Einführung in die Programmierung für Physiker

WS 2019/2020 – Marc Wagner

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

Exercise sheet 12

To be corrected in tutorials in the week from 27.01 to 31.01.2020

Exercise 1 [*New data types*]

Having the possibility to create customised data types is a common feature of many program languages and it helps in many situations. In this exercise we will deal with the common problem of having to handle many setup parameters for a large application. Often, such parameters might come from a file or from the user (or both). Here we suppose that all input from the user is given via command line parameters. You should already be familiar with the parsing of command line options.

(i) Use the `struct` keyword to define a new data type containing

- a real number `exponent`;
- an array of integer numbers `range[3]` (minimum, maximum, increment);
- a character `mode` corresponding to some functionality.

Use the name `Parameters` as tag of the structure.

(ii) Implement a function

```
Parameters ParseCommandLineOptions(int argc, char *argv[]);
```

that constructs and returns a variable of the new type using the command line parameters. You are free to decide in which way and which order options have to appear in the command line. Do not forget a `--help` option which explains your program usage.

(iii) Add another function `void PrintParameters(/*...*/)` to print the parameters to the output. Which is the difference between passing a `const struct Parameters` to a function rather than a `const struct Parameters*`? Which could be an argument in favour of the first or of the second? Did you test the passed pointer for `NULL` value in your function?

(iv) In your `main`, declare a `Parameters` variable, initialize it parsing the command line options and print the parameters to the screen.

The code implemented in this exercise can be adapted and reused in your future larger project(s).

Exercise 2 [*Exam-like questions*]

(i) Write a function that takes a number of seconds, convert it into hours, minutes and seconds and print the result to the screen.

(ii) Write a recursive function that, given two integers a and $b > 0$, returns a^b.

(iii) What operators can you use to access members of some structure?

Assume that the pointer `myTrianglePtr` has been declared to point to `struct Triangle` having a member `height` of type `float`. Assume also that the address of the structure `myTriangle` has been assigned to `myTrianglePtr`. Rewrite the expression `myTrianglePtr->height` with an equivalent one using the structure member operator `.` to access the relevant member.

(iv) Write a declaration statement for a variable of some defined structure type. Using which keyword can you create an alias for a derived data type and how? Rewrite your declaration statement making use of your alias.

(v) State whether the following are *true* or *false* and, in the latter case, explain why.

(a) The structure definition

```
struct Triangle {
    float base;
    float height;
};
```

reserves space in memory for the struct being defined.

(b) Structure members must be variables of built-in data types.

(c) One cannot just declare variables of a given structure type, but also arrays whose elements are of a given structure type and pointers to structure type variables.

(d) Structure can be compared using the comparison operators `==` and `!=` and can be operands of arithmetic operations.