Scientific Software
Session 1: Fortran 95, Acquaintance

Emil Loevbak, Joris Tavernier & Pieter Appeltans October 4, 2018

Introduction

This introductory session serves an an introduction to Fortran 95. The aim is for you to develop an understanding of the following concepts which will come back in the homework assignments:

- Basic Fortran 95 syntax: Input/output, loops, if-then, select-case, ...
- Using simple makefiles
- Fortran modules
- Elemental functions

The syntax you need for the session can be found in the preparatory reading, but you can ask further questions to the assistants in the session. If you have questions later, please post them on the Toledo forum. That way other students can see the questions that have already been asked, as well as their answer.

Exercises

1. Make a program that reads in a number (you can assume it to be an integer) and then prints out `Hello, world!` as many times. Add extra checks so that meaningless input or ridiculously large numbers are handled in a proper way.

2. Put the code from the previous exercise as a subroutine without arguments in a module `hello_module` that is contained in the file `hello_module.f95` or `hello_module.f90`. Adjust the program so that it uses this subroutine. The structure of these program units can be found in the 'cheatsheet'.

   Make the program by first compiling `hello_module` to object code, then the program to object code and then composing the executable file.

   Try making the program with only one of the two object files to get an idea of the error messages that are caused by it. What is the effect when you try to compile your program, to an object file in absence of `hello_module.mod`?

3. Get the `Makefile` from Toledo, remove possible brackets in the name of the file and compile the program from the previous exercise through `make hello_world`. Try finding out how this system uses the mutual dependences in the `Makefile` to compile as little as possible. Examine the effect of a change in the program if you execute `make hello_world` again afterwards or if you only do the `linking`.

4. Write a program that will print the character sequence "Here's to the crazy ones. The misfits. The rebels.": left aligned, centered and right aligned. Take 80 characters as screen width. A possible output is as follows:

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
Here's to the crazy ones. The misfits. The rebels.
              Here's to the crazy ones. The misfits. The rebels.
                                  Here's to the crazy ones. The misfits. The rebels.
```

5. Make an `elemental` function `graycode(n)` that computes the decimal value of the Gray-code of n for whole numbers n via the formula (1) where "$\oplus$" represents the bitwise exclusive OR-operation. With "$\gg$" we mean a shift of one bit to the right.

$$\text{GrayCode} = n \oplus (n \gg 1) \tag{1}$$

Your function should be `elemental` and not use loops but work with the available bit manipulation functions. Test with a scalar as well as a vector and an matrix and of course leave these tests in the code as well as the output at the top. Next, let the program compute the first 20 graycode numbers. What property do consecutive graycode numbers have?

Decimaal	0	1	2	3	4	5	6	...
n (binair)	0000	0001	0010	0011	0100	0101	0110	...
	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	
$n \gg 1$ (binary)	0000	0000	0001	0001	0010	0010	0011	...
	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	
$n \oplus (n \gg 1)$ (binary)	0000	0001	0011	0010	0110	0111	0101	...
Gray-code	0	1	3	2	6	7	5	...