Numerical Methods; October–November 2019
Assignment 2
Due: Friday, 29 November 2019

C

▶ Spend at least 15 minutes browsing through the C standard library documentation and GNU Scientific Library (GSL) documentation. Understand how the documentation is structured. Spot the API.

▶ Install the `gcc` C compiler on your computer. Using the `gcc` documentation, learn how to find out the version of the `gcc` compiler version installed. What version of this compiler do you have? When was this version published?

▶ Write a C code that prints the size of the `char`, `int`, `long int`, `float`, `double` data types on your computer in bytes.

▶ Write a C code that prints the maximum and minimum machine numbers available on your computer for the `char`, `int`, `long int`, `float`, `double` data types.

▶ Write a C code to do the following:

1. Using the `malloc` standard library function, create an array of hundred `float` elements.
2. Assign the values 1^2, 2^2, ..., 100^2 to the elements of this array.
3. Create a function that takes in an array of arbitrary length and returns an array of two elements. The first element of the output array is the mean of the input array. The second element of the output array is the variance of the input array. Use loops to calculate the mean and variance.
4. Using this function, calculate the mean and variance of the array you constructed in step 2 and print these to your terminal.
5. Free the memory used for your array.

Now modify the above code by moving your function to a second file. Use a header file to declare the function prototype. Use a Makefile to compile the three files over which your code is now spread.

Again modify your code. This time, while assigning values to the array, print the values of the elements of the array on the terminal *before* the assignment.

Now repeat the above step (that is, print the array element values before assigning the squares to them) but use `calloc` instead of `malloc`. What changed?

Now make another modification to the code. Write the result of your computation to a text file instead of the terminal. After running the code, open the text file using your text editor and confirm that it contains the two numbers that you expect it to contain.

Next, instead of writing the result of your computation to a text file, write it to a binary file. Open the binary file in your text editor. What do you see? Now write a second C code to read the contents of this binary file.

Using the `du` command on GNU/Linux, find out the file sizes of the text file and the binary files that you created above? Which one's smaller? Which file format will you use to store large amount of data?

Finally, change your code one last time to use GSL functions to compute the mean and variance instead of your loops. Print the result of both computational methods on the terminal. Are they identical?

(Each of the above modification should be a Git commit in your repository.)

C versus Python

▶ Write code to compute the factorial of a given integer n using four different techniques:

1. Using C
2. Using standard Python
3. Using Python, but with the `factorial` function from SciPy
4. Using Python, but writing the factorial function in C

In each case, you should have a function that computes the factorial of an arbitrary integer (use recursion) and then a main program that invokes this function to compute the factorial of a particular integer.

Now, compute the factorial of some large number and report the time taken to do this in each of the above four codes. (Use suitable C/Python library functions to measure time while the code is running.) Which technique is the fastest?

Analysis of algorithms

▶ Consider the quadratic equation $ax^2+bx+c=0$ with a, b, and c real numbers. Algorithm A computes the roots of this equation as

$$\frac{-b \pm \sqrt{b^2-4ac}}{2a}.$$

Compute the round-off error for this algorithm.

Algorithm B computes the roots as

$$\frac{2c}{-b \pm \sqrt{b^2-4ac}}.$$

Compute the round-off error for this algorithm.

Algorithm C computes the roots as q/a and c/q where

$$q = -\frac{1}{2}\left[b + \mathrm{sgn}(b)\sqrt{b^2 - 4ac}\right].$$

Compute the round-off error for this algorithm.

Of A, B, and C, which algorithm is numerically most trustworthy? Which algorithm is numerically stable?

▶ What is the machine precision ("eps") for the C `float` data type on your computer? What is the relative error in storing π on your computer using this data type? What is the relative error in storing 2.0 on your computer using this data type? When the `float` data type is used, is π a machine number on your computer? Is 2.0 a machine number?

▶ In Assignment 1, you wrote a code to calculate the first n Fibonacci numbers. What is the complexity of your code as a function of n? Write your answer using the O notation.