Einführung in die Programmierung für Physiker

SoSe 2024 – Marc Wagner

Michael Eichberg: eichberg@itp.uni-frankfurt.de

Exercise sheet 4

To be handed in on 17.05.2024.

Exercise 1 [*Floating-point numbers*] (1 + 1 + 2 + 2 + 1 + 1 = 8 pts.)

Let us assume we have a (rather primitive) computer that uses 8-bit floating-point arithmetic. The first bit represents the sign, the next 4 bits the exponent with bias $b = 7$ and the last 3 bits for the mantissa (normalized representation with leading 1 before the comma). With this we have the representation of a real number x as

$$x = (-1)^{\mathtt{s}} \left[1 + \sum_{n=1}^{3} \mathtt{m}_n \cdot 2^{-n} \right] \cdot 2^{\left(\sum_{i=0}^{3} \mathtt{e}_{3-i} \cdot 2^{3-i} \right) - b}$$

which can be stored in a bit string $\mathtt{s}\,\mathtt{e}_3\mathtt{e}_2\mathtt{e}_1\mathtt{e}_0\,\mathtt{m}_1\mathtt{m}_2\mathtt{m}_3$. Assume that non-representable numbers are rounded to the nearest representable number (as usually it happens).

(i) Which number is represented by the bit-string `10111000`?

(ii) Which is the bit-string for the number -26? And for the number 0?

(iii) How many different numbers can be represented exactly in this way? Which is the smallest and which is the largest positive number?

(iv) What are the numerical results of the differences $\left(\frac{35}{32} - \frac{33}{32}\right)$ and $\left(\frac{37}{32} - \frac{35}{32}\right)$? What does this mean for equality checks between two floating-point numbers using the equality operator `==`?

(v) Which number(s) have the largest absolute error? Which have the largest relative error in the interval between the smallest and the largest representable positive number?

(vi) Repeat task (iii) setting $b = 3$. Which role does the bias play? What happens, when you vary the bias?

Exercise 2 [*Storing multiple flags in a single variable*] (6 pts.)

This exercise is about the bitwise operators `&`, `|`, `^`, `~`, `>>` and `<<`. Try to use them extensively to understand and get used to them.

Using bitwise operators and a variable of type `unsigned char` (which consists of 8 bits), use its bits to store and retrieve the following information about a student:

- bit 6: wether the student has achieved at least 50% of the exercise points,
- bit 7: wether the student has provided two active contributions,
- bit 8: wether the student has the *klausurzulassung*,

where `1` means the student fulfills the condition. Test your code by creating a variable `s1` for a student with flags set to `100` and another variable `s2` for a student with flags set to `111`. Print corresponding messages for each flag, e.g.

```
if (/* <flag 1> */)
    printf("The student has achieved 50%% of the exercise points.\n");
else
    printf("The student has not achieved 50%% of the exercise points.\n
        ");
/* ... */
```

Hint: To make your code mor readable, dafine symbolic constants, e.g. `#define __POINTS_GT_50__ 32` or `#define __ACTIVE_FT_2__ 64`.

Exercise 3 [*Operator precedence and casts*] (6 pts.)

The details of operator precedence and casts will be discussed in the lecture on 14.05.2024.
In `C`, there are clear rules for precedence and associativity of operators. Knowing them in detail is mandatory for any scientific programmer. Set up a new Cprogram and put the following statements at the beginning of the `main` function.

```
double a, b, c, X;        int i, j, k, N;
a=1.0; b=-3.0; c=5.0;     i=5; j=-6; k=10;
```

For each of the following statements, work out which value is stored in `N` or in `X`.

(i) `X = a/b*c;`

(ii) `X = (double)i/j;`

(iii) `X = (double)(i/j);`

(iv) `N = i>k+j;`

(v) `N = i>=j && j<=k;`

(vi) `N = k/i && i/k;`

Write down, step by step, the order in which the computer executes the operations and predict the result. Afterwards, add the statement to your C code and print the result to check your prediction.

Consider now the logical operators `&&` and `||` and the ***post***-increment operators `++` and `--`. Given the following code,

```
#include <stdio.h>
int main(){
    int j=0, k=3, m;
    m = j++ || k--;
    printf("j=%d\tk=%d\tm=%d\n", j, k, m);
}
```

predict its output.