Laboratory 12 – Command-line arguments and Structures 04.01.2022

1. [2points] Write a C program that prints its name on the screen.

Test data:

```
gcc ex1.c -o first_program
./first_program
```

```
./first_program
```

```
gcc ex1.c
./a.out
```

```
./a.out
```

2. [3points] Use the command line interface to create a simple calculator. Use the `switch` command to select the calculations to be performed. If the number of command line arguments is insufficient, **print** an error message and **exit** the program. We perform calculations on variables of the `float` type, use the appropriate function to convert two command line parameters to numerical values. Don't use `scanf`. To select multiplication on the command line, don't use `"*"`, use `"x"` instead. The operator taken from the command line is a **`string`**, in the `switch` we use a `char` type.

Test data:

```
./a.out
Not enough command line arguments.
```

```
./a.out 12.13 + 13.56
12.13 + 13.56 = 25.69
```

```
./a.out 12.13 - 13.56
12.13 - 13.56 = -1.43
```

```
./a.out 12.13 / 13.56
12.13 / 13.56 = 0.89
```

```
./a.out 12.13 x 13.56
12.13 x 13.56 = 164.48
```

```
./a.out 12.13 @ 13.56
Invalid operator
```

3. [10points] Operations on vectors in three-dimensional space.

a) [1points] Declare a structure for a vector with three `float` coordinates.

b) [2points] Define a function that computes the dot product of two vectors. We pass two structures to the function and return a number of the `float` type.

The dot product of two vectors **a** = $[a_1, a_2, \ldots, a_n]$ and **b** = $[b_1, b_2, \ldots, b_n]$ is defined as:[2]

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

where Σ denotes summation and n is the dimension of the vector space. For instance, in three-dimensional space, the dot product of vectors $[1, 3, -5]$ and $[4, -2, -1]$ is:

$$\begin{aligned}[1,3,-5] \cdot [4,-2,-1] &= (1 \times 4) + (3 \times -2) + (-5 \times -1) \\ &= 4 - 6 + 5 \\ &= 3\end{aligned}$$

c) [1points] Define a function that will calculate the length of the vector, use `sqrt` and dot product.

This implies that the dot product of a vector **a** with itself is

$$\mathbf{a} \cdot \mathbf{a} = \|\mathbf{a}\|^2,$$

which gives

$$\|\mathbf{a}\| = \sqrt{\mathbf{a} \cdot \mathbf{a}},$$

the formula for the Euclidean length of the vector.

d) [3points] Define a function that will calculate the cross(vector) product. We pass two structures to the function and return the structure.

For vectors $\mathbf{u} = (u_x, u_y, u_z)$ and $\mathbf{v} = (v_x, v_y, v_z)$ in $\mathbb{R}^3$, the cross product in is defined by

$$\begin{aligned}\mathbf{u} \times \mathbf{v} &= \hat{\mathbf{x}}(u_y v_z - u_z v_y) - \hat{\mathbf{y}}(u_x v_z - u_z v_x) + \hat{\mathbf{z}}(u_x v_y - u_y v_x) \\ &= \hat{\mathbf{x}}(u_y v_z - u_z v_y) + \hat{\mathbf{y}}(u_z v_x - u_x v_z) + \hat{\mathbf{z}}(u_x v_y - u_y v_x),\end{aligned}$$

where $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ is a right-handed, i.e., positively oriented, orthonormal basis.

e) [1points] Define a function that will calculate the mixed(scalar triple) product. Use previously defined functions. We pass three structures to the function and return a number of the `float` type.

The **scalar triple product** (also called the **mixed product**, **box product**, or **triple scalar product**) is defined as the dot product of one of the vectors with the cross product of the other two.

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$$

f) [2points] In the `main` function:

- Using the previously declared structure, declare three vectors and initialize each coordinate with a pseudo-random number.
- Print three vectors to the screen.
- Calculate the lengths of the vectors.
- Calculate the dot product of any two vectors.
- Calculate the cross product of any two vectors.
- Calculate the mixed product of three vectors.

Test data:

```
w1 = [0.05, 0.13, 0.75]
w2 = [0.89, 0.07, 0.55]
w3 = [0.68, 0.11, 0.58]
||w1|| = 0.77
||w2|| = 1.05
||w3|| = 0.90
w1 * w2 = 0.47
w1 x w2 = w4 = [0.02, 0.64, -0.11]
w1 * (w2 x w3) = 0.02
```

4. [5points] Nested structures and arrays of structures.

Declare a **vectorlen** structure that has two fields: the **vector** structure declared in the previous task, and a `float` variable that represents the length of the vector.

In the `main` function, declare an **array** of `10` **vectorlen** structures.

In a loop, initialize each element of the **array** that is a **structure** that has `3` **vector** coordinates of a pseudo-random number. Then calculate the length of the **vector** and assign this value to the appropriate field of the **vectorlen** structure.

Define a function that prints the fields of the **vectorlen** structure, and then in the `main` function print all the elements of the **array** of `10` **vectorlen** structures.

vectorlen - is the name of the structure that we declare. Any other name can be used.

`Test data:`

```
w[0] = [0.36, 0.93, 0.89], ||w[0]|| = 1.34
w[1] = [0.09, 0.65, 0.33], ||w[1]|| = 0.73
w[2] = [0.81, 0.78, 0.89], ||w[2]|| = 1.43
w[3] = [0.56, 0.36, 0.46], ||w[3]|| = 0.81
w[4] = [0.77, 0.06, 0.36], ||w[4]|| = 0.85
w[5] = [0.16, 0.11, 0.31], ||w[5]|| = 0.37
w[6] = [0.19, 0.48, 0.55], ||w[6]|| = 0.75
w[7] = [0.99, 0.34, 0.42], ||w[7]|| = 1.12
w[8] = [0.71, 0.79, 0.83], ||w[8]|| = 1.35
w[9] = [0.27, 0.04, 0.88], ||w[9]|| = 0.92
```

Next lab 13 – Pointers to structures, pointers to functions, qsort, bseach.