Zero-weighted Exercise (0%): Introduction to C

CS2002 - Jon Lewis <jon.lewis@st-andrews.ac.uk>

Due date: Wednesday 12th February (Week 3), 21:00

This exercise does not count towards your module grade. However, it is recommended that you attempt the exercise as it gives you the opportunity to get used to the C programming language and run stacscheck over your program developed in C.

Objective

- to introduce you to programming in C
- to introduce you to running stacscheck over your program developed in C

Learning Outcomes

By the end of this practical you should understand:

- writing a makefile to build your program
- writing simple functions in C
- how to run the automated checker on your solution written in C

Getting started

To start with, you should create a suitable assignment directory such as `CS2002/W03-Exercise` on the Linux lab clients.

Requirements

In this exercise, you will write a program, in three stages, to calculate a sequence of numbers called the "Weighted Fibonacci sequence". You will do this in two stages. You should submit a zip file containing:

- Your source code, with a Makefile, which must be in a directory called `src` within your assignment directory, for stacscheck.
- A short report covering your design, implementation and testing, any problems encountered and lessons learned, as a PDF, and including the output of stacscheck for your submission.

A C Program to Calculate The Weighted Fibonacci Sequence

You are expected to create correct Makefile entries for each stage. These should have correct dependencies, so if one file is changed the correct objects are rebuilt. The directory below contains some examples which may be of help.

`https://studres.cs.st-andrews.ac.uk/CS2002/Examples/`

The standard Weighted Fibonacci series begins $0, 1$. After this, each term in the Weighted Fibonacci sequence is generated by adding the previous term, plus twice the term before that. Therefore the third element is $0 * 2 + 1 = 1$, and the fourth element is $1 * 2 + 1 = 3$.

Stage 1: Recursive Calculation

1) Write a function `int fibcalc(int n)` to calculate the Weighted Fibonacci of n recursively. This function should return 0 for `fibcalc(0)`, 1 for `fibcalc(1)`, and calculate other values recursively.

You should put the function implementation in a source code file called `recursive_fib.c` and the function prototype in a header file called `fib.h`, to `#include` in the files that need to use the function.

2) Write a function `void print_fib()` in a new file, `print_fib.c` and once again put the function declaration (prototype) into a file `print_fib.h`. Your `void print_fib` function should:

- Print the prompt, `Length?`
- Read an integer n from the user
- Print the first n elements of the Weighted Fibonacci sequence, using the `fibcalc` function.

For example, with input '7' your program's output should look like the following (where the '7' is entered by the user). Note the list consists of a space after each comma. Empty lists should be denoted as "[]".

```
Length? 7
[0, 1, 1, 3, 5, 11, 21]
```

You are not required to check whether user input is valid.

3) Write a main function in a new file `run_fib.c`. Your main function merely needs to call `print_fib`.

4) Write a makefile, building an executable called "stage1" from your three `.c` files. Be sure to get the dependencies right. Your Makefile should also have a target 'clean', which removes any binary or object files.

5) You should test your program using stacscheck. To do this you should:

- Name your assignment directory `W03-Exercise`
- Make sure `make clean` works. `make stage1` should work and produce an executable called `stage1`.
- From inside your `W03-Exercise`, run from a lab machine:
  `stacscheck /cs/studres/CS2002/Practicals/W03-Exercise/Tests`

At this stage of the practical, the 'build-clean' and 'stage1' tests should pass. We will work on the 'stage2' and 'invalid-input' tests in later parts.

If nothing is working and you don't know why, please don't suffer in silence, ask one of the demonstrators in the lab.

Stage 2: Iterative Calculation

1) In a new file `iterative_fib.c`, write a new function that calculates the Weighted Fibonacci of a variable iteratively, rather than recursively. In particular, this function should only calculate the fib of each value at most once for a given input. Give this function the same prototype as your original fib function.

2) Add a rule to your Makefile which builds a new executable, `stage2`, using this new function, by linking the output from compiling `run_fib.c`, `print_fib.c` and `iterative_fib.c`.

Optional Checking input

Modify the files used in Stage 2 so your program never misbehaves for any inputs. Discuss the things you check for, and how you check.

If your program detects an invalid input from the user, it should print `"Invalid input\n"`, then call ’exit’. If it detects an overflow, it should print `"Overflow\n"`, then call ’exit’.
Some hints:

- Don’t assume long is bigger than int (you can assume int is at least 32 bits, and long long is at least 64 bits, if you wish)
- You should reject negative inputs in all places.
- It is undefined behaviour to overflow an int.
- Read the documentation of `scanf` (type `man scanf` at the command line, or consult the web) to find out how to check the return value of scanf to see if there was an error.

Deliverables

Hand in via MMS, by the deadline of 9pm on Wednesday of Week 3, a zip file containing your assignment directory in which there is a `src` directory containing all source code files and your PDF report.

Marking

The submission does not count towards your module grade and you will not receive a mark on MMS. However, we will upload the output from the automated checker for all our tests to MMS.

I would remind you to ensure you are following the relevant guidelines on good academic practice as outlined at

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`