

Informatik I für D-ITET

Serie 3

WS 04/05

URL: <http://www.ti.inf.ethz.ch/ew/courses/inf1-ITET/>

Aufgabe 1 (2 Punkte)

Holen Sie sich das Programm `harakiri.C` von der Vorlesungswebseite. Übersetzen Sie das Programm und probieren Sie einige Eingaben aus (1,2,3,...). Beschreiben Sie, was passiert, und begründen Sie, wie es dazu kommt.

Aufgabe 2 (4 Punkte)

Werten Sie folgende Ausdrücke „per Hand“ in Einzelschritten und in der durch Priorität und Assoziativität bestimmten Reihenfolge aus. Nehmen Sie an, dass das Ergebnis von Fließkommaoperationen zur nächsten darstellbaren Zahl gerundet wird (*round-to-nearest*), und gehen Sie von folgenden Bitbreiten aus.

`int` 32 Bits
`float` 24 Bits (Mantisse)
`double` 53 Bits (Mantisse)

Zudem muss bei allen Zahlen der Datentyp erkennbar sein. Zum Beispiel

$$7.0f * 6 / 3 - 3.0 \rightarrow 42.0f / 3 - 3.0 \rightarrow 14.0f - 3.0 \rightarrow 11.0.$$

- a) $6 / 4 * 2.0f - 3$
- b) $2 + 15.0e7f - 3 / 2.0 * 1.0e8$
- c) $392593 * 2735.0f - 8192 * 131072 + 1.0$
- d) $16 * (0.2f + 262144 - 262144.0)$

Tips:

- Schreiben Sie die Zahlen in Aufgabenteil c) und d) binär hin.

Beispiel: $8388615 = 2^{23} + 7 = (10 \underbrace{\dots 0}_{20\text{-mal}} 111)_2$.

- $8192 = 2^{13}$, $131072 = 2^{17}$, $262144 = 2^{18}$, $1073741824 = 2^{30}$.

Aufgabe 3 (6 Punkte)

Schreiben Sie ein Programm `dual.C`, das eine Dezimalzahl x , $1 \leq x < 2$, in eine Variable vom Typ `double` einliest und die Mantisse der `double`-Darstellung ausgibt. Beispiel: Bei Eingabe 1.25 soll die Zeichenkette $1010 \underbrace{\dots 0}_{50\text{-mal}}$ ausgegeben werden.

Abgabe: Aufgabe 1 und 2: am 8. November 2004, am Anfang der Übungen, schriftlich.

Aufgabe 3: bis 15. November 2004, 12.00 Uhr, per Email.

Achtung: In der nächsten Woche wird es eine Schnellübung geben!

Institut für theoretische Informatik
Dr. J. Giesen

3. November 2004

Informatik I:

Material aus der Vorlesung

Programm: subtract.C _____

```
// Programm: subtract.C
// Teste Subtraktion zweier Fließkommazahlen

#include <iostream>

int main()
{
    // Eingabe
    std::cout << "Subtraktions-Test\nminuend =? ";
    float minuend;
    std::cin >> minuend;

    std::cout << "subtrahend =? ";
    float subtrahend;
    std::cin >> subtrahend;

    std::cout << minuend << " - " << subtrahend << " =? ";
    float result;
    std::cin >> result;

    // Berechnung des "korrekten" Ergebnisses
    float real_result = minuend - subtrahend;

    // Vergleiche beide Ergebnisse und gib
    // die Beurteilung aus:
    if (result == real_result)
        std::cout << "korrekt." << std::endl;
    else
        std::cout << "Falsch, um "
        << real_result - result
        << " daneben." << std::endl;

    return 0;
}
```

Programm: harmonic.C _____

```
// Programm: harmonic.C
// Berechnung der N-ten harmonischen Zahl auf zwei Arten.

#include <iostream>

int main()
{
    // Eingabe
    std::cout << "Welche harmonische Zahl H_N (1 <= N <= "
    << std::numeric_limits<unsigned int>::max()
    << ") ? ";

    unsigned int N;
    std::cin >> N;

    // Berechnung der harmonischen Zahl beginnend bei 1
    float sum1 = 0.0f;
    for (unsigned int i = 1; i <= N; ++i)
        sum1 += 1.0f / i;

    // Berechnung der harmonischen Zahl beginnend bei N
    float sum2 = 0.0f;
    for (unsigned int i = N; i >= 1; --i)
        sum2 += 1.0f / i;

    // Ausgabe
    std::cout << "Vorwaertssumme = " << sum1 << "\n"
    << "Rueckwaertssumme = " << sum2 << std::endl;

    return 0;
}
```

Programm: one.C _____

```
// Programm: one.C
// Berechnet etwas umstaendlich die Zahl eins.

#include <iostream>

int main()
{
    // summiere solange, bis der aktuelle Fehler error
    // kleiner wird als max_error
    // oo
    // -- 1
    // Da \ ----- = 1 , bricht die Schleife
    // / i * (i + 1)
    // --
    // i = 1
    //
    // nach endlich vielen Iterationen ab.

    float sum = 0.0f; // Partialsumme
    float error; // momentaner Fehler
    const float max_error = 0.0001f; // Fehlerschranke

    unsigned int i = 1;
    do {
        sum += 1.0f / (i * (i + 1.0f));
        ++i;
        error = 1.0f - sum;
        std::cout << "aktueller Fehler ist " << error
        << std::endl;
    } while (error > max_error);

    // Ausgabe:
    std::cout << "Summe = " << sum << ".\n"
    << "Fehler = " << error << std::endl;

    return 0;
}
```