Einführung in die Programmierung für Physiker

WS 2019/2020 – Marc Wagner

Alessandro Sciarra: sciarra@itp.uni-frankfurt.de

Exercise sheet 4

To be corrected in tutorials in the week from 11.11 to 15.11.2019

Exercise 1 [*The if-clause*]

One of the most important building blocks in every programming language is the `if`-clause that is needed to test conditions and act according to the logical value of a given condition.

(i) Write `if`-clauses for each of the following cases. Use an appropriate `printf` command in every branch to give information to the user.

(a) Given a non negative integer number, check if it is even.

(b) Given 2 numbers, find the largest.

(c) Given 4 numbers, find the smallest.

(ii) In `C` there is the so-called *conditional operator* (also known as *ternary operator*), whose syntax reads

```
condition ? value_if_true : value_if_false
```

and it evaluates to `value_if_true` if `condition` is `true`, to `value_if_false` otherwise. It is important to remark that the conditional operator is an *expression*, whereas `if-else` is a *statement*. Being the result of an expression a value, the ternary operator can then be used wherever a value is needed, e.g. on the right-hand side in assignments. Practice a bit with this operator using it *exclusively* to accomplish the following tasks (do not use any `if`-clause here).

(a) Given an integer number, assign to a different variable its sign.

(b) Given 2 numbers, assign to a different variable the smallest.

(c) Given 3 numbers, assign to a different variable the largest.

As you might have notice in the last task, code can quickly become hard to read as soon as the conditions become more complicated and, maybe, more than one ternary operator is needed. This teaches you that an `if`-clause should be preferred in every non-trivial case.

Exercise 2 [*Triangles taxonomy*]

Write a program which takes as input 3 floating point numbers, stores them in variables of type `double`, and checks whether they could correctly represent the length of the three sides of a triangle based on the constraints that

(i) all given values are positive numbers;

(ii) the length of any side is between the absolute value of the difference and the sum of the other two.

What kind of triangle can be drawn with the sides read in input? Print to the user

(iii) whether it is an equilateral triangle;

(iv) whether it is an isosceles triangle;

(v) whether it is a scalene triangle;

(vi) and whether, being it isosceles or scalene, it is a right-angled triangle.

Produced with the ExerciseHandler

Time to Test! This exercise, despite its harmless appearing, hides some profound aspect of computer programming. Checking if two floating point numbers are equal is a very complicated business and, strictly speaking, there is not a correct way of doing that. Whaaat? We have the `==` operator, isn't it?! Well, yes and no. Let us test together your code to understand why it is complicated to compare non-integer numbers. If we choose easy cases like 3, 4 and 5 as input values, we should get what we expect. Run your code and check it. But what happens in a trickier case, like with 3/7, 4/7 and 5/7? Try to give `0.42857142857142857`, `0.57142857142857142857` and `0.71428571428571428571` as input and check if you obtain a right triangle as output. Are you able to explain what is going on?
If everything is as you expected, then you should be able to explain the output of the following code

```
#include<stdio.h>
int main(){
    float x = 0.1f, sum = 0, product = x*10;

    for(int i=0; i<10; ++i)
        sum += x;

    if (sum == product)
        printf("10*0.1f is equal to 0.1f+...+0.1f ten times\n");
    else
        printf("10*0.1f is NOT equal to 0.1f+...+0.1f ten times\n");

    printf("sum-product = %1.15f\n", sum-product);
}
```

which gives the following output on on most of systems.

```
10*0.1f is NOT equal to 0.1f+...+0.1f ten times
sum-product = 0.000000119209290
```

Discuss with your tutor the following points.

- What does the `==` operator do exactly (i.e. also when used between floating point numbers)?
- Is this what you need in general?
- Which might be a good idea to test equality of floating point numbers?

Produced with the ExerciseHandler