Informatik I (D-ITET)
Übungsstunde 5, 23.10.2017

Hossein Shafagh
shafagh@inf.ethz.ch
Problem 4.1. Loop mix-up

```cpp
// Fakultät
unsigned int n;
std::cin >> n;

unsigned int f = 1;
if(n != 0) {
    do {
        f = f * n;
        --n;
    } while(n > 0);
}
std::cout << f << std::endl;
```

```cpp
unsigned int n;
std::cin >> n;

unsigned int f;
for(f = 1; n > 0; --n)
    f = f * n;
std::cout << f << "\n";
```
Übungsblatt 4

- Problem 4.1. Loop mix-up

// Compute the sum

```cpp
int again;
while(true) {
    int i1, i2;
    std::cin >> i1 >> i2;
    std::cout << i1 + i2 << "\n";
    int again;
    std::cout << "Again? (0/1)\n";
    std::cin >> again;
    if(!again) break;
}
```

```cpp
int again;
do {
    int i1, i2;
    std::cin >> i1 >> i2;
    std::cout << i1 + i2 << "\n";
    std::cout << "Again? (0/1)\n";
    std::cin >> again;
} while(again);
```
Problem 4.1. Loop mix-up

// Compute Modulo

unsigned int z;
unsigned int d;

for(std::cin >> z >> d ; z >= d ; z = z-d);

std::cout << z << std::endl;

unsigned int z, d;
std::cin >> z >> d;

while(z >= d) {
    z = z - d;
}

std::cout << z << "\n";
Problem 4.2. Loop Analysis

a) \( x = 2^n \)
b) \{0, \ldots, 31\}, \( 2^{32} \rightarrow \) overflow!
c) For \( n = 0 \) the program terminates after not entering the if-block.

For \( n \in \mathbb{N} \cap [1, 31] \), \( k \) is strictly monotonic increasing from start value 0, eventually fulfilling \( k = n \) and setting the termination condition. This is independent of the value of \( x \).

```cpp
unsigned int x = 1;
if (n > 0) {
    unsigned int k = 0
    bool e = true;
    do {
        if (++k == n) {
            e = false;
        }
        x *= 2;
    } while (e);
}
std::cout << x << std::endl;
```

```cpp
unsigned int x = 1;
for (unsigned int i = 0; i < n; ++i) {
    x *= 2;
}
std::cout << x << std::endl;
```
Problem 4.3. Floats and Conversions

Determine type and value for each of the following expressions:

(a) $2e2 - 3e3f > -23.0$
   - false (boolean)

(b) $-7 + 7.5$
   - 0.5 (double)

(c) $1.0f / 2 + 1/3 + 1/4$
   - 0.5 (float)

(d) $1u - 2u < 0$
   - false (boolean)

(e) $1 + 2 \times 3 + 4$
   - 11 (int)

(f) int(8.5) − int(7.6)
   - 1 (int)
Problem 4.4. Approximation of $\pi$

double sum = 0.0;
double a = 1.0;
double b = 1.0;
for (unsigned int count = 1; n >= count; ++count)
{
    sum += (a / b);
    a *= (-1);
    b += 2;
}
cout << "sum = " << 4 * sum << endl;

// initialize auxiliary variables for first term of sum
double numer = 2.0; // numerator i-th term
double denom = 1.0; // denominator i-th term

// pi: value after term i (i=0 initially, then i=1,2,...,n-1)
double pi = 2.0;
for (int i = 1; i < n; ++i) {
    numer *= i;
    denom *= (2 * i + 1)
    pi += numer/denom;
}
Problem 4.4. Approximation of $\pi$

For $n = 1000000$, the output of sum1 is correct to five digits after the decimal point: 3.14159.

Sum2 already gives the result 3.14159 for $n = 17$, so this version is obviously preferable.

Caveat: sum2 quickly results in (on our platform after $n = 172$) terms that are so small that they cannot be represented as 64-bit (double) floating point number (NaN: not a number)
Binary Representation

Binary representation of the decimal number 1.9?

<table>
<thead>
<tr>
<th>x</th>
<th>b_i</th>
<th>x−b_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>1.8</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td><strong>1.6</strong></td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>0.8</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td><strong>1.6</strong></td>
<td>1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

1.11100
## Exercise 1) Binary Representation

a) 0.25

<table>
<thead>
<tr>
<th>x</th>
<th>b_i</th>
<th>x-b_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>1.0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>0.01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) 11.1

<table>
<thead>
<tr>
<th>x</th>
<th>b_i</th>
<th>x-b_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>0.8</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>1.6</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

1011.00011
A 10 bit floating point type

- scientific notation: \(2.73 \times 10^{12}\)
- 1 digit for decimal point
- 5 for after decimal point
- 4 for exponent
  - \(1.00110 \times 2^{15}\) or \(0.00001 \times 2^0\)

\[F(\beta, p, e_{\text{min}}, e_{\text{max}}), \beta \geq 2, p \geq 1\]

- So for our 10 bit example, \(\beta = 2\) and \(p = 6\), but what are \(e_{\text{min}}\) and \(e_{\text{max}}\)?
A 10 bit floating point type

- \( F(2, 6, e_{\text{min}}, e_{\text{max}}) \)

- IEEE 754 standard
  - unsigned int - bias
  - Bias: 8 \( \rightarrow \) \( e_{\text{min}} = 0 - 8 = -8 \) and \( e_{\text{max}} = 15 - 8 = 7 \)
  - \( F(2, 6, -8, 7) \)
    - smallest representable number: \( 0.00001 \times 2^{-8} \)
    - largest representable number: \( 1.11111 \times 2^{7} \)
A 10 bit floating point type

- \( F(\beta, p, e_{\text{min}}, e_{\text{max}}) \)
- \( F(2, 6, -8, 7) \)
  - smallest representable number: \( 0.00001 \times 2^{-8} \)
  - largest representable number: \( 1.11111 \times 2^7 \)
- Two important changes
  - One bit for sign to represent negative values
  - There is always exactly one non-zero digit in front of the decimal point! (scientific notation) \( \rightarrow \) don’t store this bit \( \rightarrow \) hidden bit
- \( F^*(2, 6, -8, 7) \)
  - \( +1.11111 \times 2^7 \) (biggest number)
  - \( -1.11111 \times 2^7 \) (smallest number)
  - \( +1.00000 \times 2^{-8} \) (smallest positive number)
A 10 bit floating point type

- \( F^*(2, 6, -8, 7) \)
  - \(+1.11111 \times 2^7\) (biggest number)
  - \(-1.11111 \times 2^7\) (smallest number)
  - \(+1.00000 \times 2^{-8}\) (smallest positive number)

- How to represent 0?

- Exceptional values:
  - exponent \(-8\) (0000) is the signal that we have an exceptional value
  - 5 bits after the decimal point:
    - 00000 means 0
    - 00001 means \(\infty\)
    - 00010 means \(-\infty\)
    - 00011 means NaN (not a number)

- This leaves us with \( F^*(2, 6, -7, 7) \)
The floating point system $F^*(2, 24, -126, 127)$

- float type’s internal representation
- for double it is $F^*(2, 53, -1022, 1023)$
- 24: 23 digits + 1 sign bit
- 8 for exponent:
  - Bias of 127
  - 0, 255 for special values:
    - 0, $\infty$, $-\infty$ and NaN (not a number)
Exercise 2) Largest and smallest positive numbers

- What are the largest and smallest positive normalized single and double precision floating point numbers, according to the IEEE 754 standard?

  - $2^{e_{\text{min}}}$
  - Single precision: $2^{-126}$
  - Double precision: $2^{-1022}$

- Largest normalized number:
  $+1.11111111111111111111111111111111 \times 2^{127}$
Exercise 2) Largest and smallest positive numbers

What are the largest and smallest positive normalized single and double precision floating point numbers, according to the IEEE 754 standard?

- Largest normalized number:
  \[ +1.11111111111111111111111111111111 \times 2^{127} \]

- More elegant: \( (1 - (1/\beta)^p) \beta^{\text{emax}+1} \)

- Single: \( (1 - (1/2)^{24}) 2^{128} = 2^{128} - 2^{104} \)

- Double: \( (1 - (1/2)^{53}) 2^{1024} = 2^{1024} - 2^{971} \)
Floating Point Guidelines
Function Definition and Declaration

// PRE: Value representing an angle expressed in radians.
// One radian is equivalent to 180/PI degrees.
// POST: Cosine of x in radians.
double cos (double x); // available in #include <cmath>

// PRE: y != 0
// POST: returns quotient x / y
double quotient (double x, double y) {
    assert (y != 0);
    return x / y; // possible PRE condition violation here
}
Function Exercises
Übungsblatt 5

- Problem 5.1. Floating-Point Number Representation
- Problem 5.2. Point on Line?
- Problem 5.3. Rounding
- Problem 5.4. Binary Expansion
- Problem 5.5. Fixing Functions