Chapter 2: Program Structure

2.1 Hello World

Let's look at the basic structure of a Fortran program by writing a customary "Hello World" program.

hello.f95

```
1 program hello
2 implicit none
3    write(*,*) 'Hello world'
4    ! Equivalently,
5    ! print*, 'Hello world'
6    ! write(6,*) 'Hello world'
7 end program hello
```

- The *source code* for the program is delimited by the `program PROGRAM_NAME ... end program PROGRAM_NAME` tags.
- Fortran is a compiled language in which variables are *explicitly* declared. For example, an integer `i` is declared by including `integer :: i` at the top of the source code. If after declaring `i` as an integer, you assign it a value of 1 with `i=1`, your computer knows not to waste disk space on storing the decimal component of `i` since it is zero. Your computer also has certain rules for dealing with undeclared, that is *implicitly* declared, variables. Using implicitly declared variables makes it more difficult to read or debug code so we avoid using them by including the statement `implicit none` immediately after the program declaration. This statement instructs your computer to throw an error when an undeclared variable is encountered.
- The `write` command is used to output data to a particular destination and in a particular format. The asterisks ("*" is an asterisk) in `write(*,*)` tell your computer to output to the default destination (the terminal screen) with the default formatting ("list-directed" or free format). The first asterisk is for destination and the second is for formatting. The default destination is assigned the file unit "6" so `write(6,*)` has the same effect as `write(*,*)`. Additionally, `print*,` has the same effect as `write(*,*)`.
- An exclamation point ("!" is an exclamation point) is used to comment. The compiler will ignore anything to the right on the same line as an exclamation point.
- The file extension `.f95` indicates that the source code is written in the 1995 version of Fortran. The language has gone through a number of revisions since it first appeared, but the most recent version that is fully supported by `gfortran` is Fortran 95.
- Fortran 95 is **not case-sensitive**. For example, the keywords `program`, `PROGRAM`, and `PrOgRaM` all have exactly the same effect. Furthermore, if you try to declare two variables with names `i` and `I`, an error will be thrown indicating that a duplicate variable was declared.

Invoking the `gfortran` *compiler* translates Fortran source code into executable *machine code*, a *binary*, that can be called to run.

hello - commands and output

```
gfortran hello.f95 -o hello
./hello
Hello world
```

- Calling `gfortran` with the option `-o hello` instructs the compiler to ouput to the file `hello`. If this option is excluded, the compiler by default outputs to `a.out`.
- While in the same directory, we can execute the binary with `./hello`, which prints "Hello World" to the terminal screen.

2.2 Automating Your Report

In numerical mathematics courses, you will be expected to write programs and submit reports that explain how your program works and the results of any tests you ran with it. You will often need to create tables of numerical data, graphical plots, and code listings. It is useful to automate as much of this process as possible so that incremental updates can easily be incorporated.

In this section, we present a technique for creating a automated report. In particular, we create a single program from Fortran that does some computations, creates tables, creates figures, and collects everything into a document. Most of the Fortran syntax may be new to you now but we will look at it more closely in subsequent chapters. Furthermore, the program will call a `gnuplot` script to plot data and call LaTeX to compile a report, which requires installations of `gnuplot` and LaTeX callable from the command line and knowledge of `gnuplot` and LaTeX syntax.

Let's examine the following source code for a program that creates an automated report.

automate/automate.f95

```
1  program automate
2  implicit none
3     integer :: i
4     real :: x(0:10) ! an array indexed from 0 to 10
5     ! compute pi and store as a constant
6     real, parameter :: pi = 2.*acos(0.)
7
8     ! populate array of x-values between 0 and 2 pi
9     x=(/(i/5.*pi,i=0,10)/)
10
11    ! write sine and cosine data to file 'figure.dat'
12    open(10,file='figure.dat',action='write',status='replace')
13    do i=0,10
14       write(10,*) x(i), sin(x(i)), cos(x(i))
15    enddo
16    close(10)
17    ! call gnuplot script 'automate.plt' that plots data
18    call execute_command_line('gnuplot automate.plt', wait=.true.)
19
20    ! write LaTeX table to file 'table.tex'
21    open(10,file='table.tex',action='write',status='replace')
22    write(10,*) '\begin{tabular}{|c|c|c|} \hline'
23    write(10,*) '$x$ & $\sin x$ & $\cos x$ \\ \hline'
24    do i=0,10
25       write(10,*) x(i), '&', sin(x(i)), '&', cos(x(i)), '\\'
26    enddo
27    write(10,*)'\hline \end{tabular}'
28    close(10)
29    ! call pdflatex on 'automate.tex' to compile report to pdf
30    call execute_command_line('pdflatex automate.tex', wait=.true.)
31 end program automate
```

- Lines 3-9 declare variables (an integer `i`, an array of real numbers `x`, and a real parameter `pi`) and populate `x` with values $x = i\frac{\pi}{5}$ for $i = 0, 1, 2, \ldots, 10$.

- Lines 11-16 open a file with unit 10 to be overwritten called `figure.dat` and output data in three columns: x, $\sin x$, and $\cos x$, for $x = i\frac{\pi}{5}$ for $i = 0, 1, 2, \ldots, 10$, then close the file. Similarly, lines 21-28 open a file with unit 10 to be overwritten called `table.tex` and output the same data but in LaTeX table syntax.
- Line 18 calls `gnuplot` to execute the script `automate.plt`. The script was written separately and requires knowledge of the `gnuplot` syntax. The script produces the plot `figure.eps`.
- Line 30 calls `pdflatex` to compile the report source file `automate.tex`. It was written separately and requires knowledge of the LaTeX syntax. The table is included in `automate.tex` with the line `\input{table.tex}`. The plot is included in `automate.tex` using the LaTeX `graphicx` package.

The source files `automate.f95`, `automate.plt`, and `automate.tex` located in `f95/automate` can be used as a starting point for creating your own automated report.

Exercise

1. Install `gfortran`. Write and execute a "Hello World" program in Fortran.