UNIVERSITÄT DUISBURG ESSEN

NES Networked Embedded Systems

Programming in C/C++

Exercises 1

!!! Deadline: Wed, Oct 29th, 11:00 !!!

1. Write a program that prints the sizes of all simple data types (char, short, int, long, long long, float, double, long double, pointer). Include the output in your report. If possible, run with different compilers and on different platforms. Report the differences.
(code 10 pts, report 5 pts, no comments necessary)

2. Download file task2.c from moodle. First look at the source code, think what it is doing (especially what is printed) and why, and fill out the comments. Then run the program, compare the output with your expectations. Add more comments if there are any and try to explain.
(comments 30pts, no other report necessary, no code to write)

3. Write a program with the following global variable:

```
int cube[3][2][4] = { { {1,2,3,4} }, { {2,3}, {4,6,8,10} }, { {3,4,5,6}, {6,8,10} } };
```

In main(), first

- define three int variables x=1, y=1, z=2,
- define a pointer 'p' to int and assign it the address of cube[0][0][0],
- define a pointer 'vptr' of appropriate type and assign it the address of cube[x][y],
- define a pointer 'mptr' of appropriate type and assign it the address of cube[x].

Then, add some printf() statements printing out the addresses of some cube elements so that you are able to determine the internal storage structure of the cube. Report on it!
Add printf() statements to print

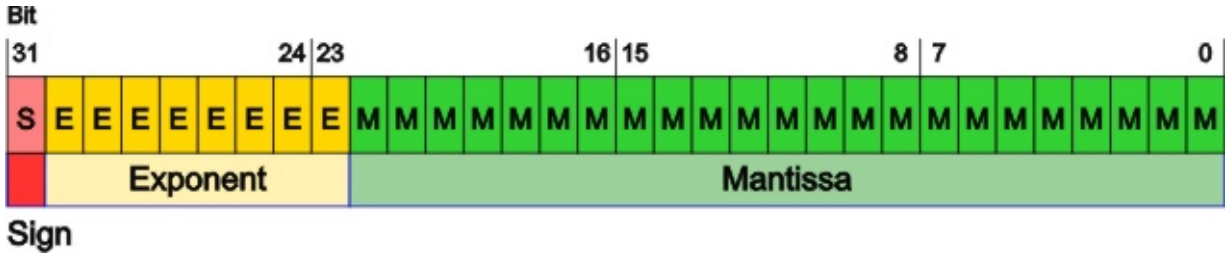
- the element cube[x][y][z],
- the same element using pointer p and the variables x,y,z (Hint: you need to do a calculation based on your previous findings),
- the same element using vptr,
- the same element using mptr.

(code 20 pts, report 5 pts, comments 5 pts)

4. Write a program that prints out the length of a string without using strlen(), i.e. implement the functionality yourself. The string is hardcoded as global variable. Implement a solution using an integer index and a solution using pointer arithmetic only, i.e. no additional length counter or index, and no sizeof() operator!
(code 15 pts, comments 10 pts, no report necessary)

p.t.o.

5. According to IEEE 754, floating point numbers are internally stored as follows



(Source: Wikipedia)

and calculated as $(1+m)2^{(e-127)}$ with m being only the fractional part. s determines the sign, if s==1 then the number is negative. For example 11 would be stored as m=0.375 and e=130 and be calculated as $(1+0.375)\bullet 2^{(130-127)} = 1.375\bullet 2^3 = 1.375\bullet 8 = 11$.

Create a data structure that allows you to store a float, read its raw byte representation and its internal representation as s, e and m. Use a combination of union and bit-field-struct.

Write a program where a float number is assigned to the float part of the structure and the raw and s/e/m representation is printed. Use hexadecimal output for raw and m.

Hint: Assign a negative and non-negative number to figure out in which order you have to define the s/e/m in the bit-field.

Bonus task: Print out the internal representation in decimal form, i.e. “m * 2 ^e”. Since calculation of the fractional part is painful use a trick and manipulate the internal representation in such a way that you can use the float representation in printf to print just “m”.

(code 20 pts, report 10pts, comments 10 pts. Bonus 10 pts)

6. Write a calculator program that accepts a single line input in form “<int> <op> <int>” with +-*/ as <op> and executes this calculation. Don’t forget error checks. The program should run in a loop that is exited with a single ‘0’ as input. First try scanf() for the input and report the behaviour. Then change to a combination of fgets()/sscanf(). Look at the man pages how to use them.
(code 40 pts, report 10 pts, comments 10 pts)