Exercise 3
C/C++
(handed out: 23.04.2025 – hand in: 30.04.2025)

Review

1. What must be in every C/C++ *program* (1P)?
   - ☐ `#include`
   - ☐ `main(){}`
   - ☐ `return`
   - ☐ `;`

2. How to loop correctly over the whole array `int m[n]` (1P)?
   - ☐ `for (int i=1; i<=n ; ++i)`
   - ☐ `for (int i=0; i<n  ; ++i)`
   - ☐ `for (int i=1; i<n  ; ++i)`
   - ☐ `for (int i=0; i<=n ; ++i)`

---

1. Task *Importing data into arrays* (4 P)
Write a C++ program that imports x-y data pairs into an array.

a) First, arrays `x[100]` and `y[100]` of type `double` shall be declared. (0.5 P)

b) The user should specify how many data pairs n they want to enter. (0.5 P)

c) The program then imports via a `for`-loop the x- and y-values entered by the user into the arrays `x[]` and `y[]` (0.5 P):
`cin >> x[i] >> y[i] ;`

d) The imported data should be printed out pairwise in form of an x-y table for checking. (0.5 P)

The imported data points shall be analyzed (2 P):

e) The line of best fit for the entered values shall be determined:

$$y = b \cdot x + a \tag{1}$$

where

$$b = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}\,, \tag{2}$$

$$a = \overline{y} - b \cdot \overline{x}\,, \tag{3}$$

with the means $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.
The coefficients a, b shall be printed out.

As a test case: For the following pairs one should obtain $a \approx 0.3883$ and $b \approx 2.0511$

```
x = 1.2  y = 2.6
x = 3.4  y = 8.7
x = 6.9  y = 12.6
x = 8.9  y = 18.2
x = 10.1 y = 22.4
```

2. Task *Pointers, references, arrays* (6 P)

As we will use pointers, references, and arrays in upcoming exercises it might be helpful to bring their functionality to mind:

a) Write a short C++ program which uses *references* as shown in the lecture (subsection "References").

Pointers and arrays correspond to each other in some way. Have a look on the following program lines and answer the questions:

```
int array[6] ;        // (1)
int *parray = 0 ;     // (2)
parray = array ;      // (3)
parray[2] = 1 ;       // (4)
parray = &array[3] ; // (5)
parray[2] = 6 ;       // (6)
```

b) Why should a pointer always be initialized with 0, as in line (2)?

c) By the intriguing assignment in line (3) the pointer contains now the start address of the array. How is this assignment normally done?

d) The pointer behaves then like an array, i.e. it can be "indexed" by square brackets. How does this work?

e) What is the effect of the lines (5) and (6)?

f) At which *array* index is then the value 6 stored?

(each sub task: 1P)

3. Task *Catching invald input* (2 P + 1 BP)

Extend your C++ program from exercise 2.4 for radius calculation so that it can catch invalid input (2 P):

a) The user should be asked for a valid value if they enter a non-positive number for the temperature.

b) Use the appropriate kind of loop (which one?) to do this and provide meaningful output to the user.

Bonus: Also try to catch non-number input (e.g., letters). (1 BP)