
}
```

```

void ellipse::draw (ostream &out, const xycoords &coords,
                    const degrees &angle) {
   DTRACE ("height=" << height << " width=" << width
           << " thick=" << thick);
}

void polygon::draw (ostream &out, const xycoords &coords,
                    const degrees &angle) {
   DTRACE ( "thick=" << thick << " coords=" << endl
            << coordinates);
}

coordlist rectangle::make_list (
            const inches &height, const inches &width) {
   coordlist coordlist;
   return coordlist;
}

coordlist line::make_list (const inches &length) {
   coordlist coordlist;
   return coordlist;
}


RCSC(__object_cc__,
"$Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $")
```

```

#include <climits>
#include <iostream>
#include <limits>
#include <vector>

using namespace std;

#include "trace.h"

//
// ** BUG IN STL ** BUG IN STL **
// We should use vector<bool> instead of vector<char>,
// but vector<bool> has a bug:
// http://forums.sun.com/thread.jspa?threadID=5277939
// Static linking works, but doubles the size of the executable
// image.
// ** BUG IN STL ** BUG IN STL **
//

typedef vector<char> boolvec;
boolvec traceflags::flags (UCHAR_MAX + 1, false);
const boolvec trueflags (UCHAR_MAX + 1, true);

void traceflags::setflags (const string &optflags) {
   string::const_iterator itor = optflags.begin();
   string::const_iterator end = optflags.end();
   for (; itor != end; ++itor) {
      if (*itor == '@') {
         flags = trueflags;
      }else {
         flags[*itor] = true;
      }
   }
   // Note that TRACE can trace setflags.
   TRACE ('t',  "optflags = " << optflags);
}

//
// getflag -
//    Check to see if a certain flag is on.
//

bool traceflags::getflag (char flag) {
   // Bug alert:
   // Don't TRACE this function or the stack will blow up.
   bool result = flags[flag];
   return result;
}

RCSC(__trace_cc__,
"$Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $")

```

```

#include <cerrno>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <sstream>
#include <stdexcept>
#include <string>
#include <typeinfo>

using namespace std;

#include "util.h"

int sys_info::exit_status = EXIT_SUCCESS;
string *sys_info::execname = NULL; // Must be initialized from main().

void sys_info_error (const string &condition) {
   throw logic_error ("main() has " + condition
               + " called sys_info::set_execname()");
}

void sys_info::set_execname (const string &argv0) {
   if (execname != NULL) sys_info_error ("already");
   int slashpos = argv0.find_last_of ('/') + 1;
   execname = new string (argv0.substr (slashpos));
   cout << boolalpha;
   cerr << boolalpha;
   TRACE ('u', "execname = " << execname);
}

const string &sys_info::get_execname () {
   if (execname == NULL) sys_info_error ("not yet");
   return *execname;
}

void sys_info::set_exit_status (int status) {
   if (execname == NULL) sys_info_error ("not yet");
   exit_status = status;
}

int sys_info::get_exit_status () {
   if (execname == NULL) sys_info_error ("not yet");
   return exit_status;
}

const string datestring () {
   time_t clock = time (NULL);
   struct tm *tm_ptr = localtime (&clock);
   char timebuf[128];
   size_t bufsize = strftime (timebuf, sizeof timebuf,
          "%a %b %e %H:%M:%S %Z %Y", tm_ptr);
   return timebuf;
}

list<string> split (const string &line, const string &delimiters) {
   list<string> words;
   int end = 0;
   // Loop over the string, splitting out words, and for each word
   // thus found, append it to the output list<string>.
   for (;;) {
      int start = line.find_first_not_of (delimiters, end);
      if (start == string::npos) break;
      end = line.find_first_of (delimiters, start);
```

```
         words.push_back (line.substr (start, end - start));
      }
      TRACE ('u', words);
      return words;
   }

   ostream &complain() {
      sys_info::set_exit_status (EXIT_FAILURE);
      cerr << sys_info::get_execname () << ": ";
      return cerr;
   }

   void syscall_error (const string &object) {
      complain() << object << ": " << strerror (errno) << endl;
   }

   template <typename item_t>
   ostream &operator<< (ostream &out, const list<item_t> &vec) {
      typename list<item_t>::const_iterator itor = vec.begin();
      typename list<item_t>::const_iterator end = vec.end();
      // If the list is empty, do nothing.
      if (itor != end) {
         // Print out the first element without a space.
         out << *itor++;
         // Print out the rest of the elements each preceded by a space.
         while (itor != end) out << " " << *itor++;
      }
      return out;
   }

   template <typename type>
   string to_string (const type &that) {
      ostringstream stream;
      stream << that;
      return stream.str ();
   }

   template <typename type>
   type from_string (const string &that) {
      stringstream stream;
      stream << that;
      type result;
      if ( !(stream >> result   // Can we read type from string?
           && stream >> std::ws  // Flush trailing white space.
           && stream.eof ())     // Must now be at end of stream.
      ) {
         throw domain_error (string (typeid (type).name ())
               + " from_string (" + that + ")");
      }
      return result;
   }

   #include "templates.h"

   RCSC(__util_cc__,
   "$Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $")
```

```
# $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} :
        @ touch ${DEPSFILE}
        ${GMAKE} deps

idents : ${ALLSOURCES} ${OBJECTS} ${EXECBIN}
        ldd ${EXECBIN}
        ident ${ALLSOURCES} ${OBJECTS} ${EXECBIN}

again :
        ${GMAKE} spotless deps ci all lis

ifeq (${NEEDINCL}, )
include ${DEPSFILE}
endif

```

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

```
# Makefile.deps created Mon Feb  8 13:45:19 PST 2010
interp.o : interp.cc
interp.o : interp.h
interp.o : numbers.h
interp.o : object.h
interp.o : trace.h
interp.o : util.h
main.o : interp.h
main.o : main.cc
main.o : numbers.h
main.o : object.h
main.o : trace.h
main.o : util.h
numbers.o : numbers.cc
numbers.o : numbers.h
numbers.o : trace.h
numbers.o : util.h
object.o : numbers.h
object.o : object.cc
object.o : object.h
object.o : trace.h
object.o : util.h
trace.o : trace.cc
trace.o : trace.h
util.o : numbers.h
util.o : templates.h
util.o : trace.h
util.o : util.cc
util.o : util.h
```

```
ldd draw
        libCstd.so.1 =>  /usr/lib/libCstd.so.1
        libCrun.so.1 =>  /usr/lib/libCrun.so.1
        libm.so.2 =>     /lib/libm.so.2
        libc.so.1 =>     /lib/libc.so.1
ident interp.h numbers.h object.h trace.h util.h templates.h main.cc interp.cc numbers.cc object.cc trace.cc util.cc Makefile README main.o interp.o numbers.o object.o trace.o util.o draw
interp.h:
     $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $

numbers.h:
     $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $

object.h:
     $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $

trace.h:
     $Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $
     $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $

util.h:
     $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $

templates.h:
     $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $

main.cc:
     $Id: main.cc,v 1.2 2010-02-04 19:35:22-08 - - $

interp.cc:
     $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $

numbers.cc:
     $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $

object.cc:
     $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $

trace.cc:
     $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $

util.cc:
     $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $

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

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

main.o:
     $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
     $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
     $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
     $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
     $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $

interp.o:
     $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
     $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
     $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
     $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
     $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
```

```
        $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $
        $Compiled: interp.cc Feb  8 2010 13:45:21 $

numbers.o:
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $
        $Compiled: numbers.cc Feb  8 2010 13:45:22 $

object.o:
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $
        $Compiled: object.cc Feb  8 2010 13:45:22 $

trace.o:
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $
        $Compiled: trace.cc Feb  8 2010 13:45:23 $

util.o:
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $
        $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $
        $Compiled: util.cc Feb  8 2010 13:45:23 $

draw:
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
        $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
        $Id: interp.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: interp.cc,v 1.3 2010-02-05 14:09:12-08 - - $
        $Compiled: interp.cc Feb  8 2010 13:45:21 $
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: numbers.cc,v 1.2 2010-02-05 14:09:12-08 - - $
        $Compiled: numbers.cc Feb  8 2010 13:45:22 $
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: object.h,v 1.4 2010-02-04 19:09:00-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: object.cc,v 1.1 2010-01-29 18:07:32-08 - - $
        $Compiled: object.cc Feb  8 2010 13:45:22 $
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: trace.cc,v 1.2 2010-02-02 18:23:29-08 - - $
        $Compiled: trace.cc Feb  8 2010 13:45:23 $
        $Id: trace.h,v 1.1 2010-01-29 18:07:32-08 - - $
        $Id: util.h,v 1.4 2010-02-05 14:09:12-08 - - $
        $Id: numbers.h,v 1.3 2010-02-05 14:09:12-08 - - $
        $Id: templates.h,v 1.3 2010-02-05 17:20:52-08 - - $
        $Id: util.cc,v 1.9 2010-02-05 17:20:52-08 - - $
        $Compiled: util.cc Feb  8 2010 13:45:23 $
```

```perl
#!/usr/bin/perl
# $Id: draw.perl,v 1.2 2010-01-29 18:15:03-08 - - $

use strict;
use warnings;
use POSIX qw (strftime);
use Data::Dumper;

my $script = $0;
my $date = strftime "%a %b %e %H:%M:%S %Z %Y", localtime;
my $debug = 0;

$0 =˜ s|^(.*/)?([^/]+)/*$|$2|;
my $exit_status = 0;
END{ exit $exit_status; }
sub note(@){ print STDERR "$0: @_"; };
$SIG{'__WARN__'} = sub{ note @_;; $exit_status = 1; };
$SIG{'__DIE__'} = sub{ warn @_; exit; };

sub startpage ($) {
   my ($state) = @_;
   my $outfile = $state->{OUTFILE};
   print $outfile
      "%%Page: $state->{PAGENR} $state->{PAGENR}\n",
      "18 18 translate\n",
      "/Courier findfont 10 scalefont setfont\n",
      "0 0 moveto ($state->{INFILENAME}:$state->{PAGENR}) show\n";
}

sub endpage ($) {
   my ($state) = @_;
   my $outfile = $state->{OUTFILE};
   print $outfile
      "showpage\n",
      "grestoreall\n";
}

sub prolog ($) {
   my ($state) = @_;
   my $outfile = $state->{OUTFILE};
   print $outfile
      "%!PS-Adobe-3.0\n",
      "%%Creator: $script\n",
      "%%CreationDate: $date\n",
      "%%PageOrder: Ascend\n",
      "%%Orientation: Portrait\n",
      "%%SourceFile: $state->{INFILENAME}\n",
      "%%EndComments\n";
   $state->{PAGENR} = 1;
   startpage $state;
}

sub epilog ($) {
   my ($state) = @_;
   my $outfile = $state->{OUTFILE};
   endpage $state;
   print $outfile
      "%%Trailer\n",
      "%%Pages: $state->{PAGENR}\n",
      "%%EOF\n",
}

sub error ($;$) {
   my ($state, $message) = @_;
   $message = "syntax error" unless $message;
   warn "$state->{INFILENAME}: $.: $message\n";
}

sub numeric ($) {
   my ($number) = @_;
   return $number =~ m/^[+-]?(\d+\.?\d*|\.\d+)([Ee][+-]?\d+)?$/;
}

sub numbers ($$) {
   my ($count, $numbers) = @_;
   return 0 if $count != @$numbers and $count > 0;
   return ! grep {! numeric $_} @$numbers;
}

sub make_text ($$) {
   my ($state, $words) = @_;
   my $size = (numeric $words->[0]) ? (shift @$words) : 12;
   my $font = shift @$words;
   do {error $state; return} unless @$words >= 1;
   my $text = join ' ', @$words;
   $text =~ s/[(\\)]/\\$&/g;
   my $outfile = $state->{OUTFILE};
   return sub {
      my ($place) = @_;
      my ($x0, $y0, $angle) = @$place;
      map {$_ *= 72} $x0, $y0;
      $angle = 0 unless $angle;
      print $outfile
            "gsave\n",
            "   /$font findfont\n",
            "   $size scalefont setfont\n",
            "   $x0 $y0 translate\n",
            "   $angle rotate\n",
            "   0 0 moveto\n",
            "   ($text)\n",
            "   show\n",
            "grestore\n";
   }
}

sub make_ellipse ($$) {
   my ($state, $words) = @_;
   push @$words, 2 unless @$words == 3;
   do {error $state; return} unless numbers 3, $words;
   my ($height, $width, $thick) = @$words;
   map {$_ *= 72} $height, $width;
   do {error $state, "syntax error height"; return} if $height == 0;
   my ($xscale, $yscale, $radius);
   if ($height < $width) {
      $xscale = 1;
      $yscale = $height / $width;
      $radius = $width / 2;
   }else {
      $xscale = $width / $height;
      $yscale = 1;
      $radius = $height / 2;
   }
   my $outfile = $state->{OUTFILE};
   return sub {
      my ($place) = @_;
      my ($x0, $y0, $angle) = @$place;
      map {$_ *= 72} $x0, $y0, $width;
      $angle = 0 unless $angle;
      print $outfile
            "gsave\n",
            "   newpath\n",
            "   /save matrix currentmatrix def\n",
            "   $x0 $y0 translate\n",
            "   $angle rotate\n",
            "   $xscale $yscale scale\n",
            "   0 0 $radius 0 360 arc\n",
            "   save setmatrix\n",
            "   $thick setlinewidth\n",
            "   stroke\n",
            "grestore\n";
   }
}

sub make_circle ($$) {
   my ($state, $words) = @_;
   unshift @$words, $words->[0];
   return make_ellipse $state, $words;
}

sub make_polygon ($$) {
   my ($state, $words) = @_;
   do {error $state; return} unless @$words >= 2 and numbers 0, $words;
   my $thick = (@$words % 2 == 0) ? 2 : pop @$words;
   my $outfile = $state->{OUTFILE};
   return sub {
      my ($place) = @_;
      my ($x0, $y0, $angle) = @$place;
      map {$_ *= 72} $x0, $y0;
      $angle = 0 unless $angle;
      print $outfile
            "gsave\n",
            "   newpath\n",
            "   $x0 $y0 translate\n",
            "   $angle rotate\n",
            "   0 0 moveto\n";
      for (my $gon = 0; $gon < @$words; $gon += 2) {
         my ($xrel, $yrel) = @{$words}[$gon, $gon + 1];
         map {$_ *= 72} $xrel, $yrel;
         print $outfile
            "   $xrel $yrel rlineto\n";
      }
      print $outfile
            "   closepath\n",
            "   $thick setlinewidth\n",
            "   stroke\n",
            "grestore\n";
   }
}

sub make_rectangle ($$) {
   my ($state, $words) = @_;
   push @$words, 2 unless @$words == 3;
   do {error $state; return} unless numbers 3, $words;
   my ($hght, $wid, $thick) = @$words;
   return make_polygon $state, [0, $hght, $wid, 0, 0, - $hght, $thick];
}

sub make_square ($$) {
   my ($state, $words) = @_;
   unshift @$words, $words->[0];
   return make_rectangle $state, $words;
}
```

```

sub make_line ($$) {
   my ($state, $words) = @_;
   push @$words, 2 unless @$words == 2;
   do {error $state; return} unless numbers 2, $words;
   my ($length, $thick) = @$words;
   $thick = 2 unless $thick;
   return make_polygon $state, [$words->[0], 0, $thick];
}

my %objects = (
   "text" => \&make_text,
   "ellipse" => \&make_ellipse,
   "circle" => \&make_circle,
   "polygon" => \&make_polygon,
   "rectangle" => \&make_rectangle,
   "square" => \&make_square,
   "line" => \&make_line,
);

sub do_define ($$) {
   my ($state, $words) = @_;
   do {error $state; return} unless @$words > 2;
   my ($name) = shift @$words;
   my ($class) = shift @$words;
   my ($maker) = $objects{$class};
   do {error $state, "no such shape"; return} unless $maker;
   my $object = $maker->($state, $words);
   $state->{SYMTAB}->{$name} = $object if $object;
}

sub do_draw ($$) {
   my ($state, $words) = @_;
   my $object = $state->{SYMTAB}->{shift @$words};
   do {error $state, "no such object"; return} unless $object;
   push @$words, 0 if @$words == 2;
   do {error $state; return} unless numbers 3, $words;
   $object->($words);
}

sub do_newpage ($$) {
   my ($state, $words) = @_;
   do {error $state; return} unless @$words == 0;
   endpage $state;
   $state->{PAGENR}++;
   startpage $state;
}

my %commands = (
   "define" => \&do_define,
   "draw" => \&do_draw,
   "newpage" => \&do_newpage,
);

sub parsefile ($) {
   my ($state) = @_;
   prolog $state;
   my $infile = $state->{INFILE};
   my $outfile = $state->{OUTFILE};
   while (defined (my $line = <$infile>)) {
      for (;;) {
         chomp $line;
         last unless $line =~ s/\\$//;
         my $contin = <$infile>;
```

```
            $line .= $contin;
         }
         chomp $line;
         print $outfile "\n",
               "%%Command[$.]: $line\n";
         next if $line =~ m/^\s*(#|$)/;
         my @words = split " ", $line;
         my $command = $commands{shift @words};
         do {error $state; next} unless defined $command;
         $command->($state, \@words);
      }
      epilog $state;
}

sub main () {
   @ARGV and $ARGV[0] =~ m/^-D/ and $debug = shift;
   unless (@ARGV) {
      parsefile {INFILENAME => "-",
                 INFILE => *STDIN,
                 OUTFILE => *STDOUT};
   }else {
      for my $infilename (@ARGV) {
         my $outfilename = $infilename;
         $outfilename =~ s|^.*/([^/]+)/*$|$1|;
         $outfilename =~ s/(\.dr)?$/.ps/;
         open my $infile, "<$infilename"
              or warn "$infilename: $!\n" and next;
         open my $outfile, ">$outfilename"
              or warn "$outfilename: $!\n" and next;
         print "$0: $infilename => $outfilename\n";
         parsefile {INFILENAME => $infilename,
                    INFILE => $infile,
                    OUTFILE => $outfile};
         close $infile;
         close $outfile;
      }
   }
}

main;

```