Informatik für Mathematiker und Physiker - AS18

Exercise 5: Floating Point Calculations & Basic Methods

Handout: 16. Okt. 2018 06:00

Due: 22. Okt. 2018 23:59

Task 5: Fixing Functions

Open Task (https://expert.ethz.ch/solve/84xqiHBwTMJtbXRXJ)

This task is a text based task. You do not need to write any program/C++ file: the answer should be written in main.txt *(and might include code fragments if questions ask for them).*

Task:

What are the problems (if any) with the following functions? Fix them and find appropriate pre- and post conditions.

1. function `is_even` :

```
bool is_even (unsigned int i) {
    if (i % 2 == 0) return true;
}
```

2. function `invert` :

```
double invert (double x) {
    double result;
    if (x != 0)
        result = 1 / x;
    return result;
}
```

Task 2: Point on Parabola ?

Open Task (https://expert.ethz.ch/solve/85ZDyQAW8uEh2RBFs)

Task

Consider a parabola $(\mathcal{P})$ defined as $y = g(x)$, with $g(x) = 0.9 \cdot x^2 + 1.3 \cdot x - 0.7$.

Write a program that determines if a point (x, y) lies on parabola $(\mathcal{P})$ or not. The input is provided by two decimal numbers in the sequence x, y. The program must output `yes`, if the point lies on the parabola, otherwise `no`. Use datatype `double` for all variables and numbers used in the calculation of $g(x)$.

You will notice that a straight forward approach (comparing for equality) does not work, i.e., for some points that clearly should be on parabola g such an approach returns result `no`.

Hint: Look at the difference between the exact values of the function and the values that your program calculates. Change the program so that it works properly for all points that the submission system uses as test input *without hard-coding the points*.

Input

Two decimal numbers x, y.

Example:

```
1.5 2.3
```

Output

`yes` if (x, y) is on the parabola, `no` otherwise. No other output is expected.

Example:

```
no
```

Note: Outputing both `yes` and `no` will get you past the automatic grading, but doing so counts as hard-coded solution and results in zero points.

Task 3: Rounding

Open Task (https://expert.ethz.ch/solve/LHR8JCdcnamnh9h3m)

Task

1. Implement the following rounding function that rounds a 64-bit floating point number (type `double`) to the nearest 32-bit integer (type `int`). You may assume that the type `double` complies with the IEEE standard 754. The function is only required to work correctly if the nearest integer is in the value range of the type `int`, otherwise, the return value of the function is undefined.
   Note: Usage of library rounding functions (standard or others) is not allowed.

```
// PRE:  x is roundable to a number in the value range of type @
// POST: return value is the integer nearest to x, or the one fu
//       away from 0 if x lies right in between two integers.
int round(double x);
```

2. Write a program which inputs a number of type `double` from the user, then rounds this number using your function from (1), and then outputs the rounded number.

Input

A decimal number x, such that the integer nearest to x is representable as a `int` .

Example:

```
0.5
```

Output

The integer which is nearest to `x` .

Example:

```
1
```

Task 1: Floating-Point Number Representation

Open Task (https://expert.ethz.ch/solve/LJ3BSRjwbnmdsjZfa)

This task is a text based task. You do not need to write any program/C++ file: the answer should be written in main.txt *(and might include code fragments if questions ask for them).*

Task

We examine the normalized binary representation of floating point numbers ($\beta = 2$). Your system has a precision $p = 4$ and an exponent range of $e \in [-3; 3]$.

1. Express this setting in the lecture notation $F^*(\beta, p, e_{min}, e_{max})$.
2. Convert the following decimal numbers to the *normalized binary representation*. For each number, choose the appropriate exponent e and round to the nearest value if you cannot represent the exact value.
   - 3.1416_{10}
   - 2.718_{10}
   - 7_{10}
   - 0.11_{10}
3. For each number of (2), convert the binary representation back to their decimal

form, and determine the absolute rounding error of the conversion.

4. Calculate $2.718_{10} + 3.1416_{10} + 0.11_{10}$ in the binary representation. What do you observe?

Note: To round a floating point number use binary arithmetic rounding similar to decimal arithmetic rounding, i.e. round up for a 1 and down for a 0. For example, if we round to five significant digits:

- 11.001011_2 is rounded down because the sixth significant digit is a 0 and therefore is truncated to 11.001_2.
- 11.001101_2 is rounded up because the sixth significant digit is a 1 and therefore is $11.001_2 + 0.001_2 = 11.010_2$.

Task 4: Binary Expansion

Open Task (https://expert.ethz.ch/solve/hNZrmtWgJuLfePh5s)

Task

Write a program that performs the binary expansion for a given decimal input number x, where $0 \leq x < 2$. Use the algorithm presented in the lecture. The program must output the first 16 digits of the number in the format: $b_0.\, b_1 b_2 b_3 \ldots b_{15}$. Always print all 16 digits, even the trailing zeros. Do not normalize or round the number. You can structure your program into functions to avoid code repetition. Do not forget to annotate functions with pre- and post conditions.

Input

A decimal number between 0 (inclusive) and 2 (exclusive).

Example:

```
0.75
```

Output

The 16 binary digits of the number, starting from the digit before the fractional point.

Example:

```
0.110000000000000
```