Appendix A

Software resources for this book

(This appendix is a considerably modified version of the one that appeared in the printed book. For interest, the original one can be found in the file appa0.pdf.)

A.1 Source code for the programs

The software that accompanies this text was originally developed in Modula-2. It was subsequently converted to Turbo Pascal, and to C++. Although C++ code is used for most of the illustrations in the text, highly self-consistent source code in all three languages is to be found in this distribution, along with language-specific implementation notes.

The C++ source code was mainly developed under MS-DOS using Borland C++ 3.1. It has also been successfully compiled under Linux, using G++, the GNU C compiler. Although many of the case studies have also been tested with Turbo C++ 3.1, there appears to be a bug in that compiler that prevents the template set class from compiling correctly.

The Turbo Pascal source code was developed to run on any version of Turbo Pascal from 5.5 onwards (although Delphi users should read the notes on Delphi below). However, it makes little use of OOP extensions.

The Modula-2 source code should be immediately usable on MS-DOS based systems using the shareware compiler marketed by Fitted Software Tools (FST), the Stony Brook Modula-2 compiler marketed by Gogesch Micro Systems, Inc., or the TopSpeed Modula-2 compilers developed by Jensen and Partners International (JPI). Provided the appropriate version of the I/O module `FileIO` is used, it should also compile unchanged on XDS and Stony Brook ISO-compliant compilers, and under Gardens Point Modula-2 on a wide range of systems.

The software in the distribution is currently supplied in the form of compressed, self-extracting MS-DOS executable files. There are eight of these files.

```
cocorc.exe         71 Kb  Minimal Coco/R for C          (Updated November 1999)
cocorm.exe         76 Kb  Minimal Coco/R for Modula-2   (Updated November 1999)
cocorp.exe         65 Kb  Minimal Coco/R for Pascal     (Updated November 1999)

common.exe         93 Kb  Files common to all languages (Updated November 1999)
csources.exe      350 Kb  C++ specific sources          (Updated November 1999)
msources.exe      345 Kb  Modula-2 specific sources     (Updated November 1999)
psources.exe      312 Kb  Pascal specific sources       (Updated November 1999)

fileio.exe        153 Kb  Modula-2 FileIO module        (Updated November 1999)

lha213.exe         43 Kb  LHArc decompressor
readme.1st         11 Kb  Installation and setup instructions
```

A.2 Unpacking the software

To unpack the software, simply follow the steps below. Example MS-DOS commands are shown (these may need alteration, depending on the configuration of your computer). Windows users may

follow an equivalent sequence of operations from within the File Manager or Explorer. The files may also be unpacked directly on other systems using appropriate ports of `LHA`.

- Make a backup copy of the software and keep the original in a safe place.
- Proceed to unpack (a) the language specific case studies (b) the language independent files (c) the language specific compiler generator and, if you are a Modula-2 user, (d) the `FileIO` library.

Unpacking the case study software

- Create a directory to act as the root directory for storing your sources, for example

  ```
  MKDIR C:\SRCES
  ```

  (`C:\SRCES` is used as an example only - please yourself in this regard.)

- Log onto this as the working directory

  ```
  CD C:\SRCES
  ```

- Unpack the chosen source file to the `C:\SRCES` directory. For example, for the C++ sources, execute

  ```
  X:\DOWNLOAD\CSOURCES.EXE
  ```

  where `X:\DOWNLOAD\` is the path to the directory in which the downloaded distribution resides.

- You must also unpack the language independent files to the same directory by executing

  ```
  X:\DOWNLOAD\COMMON.EXE
  ```

  These steps will create a small directory hierarchy under the `C:\SRCES` directory, in which various subdirectories will appear, usually one for each chapter. For example, you will find the source code for the programs in chapter 10 in the directory `C:\SRCES\CHAP10\CPP` (for the C++ versions) or the directory `C:\SRCES\CHAP10\MODULA` (for the Modula-2 versions).

IMPORTANT

- Once you have unpacked the system, read the file `C:\SRCES\README.SRC` for further details on how the directories are laid out.
- You may unpack all three of the language specific sources into the same directory tree if you wish - the C++, Modula-2 and Pascal sources are stored in separate directories under the directory for each chapter.
- If you are using Borland C++ you may need to edit the configuration file `C:\SRCES\COMMON\TURBOC.CFG` to reflect the correct paths and options for your compiler.

Unpacking the compiler generator Coco/R

- Create a directory to act as the root directory for your chosen version of Coco/R, for example

  ```
  MKDIR C:\COCO
  ```

- Log onto this as the working directory

```
CD C:\COCO
```

- Unpack the chosen Coco/R system to the `C:\COCO` directory. For example, for the C++ version, execute

```
X:\DOWNLOAD\COCORC.EXE
```

  This will create a small directory hierarchy under the `C:\COCO\` directory, containing the various components of Coco/R.

IMPORTANT

- Once you have unpacked the system, read the file `C:\COCO\README.1ST` for further details on how the directories are laid out, and how to complete the installation of Coco/R so that it can be executed easily.
- If you wish to install more than one version of Coco/R, please do so into separate directories, for example `C:\COCOC\` for the C++ version, `C:\COCOM\` for the Modula-2 version, `C:\COCOP\` for the Pascal version.
- If you are using Borland C++ you may need to edit the configuration file `C:\SRCES\COMMON\TURBOC.CFG` to reflect the correct paths and options for your compiler.

A.3 The FileIO library (Modula-2 users only)

When this book was first published, standardized I/O for Modula-2 was not yet widely available (and was incompatible with extant Modula-2 compilers). The Modula-2 source code in this distribution attempts to get around this problem by providing (another!) I/O module, called `FileIO`. The definition module for `FileIO` is acceptable to all compilers tested so far; implementations have been supplied for each that differ internally only in a few places.

`FileIO` provides the usual services for opening and closing text files, and for reading and writing strings, words, whole numbers and line marks to such files. It can also handle random access binary files, as block read and write operations are provided. In addition there are some utility procedures, for obtaining command line parameters and environment strings, and for the output of dates and times. The module is of fairly widespread applicability beyond the confines of this text, and is compatible with the modules generated by Coco/R (which assumes the module to be available). As an example of a library module it is really rather too large, but has been developed in this way to minimize the number of non-portable sections and modules needed for implementing the programs in the book.

Modula-2 scanners, parsers, and compilers created by Coco/R assume that you will use the I/O module `FileIO`. If you are a Modula-2 user, you will thus need to install and compile the version of `FileIO` that matches your compiler.

Among the resources you will find a self-extracting file `FILEIO.EXE` that contains the sources of `FileIO` for a variety of Modula-2 compilers.

- Make a directory to contain these sources

```
MKDIR C:\FILEIO
```

- Log onto this as the working directory

```
CD C:\FILEIO
```

- Unpack the `FILEIO.EXE` file to the `C:\FILEIO` directory

```
X:\DOWNLOAD\FILEIO.EXE
```

  This will create a small directory hierarchy under the `C:\FILEIO` directory, in which various subdirectories will appear, one for each compiler.

  In the `C:\FILEIO` directory you will find a definition module `FILEIO.DEF`, and in the subdirectories of `C:\FILEIO` you will find various compiler specific modules, including `FILEIO.MOD`. You will need to proceed as follows, on the assumption that you have a "working" directory `C:\WORK` in which you normally develop programs. (You may, of course, find it preferable to install `FileIO` in the library directory or directories for your Modula-2 compiler.)

```
CD C:\WORK
COPY C:\FILEIO\FILEIO.DEF
COPY C:\FILEIO\xxx

where xxx =
    JPI   (TopSpeed compilers)
    FST   (Fitted Systems Tools compilers)
    LOG   (Logitech compilers)
    STO   (StonyBrook compilers)
    GPMPC (Gardens Point PC compiler)
    etc
```

  Follow this by compiling `FILEIO.DEF` and `FILEIO.MOD`.

The sources supplied will act as models of implementations for compilers not mentioned above. In case of difficulty, please contact the author.

A.4 Setting up the software for a particular case study

Installation of many (but not all) of the case studies requires you simply to

- Log onto to the appropriate source drive and subdirectory, for example

```
C:
CD C:\SRCES\CHAP15\CPP
```

- Issue the command

```
SETUP
```

  with no parameters. This will indicate what parameters might be added to the command to provide a variant appropriate for the case studies of that chapter.

- For example, follow this by obeying a `SETUP` command like

```
SETUP C:\WORK 5
```

  where `C:\WORK` is a directory that will be created if necessary, and into which the appropriate

components for SETUP option 5 will be copied.

- Log onto the working directory

```
CD C:\WORK
```

- Issue the MAKE command (C++), or equivalent Coco/R and compiler commands (Pascal or Modula-2) to compile the selected case study.

IMPORTANT

- The "makefiles" supplied are intended for use with Borland C++ release 3.1; they may need slight alteration. The configuration file TURBOC.CFG may also need alteration.
- It is crucial that you are logged onto an appropriate directory before you issue the SETUP command. If this is not the case, you might execute some other SETUP command and become confused!
- Some of the simpler case studies - typically those consisting of only one source file - are not managed by a SETUP process. Consult the file C:\SRCES\README.SRC for further details.

The Turbo Pascal case studies, and the output from Coco/R for Pascal will not compile directly under Delphi. They require WinCrt to be added to the list of units "used" (edit the compiler frame files for Coco/R). Furthermore those systems (like the assemblers and compilers) that provide for interactive input/output in their interpreters require use of WinCrt.AssignCrt instead of the current calls to Assign with an empty filename (for example in STKMC.PAS and MC.PAS).

A.5 Coco/R distributions

The compiler generator Coco/R used in this book was originally developed in Oberon by Hanspeter Mössenböck, who also did a port to Modula-2 for the Apple MacMeth system. A further port was done to TopSpeed Modula-2 by Marc Brandis and Christof Brass. This was refined and extended by the author in conjunction with John Gough and Hanspeter Mössenböck, to the point where a single version runs on most Modula-2 compilers available under MS-DOS, as well as the Mocka and Gardens Point compilers available for Unix (and other) systems, including Linux and Free BSD.

A port of Coco/R to Turbo Pascal was done by the author in conjunction with Volker Pohlers.

Coco/R was ported to C by Francisco Arzu, yielding a version that can generate either C or C++ compilers.

Coco/R has also been also ported to Java by Hanspeter Mössenböck.

For details of how to obtain the latest versions of the complete Coco/R distributions for a variety of languages and operating systems visit the Coco/R home page at
http://cs.ru.ac.za/homes/cspt/cocor.htm.

The original report on Coco/R (Mössenböck, 1990a) can be obtained from

ftp://ftp.ssw.uni-linz.ac.at/pub/Papers/Coco.Report.ps.Z
ftp://cs.ru.ac.za/pub/coco/Coco.Report.ps.Z

Professor Mössenböck may be contacted at the address below

Prof. Hanspeter Mössenböck
Institute of Computer Science
University of Linz
Alternbergerstr 69,
A-4040 Linz, Austria
Tel: +43-732-2468-9700
e-mail: moessenboeck@ssw.uni-linz.ac.at

A.6 Other compiler tools, free compilers and similar resources

The subject of compiler writing and the development of compilers is one of the classical fields of Computer Science for which there are now many freely available resources. The interested reader might do worse than to explore some of the following leads:

FAQ's, topics and archives from the Comp.Compilers news group
http://www.iecc.com/compilers

The German National Resource Centre catalogue of compiler construction tools
http://www.first.gmd.de/cogent/catalog

Idiom Consulting's catalogue of free compilers
http://www.idiom.com/free-compilers

Compiler Connection
http://www.compilerconnection.com

Burks (Brighton University Resource Kit for Students)
http://burks.bton.ac.uk/burks

Dragon Fodder
http://www.km-cd.com/dragon_fodder

Jack Crenshaw's "Let's build a compiler"
http://www.iecc.com/compilers/crenshaw/

The PCCTS compiler construction kit mentioned in Chapter 10 is available from

http://www.ANTLR.org/pccts133.html

Freely available early versions of the Cocktail compiler construction tools mentioned in Chapter 10 may be obtained by anonymous ftp from

ftp://ftp.ira.uka.de/pub/programming/cocktail
ftp://144ftp.info.uni-karlsruhe.de/pub/cocktail

For the commercial version and support visit http://www.first.gmd.de/cocktail/ or contact Josef Grosch.

`mtc`, the Modula-2 to C translator program mentioned in Chapter 2 is available by anonymous ftp

from

ftp://ftp.psg.com:/pub/modula-2/grosch/mtc.tar.Z
ftp://ftp.gmd.de/gmd/cocktail/mtc.tar.Z

`p2c`, the Pascal to C translator program mentioned in Chapter 2, and `cperf`, the perfect hash function generator mentioned in Chapter 14, are available by anonymous ftp from any of the sites that mirror the Free Software Foundation GNU archives. The primary server for these archives is at ftp://prep.ai.mit.edu. Among many others, the Linux sites, such as those at ftp://tsx-11.mit.edu or ftp://sunsite.unc.edu and ftp://src.doc.ic.ac.uk also carry copies of the GNU archives.

Also to be found among these archives is `gcc`, the well-known GNU C compiler, which can also be found at ftp://gcc.gnu.org or at one of many mirror sites

As another example of a freely available C compiler, `lcc` is a retargetable compiler for ANSI C described in "A Retargetable C Compiler: Design and Implementation" by Fraser, Hanson and Hansen (Benjamin/Cummings, 1995). `lcc` is available, along with documentation and a sample chapter of the book at

http://www.CS.Princeton.EDU/software/lcc/

Versions of the Gardens Point Modula-2 compiler for DOS, Linux and FreeBSD are available from

ftp://ftp.fit.qut.edu.au/pub/gpm_modula2
ftp://ftp.psg.com/pub/modula-2/gpm

Versions of the Mocka Modula-2 compiler for Linux and FreeBSD are available from

http://www.first.gmd.de/mocka/

The shareware FST Modula-2 compiler for MS-DOS systems is available by anonymous ftp from

ftp://ftp.psg.com/pub/modula-2/fst/fst-40s.lzh
ftp://cs.ru.ac.za/pub/languages/fst-40s.lzh

The self-extracting files in this distribution were compressed and packed using the freely available program `LHA.EXE` developed by Haruyasu Yoshizaki. In terms of the distribution agreement for this program, the complete package for LHA.EXE is itself supplied as a self-extracting executable, LHA213.EXE. You are quite welcome to unpack this file, although it is not needed for the operations described above.

Further to comply with the distribution agreement, the copyright notice for this package is given below

```
4. Our distribution Policy

   This software, this document and LHA.EXE, is a copyright-reserved free
   program.  You may use, copy and distribute this software free of charge
   under the following conditions.

1. Never change Copyright statement.

2. The enclosed documents must be distributed with as a package.
```

3. When you have changed the program, or implemented the program for other OS or environment, then you must specify the part you have changed. Also make a clear statement as to your name and MAIL address or phone number.

4. The author is not liable for any damage on your side caused by the use of this program.

5. The author has no duty to remedy for the deficiencies of the program.

6. When you are to distribute this software with publications or with your product, you have to print the copyright statement somewhere on the disk or on the package. You cannot distribute this software with copyprotected products.

Disclaimer

While every attempt has been made to ensure that the software in this distribution performs properly, the author can accept no liability for any damage or loss, including special, incidental, or consequential, caused by the use of the software, directly or indirectly.

However, please bring any problems that you may experience to the attention of the author:

Pat Terry
Computer Science Department
Rhodes University
GRAHAMSTOWN 6140
South Africa
Tel: +27-46-622-8292
FAX: +27-46-636-1915
e-mail: cspt@cs.ru.ac.za
e-mail: p.terry@ru.ac.za

Trademarks

Products and services that are referred to in this book may be either trademarks and/or registered trademarks of their respective owners. The author makes no claim to these trademarks.

Borland C++, Turbo C++, Delphi and Turbo Pascal are trademarks of Borland International Corporation.

GNU C Compiler is a trademark of the Free Software Foundation.

IBM and IBM PC are trademarks of International Business Machines Corporation.

Microsoft, MS and MS-DOS are registered trademarks, and Windows is a trademark of Microsoft Corporation.

Stony Brook Software and QuickMod are trademarks of Gogesch Micro Systems, Inc.

TopSpeed is a registered trademark of Jensen and Partners, International.

Any other trademarks inadvertently used here are also duly acknowledged.