CS111 Introduction to Systematic Programming

Second Practical Class

If you did not attend the first practical or if you did not complete the worksheet, work through the hand-out for that practical BEFORE working on this sheet.

Open the file `prog1.adb` that you created in the firsst practical class. Remember you can do this in several ways:

1. Type in an `xterm` window
```
emacs prog1.adb &
```
2. Select Emacs from the **Editors** submenu of the CS Root menu and then open the file by selecting **Open File** from the Emacs **Files** menu.
3. Start Emacs plus filename prompt from the **Editors** submenu of the CS Root menu and then type the name of the file in the dialog box that appears.

Compiling an Ada Program

The computer cannot execute an Ada program such as `prog1.adb` directly; first it needs to be converted into machine code that the compuete can execute by a process known as compiling, binding and linking

1) To compile the Ada program `prog1.adb` (say) from within Emacs, from the Emacs **Ada** menu select **Build**. This invokes the GNAT Ada compiler to convert your Ada program, to binary machine code suitable for execution by the computer. The Emacs window will split into two panes and the lower pane will show the results of the compilation process.

   If the program compiles sucessfully, this pane will show the progress of the compilation process which will end with a message of the form[1]

   ```
   Compilation finished at Wed Oct  9 12:47:03
   ```

   If the compilation is successful an executable file called `prog1` is created. You can run this by selecting **Run** from the Emacs **Ada** menu. Type in two numbers and the program will add them up and display the result. Select **Run** again if you want to run the program again with different input.

   Note the Ada menu belongs to the pane containing the Ada program and is not accessible when the compilation pane has the focus. If the Ada menu is not visible, click in the upper pane of the Emacs window so that the Ada program pane has the focus.

   If the compilation fails, the compiler outputs one or more error messages in the compilation pane followed by a message of the form

   ```
   Compilation exited abnormally with code 4 at Wed Oct  9 14:46:58
   ```

   This means that the program contains one or more Ada errors (or bugs) and that it needs **debugging**. In the Emacs window carefully correct any typos and other errors; you will need to look at the hand-out for practical 1 for the correct version of `prog1.adb`.

   Use the compiler error messages to help locate and correct the errors. Error mesages take the form:

   ```
   prog1.adb:2:01: misplaced "with"
   prog1.adb:8:10: "Frst" is undefined
   ```

[1] The compiler will issue a warning message about the file and the unit names not matching -- do not worry about this.

indicating that there is an error in line 2 column 1 of the file `prog1.adb` and that it has something to do with the `WITH` command and another error near column 10 of line 8 and that the variable `Frst` has not been defined. (In fact the programmer mistakenly typed `Frst` instead of `First`).

When you have corrected all the errors, save the modified file (choose **Save Buffer** from Emacs' **Files** menu) and try compiling it again as above.

If the program now compiles without error, run it as described above. If it still contains errors, repeat the debugging process.

If at any stage you wish to get rid of the compilation pane, click in the upper pane and then select **One Window** from the Emacs **Files** menu.

3) Before you can run your program you need to link the object (`.o`) file produced by the compiler into a complete executable program using the Gnat binder and linker. The binder and linker take the object file and link it with the required Ada library packages and support files to form a complete executable program. The binding and linking process is controlled by information in the Ada Library Info (`.ali`) file. To bind and link 'in one go', type (in an `xterm` window)

```
          gnatbl prog1.ali      # note the extension is .ali not
.adb
```

The binding and linking process creates a binary executable program called `prog1`.

4) To run this program type

```
          prog1
```

(followed as always by `<Return>`). Then type in two whole numbers (separated by at least one space) and press `<Return>` and the program will compute and output their sum.

Warning The files `prog1.o`, `prog1.ali` and the final executable program `prog1` are not intended to be read by humans and, in particular, you should not attempt to display the binary files `prog1.o` and `prog1` using utilities such as `cat`, `less` or `more`; the output will be gobble-de-gook and may render your `xterm` window unusable.

Note that if, when you try to compile or link your programs, the system complains that it can't find programs `gcc` or `gnatbl`, then this probably means you did not set up your environment correctly in the first lab class. <u>If so, ask the demonstrator for help</u>.

Scrolling Text in Windows[2]

Sometimes a window is not big enough to display all the information output by the computer and information disappears off the top of the window. In this case the scroll-bar at the side of the window becomes highlighted indicating that the window may be scrolled so this information becomes visible. The length of the scroll-bar indicates the whole of the document; the top of the scroll-bar representing the start of the document. The greyed-out portion of the scroll-bar (sometimes called the '**thumb**') represents the portion of the document that is currently visible.

[2] The summary of scrolling in this section refers to basic X-windows scroll-bars such as those used in `xterm` and `emacs`. Scrollbars in certain other applications (for example various SunTools, Netscape and Xemacs) behave differently but perhaps more intuitively.

To scroll to a different section of the document we need to move the scroll-thumb - imagine that the document is held fixed and that the scroll-thumb is a window which moves to reveal a different portion of the document. Scrolling may be done in a number of ways:

Scroll down one 'page'	Move the mouse to near the bottom of the scroll-bar and left-click.
Scroll up one 'page'	Move the mouse to near the bottom of the scroll-bar and right-click
Scroll down/up a fraction of a 'page'	Move the mouse the required fraction of the way down the scroll-bar and then left-click (scroll down) or right click (scroll up). For example to scroll half a page, click half-way down the scroll-bar.

The following additional operations are available on three-button mice on Sparc Solaris machines:

Continuous Scrolling	Move the mouse to the thumb and middle-drag the thumb until the required text is visible in the window in the file.
Scroll to the start of the file	Middle-click at the top of the scroll bar
Scroll to the end of the file	Middle-click at the bottom of the scroll bar
Scroll to a specified part of the file	Middle-click at the corresponding point in the scroll-bar.

For example to scroll to about halfway through the file, middle-click halfway down the scroll bar. To move to a third of the way through the document, middle-click a third of the way down the scroll-bar and so on.

Note with a two-button mouse you middle-click by pressing both buttons simultaneously -- this requires a little practice to master.

More Practice

If you want more practice at using the Ada compiler: copy the file `factors04.adb` from the directory `~barnesa/ISP/unit-programs` into own directory as follows:

```
cp ~barnesa/ISP/unit-programs/factors04.adb   ~
```

Now compile, bind and link `factors04.adb` -- this time use the `gnatmake` short-cut:

```
gnatmake factors04.adb
```

and then run it.

Now compile, bind and link the program, but give the executable a more meaningful name, by using the `-o` option followed by the required name. For example:

```
gnatmake -o FindFactors  factors04.adb
```

This produces an executable file called `FindFactors` which may be run by typing

```
FindFactors
```

Note

The case of characters in **Unix** commands and filenames IS significant; thus `FindFactors` and `findfactors` (etc.) are regarded as different filenames. Remember, however, that the case of characters within **Ada** programs (except in quoted strings) is not significant.