Universität Tübingen
Mathematisches Institut
Georgios Vretinaris

Tübingen, September 25, 2025

Exercise Sheet for Introduction to Programming with Julia

Exercise 1: Which of the following variable names are valid in Julia?

(a) 2after1
(b) the first
(c) theSecond
(d) the.third
(e) variable-number-1
(f) variable_number.2
(g) rates.in.%

Exercise 2: What value does the variable n have after line 4 and after line 8? Try to answer the question first without executing the commands in Julia.

```
1: n = 3;
2: n = n+1;
3: n = n/2;
4: n = n^3;
5: n = n-4;
6: n = n/4;
7: n = 5;
8: n = n+1;
```

Exercise 3: Write a script `exercise3.jl` where you enter the 8 lines from Exercise 2 and execute it. Now change the value of n in the first line and execute the script again. What do you notice? Can you explain what happens?

Exercise 4: Write a script `exercise4.jl` where you declare two variables a and b and output the quotient c=a/b. Execute this script for different values of a and b.

Exercise 5: Write a script `exercise5.jl` where you declare two variables x and y. The script should subtract the smaller number from the larger one using an if statement.

Exercise 6: Write a script `exercise6.jl` where you declare a natural number n. The script should output whether the number n has three digits using an if statement.

Exercise 7: Write a script that outputs the number of decimal places for a given integer x. The program should output an error message if $x < 0$, the exact number of digits if $0 \leq x < 1000$, and "more than three digits" if $x \geq 1000$.

What does the script output if you call it with a real number $x \in (0, 1000)$ that is not an integer?

Exercise 8: (a) Write a script that outputs the numbers from 0 to 10.

(b) Modify your program so that it outputs the odd numbers from 1 to 21.

Exercise 9: Write a script that calculates the sum

$$S = \sum_{k=1}^{n} \frac{1}{k^2}$$

for a given n.

Note: The limit for $n \to \infty$ of this series is $\frac{\pi^2}{6}$.

Exercise 10: Write a script that calculates the sum

$$S = \sum_{k=0}^{n} q^k$$

for given q and n.

What is the meaning of this formula? What is the limit for $n \to \infty$ (and for which q)?

Exercise 11: Write a script where you declare a positive number x and a tolerance tol. Make sure that the tolerance is less than the number. Then write a while loop where the number x is halved until it is smaller than tol.

Include a variable count that counts how often x was halved.

Exercise 12: It can be shown that $\lim_{n\to\infty} \sum_{k=0}^{n} (-1)^k/(2k+1) = \pi/4$. Thus, the number π can be approximated by calculating $4S_n$ for large n, where S_n denotes the n-th partial sum.

Using a while loop, write a program that determines the smallest n such that $|4S_n - \pi| < 10^{-8}$. You may use the number pi stored in Julia for the termination condition.

Why is your termination condition actually nonsensical if you want to approximate the number ?

Consider whether the above formula (from a numerical perspective) is reasonable for determining an approximation of π.

Exercise 13: Temperatures in degrees Celsius can be converted to degrees Fahrenheit using the formula $t_{\text{Fahr}} = \frac{9}{5} t_{\text{Cel}} + 32$. Write a Julia function `CelToFahr(tCel)` for this.

Exercise 14: Write a function `geomSum(q,N)` that calculates the N-th partial sum of the geometric series:

$$S = \sum_{k=0}^{N} q^k$$

Call the function for different values of q and N.

Clarify again the difference between a script and a function by comparing this function with the script from Exercise 10.

Exercise 15: Given two vectors $a, b \in \mathbb{R}^n$. What types of vector multiplication do you know? What dimensions do the respective results have? Write a function `innerProduct(a,b)` that calculates the dot product of two vectors.

Exercise 16: Write a Julia function `mean(v)` that calculates the arithmetic mean of the elements of v.

Exercise 17: Write a Julia function `norms(v)` that calculates the $\|\cdot\|_1, \|\cdot\|_2$ and $\|\cdot\|_\infty$ norms of the vector v.

Exercise 18: Write a Julia function `addElements(A)` that adds all entries of a matrix A.

Exercise 19: Write a Julia function that calculates the matrix-vector product w = Av for a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $v \in \mathbb{R}^n$.

Exercise 20: Write a function `matMult(A,B)` that multiplies two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times k}$.

Hints: The elements of C are

$$c_{ij} = \sum_{l=1}^{m} a_{il} b_{lj}, \quad i = 1, \ldots, n \quad j = 1, \ldots, k$$

Use the `size` function to determine m, n, and k. Pay attention to the correct initialization of C.

Exercise 21: Consider whether the function programmed in Exercise 20 can also calculate the matrix-vector product as in Exercise 19.

Exercise 22: Plot the function

$$f : [-2, 2] \to \mathbb{R}, \quad x \mapsto \begin{cases} -x^2, & x < 0 \\ x, & 0 \leq x \leq 1 \\ x^2 + 1, & x > 1 \end{cases}$$

Write a function `func_f` that implements this function, as well as a script `plot_f` that evaluates and plots the above function.

Pay attention to meaningful labeling of the plot.

How does the behavior of the plot differ from the behavior of the function f at the point x=1? Explanation?

Exercise 23: Write a function `fschar(x,b)` that expects the parameters x and b and returns the value $y(x) = bx^2$. Write a script `plot_fschar` that plots this function for $x \in [-1, 1]$ and $b \in 1/2, 1, 3/2, 2$ in one figure. Pay attention to meaningful labeling.

Exercise 24: For a function $f\colon [a, b] \to \mathbb{R}$, the derivative $f'(x_0)$ at a point x_0 can be approximated by calculating the difference quotient

$$\frac{f(x_0 + h) - f(x_0)}{h}$$

for a small h.

Create a program where you specify an interval [a,b], a function $f\colon [a, b] \to \mathbb{R}$, a number of (equidistant) support points in the interval [a,b], and a parameter h. The program should now calculate $f(x_i)$ at the support points and approximate the derivative $f'(x_i)$ at the support points x_i using the difference quotient. As an approximation for $f'(b)$, use the left-sided difference quotient $(f(b) - f(b - h))/h$. Then plot f and f' in one figure.

Clarify which parts of the program belong to preprocessing, processing, and postprocessing.

Exercise 25: Write a script where you approximately differentiate the sine function at $x_0 = 0$ using the difference quotient. For this, you need a vector with different values for $h > 0$, e.g., you can create a vector with the entries 0.01, 0.02, ..., 1 using h=0.01:0.01:1. For each value of h, calculate the difference quotient

$$\frac{f(h) - f(0)}{h}$$

as well as the absolute value of the difference to the exact value of the derivative $f'(0)$. Plot the errors in a doubly logarithmic plot. Plot the functions $h \mapsto h$ and $h \mapsto h^2$ in the same plot. With what order of convergence does the difference quotient converge?

Repeat the same with the exponential function. With what order of convergence does the difference quotient converge in this case?

Bonus question (difficult): Can you explain why this is?

Exercise 26: Write a non-recursive function to calculate the N-th number of the Fibonacci sequence

$$f_{n+2} = f_{n+1} + f_n, \quad f_2 = f_1 = 1$$

Hint: Initialize a suitable vector of length N and use a loop.

Compare the runtime of this (iterative) function with the recursive function for different values of N.

Bonus question: Approximately how many operations are required in the recursive algorithm to calculate the n-th Fibonacci number? How does the number of operations grow with n?

Exercise 27: In this exercise, you will implement a program to visualize the Mandelbrot set.

The Mandelbrot set $\mathcal{M}$ is defined as the set of all complex numbers c for which the recursive sequence

$$z_{n+1} = z_n^2 + c \quad \text{with initial value} \quad z_0 = 0$$

remains bounded for $n \to \infty$. Since it is not possible to perform infinitely many iterations to determine whether $c \in \mathcal{M}$, we proceed as follows: It is known that as soon as a z_n with $|z_n| > 2$ appears (complex absolute value), the sequence becomes unbounded. In particular, $\mathcal{M}$ is thus contained in a circle with radius 2 around the origin. One therefore specifies a maximum number of iterations $n_{\max}$ and calculates z_1, z_2, ... until either a z_n is found with $|z_n| > 2$, or the maximum number of iterations is reached, i.e., z_{nmax} has been calculated. In the first case, or if $|z_{\mathrm{nmax}}| > 2$, one decides that $c \notin \mathcal{M}$, in the second case $c \in \mathcal{M}$.

To graphically represent the entire set M or subsets of it, implement the following algorithm formulated in colloquial language:

- Choose a rectangle $R = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.
- Choose finitely many points from R that are to be checked for membership in $\mathcal{M}$. The simplest method is to specify a number of grid points N_x in the x-direction and N_y in the y-direction and then select equidistant points x_i, i=1,...,N_x from $x_{\min}$ to $x_{\max}$ and y_j, j=1,...,N_y from $y_{\min}$ to $y_{\max}$. The Julia command `LinRange` can be helpful here.
- Choose an upper bound $n_{\max}$ for how many sequence terms should be calculated at most.
- Initialize an $(N_x \times N_y)$ zero matrix M.
- For each point $c_{ij} = (x_i, y_j)$, perform the following steps:
  - Calculate the corresponding complex number $c_{ij} = x_i + \mathrm{i}y_j$ (where the small i stands for the index, the other for the imaginary unit).
  - Calculate the sequence (z_n) for this number (don't forget to initialize!).
  - After each newly calculated z_n, check whether $|z_n| > 2$. In this case, the calculation can be aborted.
  - If the maximum number $n_{\max}$ of iterations is reached and $|z_{\mathrm{nmax}}| < 2$, set M(i,j) = 1. This marks that $c_{ij} \in \mathcal{M}$.

  - Bonus: For those points c_{ij} where the calculation was aborted, set M[i,j] $= \ell / n_{\max}$, where ℓ was the index at which the calculation was aborted. This allows you to assign a value even to points that do not belong to $\mathcal{M}$, indicating how quickly the sequence exceeds the critical threshold of 2: the higher the value, the longer it took.
- The matrix M that you obtain as a result can be viewed as a function from $\mathbb{R}^2 \to \mathbb{R}$, where M[i,j] indicates how long the above sequence with $c_{ij} = x_i + \mathrm{i}y_j$ remains bounded. Functions from $\mathbb{R}^2 \to \mathbb{R}$ can be visualized using a colormap, where equal colors represent equal function values (cf. contour lines). This is already implemented in the template.

Download the script `mandelbrot_template.jl` and implement the described algorithm at the marked location in the template. Test your algorithm with different parameters that you can take from `mandelbrot_parameters.jl` or with your own settings. If you have implemented everything correctly, you should get beautiful images like the following.