Gruppe 5
Freitag 13:15-14:45, B-Woche
(Dr. B. Schulze)

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

Numerik

Serie 5

1. Sei $y = \frac{x_1}{x_2}$, d.h. $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ mit $f(x_1, x_2) = \frac{x_1}{x_2}$:

$$\begin{aligned} \tilde{y} &= \frac{\tilde{x_1}}{\tilde{x_2}} \\ &= \frac{x_1(1+\varepsilon_1)}{x_2(1+\varepsilon_2)} \\ &= \frac{x_1}{x_2}(1+\varepsilon_1)(1-\varepsilon_2) \\ &= \frac{x_1}{x-2}(1+\varepsilon_1-\varepsilon_2) \end{aligned}$$

bzw.

$$\begin{aligned} \frac{|\tilde{y}-y|}{|y|} &\doteq |\varepsilon_1 - \varepsilon_2| \leq |\varepsilon_1| + |\varepsilon_2| \\ &\leq 2eps \end{aligned}$$

2. a) Der Algorithmus üeberprüft zunächst ob der Exponent negativ ist. Wenn ja wird später der Kehrwert des Ergebnisses zurück gegeben. Danach berechnet der Algorithmus für jedes Bit eine Potenz der Basis. Für das erste Bit Basis b^1, für das zweite b^2 etc. Nur wenn das entsprechende Bit gleich eins ist, wird die Potenz mit dem Ergebnis multipliziert.[1]
   Der Algorithmus berechnet also pro Anzahl der Bits des Exponenten sowie pro Anzahl der Eins-Bits im Exponenten eine Multiplikation.

   b) Die Analyse des Zeitverhaltens der Algorithmen:
      i. linear, sofort ersichtlich (es handelt sich um einen iterativen Algorithmus).
      ii. fast konstant (es scheint sich hierbei um eine direkte Implementierung von iv) zu handeln.
      iii. linear zur Anzahl der Bits im Exponenten sowie der Anzahl der Eins-Bits (siehe oben). $O(n)$
      iv. konstant "hohe" Berechnungszeit aufgrund zweier konstanter Funktionsaufrufe.

[1] wir zählen die Bits von rechts

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

Hier die Ergebnisse:

```
Ausgabe:
Funktion i), n = 2
System: 0.000000 User: 0.360000
Ausgabe:
Funktion ii), n = 2
System: 0.020000 User: 2.830000
Ausgabe:
Funktion iii), n = 2
System: 0.000000 User: 0.820000
Ausgabe:
Funktion iv), n = 2
System: 0.000000 User: 4.600000
Ausgabe:
Funktion i), n = 3
System: 0.000000 User: 0.530000
Ausgabe:
Funktion ii), n = 3
System: 0.020000 User: 2.840000
Ausgabe:
Funktion iii), n = 3
System: 0.000000 User: 0.820000
Ausgabe:
Funktion iv), n = 3
System: 0.010000 User: 4.770000
Ausgabe:
Funktion i), n = 19
System: 0.030000 User: 3.030000
Ausgabe:
Funktion ii), n = 19
System: 0.020000 User: 2.950000
Ausgabe:
Funktion iii), n = 19
System: 0.000000 User: 1.320000
Ausgabe:
Funktion iv), n = 19
System: 0.020000 User: 4.760000
Ausgabe:
Funktion i), n = 233
System: 0.210000 User: 35.590000
Ausgabe:
Funktion ii), n = 233
System: 0.050000 User: 3.290000
Ausgabe:
Funktion iii), n = 233
System: 0.010000 User: 1.800000
Ausgabe:
Funktion iv), n = 233
System: 0.030000 User: 4.770000
```

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

... und hier der Sourcecode:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <sys/times.h>
#include <unistd.h>

#ifdef CLOCKS_PER_SEC
#undef CLOCKS_PER_SEC
#endif

#define CLOCKS_PER_SEC sysconf(_SC_CLK_TCK)


double powi(register double base, register int exponent)
{
  register int negative;
  register double power;


  negative= exponent < 0;
  if (negative)
    exponent= -exponent;
  power= 1.0;
  while (exponent)
  {
    if (exponent&1)
      power*=base;
    exponent>>=1;
    if (!exponent)
      break;
    base*=base;
  }
  if (negative)
    return 1.0/power;
  else
    return power;
}



void test_of_time (int n) {
  int     i,j;
  struct  tms time1[2];
  double  base = 10,y;
  clock_t erg1,erg2;


  times(&(time1[0]));
```

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

```
for(i= 0; i < 10000000; i++) {
  y= base;
  for(j= 1; j < n; j++) y*= base;
}
times(&(time1[1]));
printf("Ausgabe:\n");
printf("Funktion i), n = %d\n",n);
erg1= time1[1].tms_stime - time1[0].tms_stime;
erg2= time1[1].tms_utime - time1[0].tms_utime;
printf("System: %f User: %f\n",
  ((double)erg1)/CLOCKS_PER_SEC, ((double)erg2)/CLOCKS_PER_SEC);

times(&(time1[0]));
for(i= 0; i < 10000000; i++) {
  y= pow(base, n);
}
times(&(time1[1]));
printf("Ausgabe:\n");
printf("Funktion ii), n = %d\n", n);
erg1= time1[1].tms_stime - time1[0].tms_stime;
erg2= time1[1].tms_utime - time1[0].tms_utime;
printf("System: %f User: %f\n",
  ((double)erg1)/CLOCKS_PER_SEC, ((double)erg2)/CLOCKS_PER_SEC);

times(&time1[0]);
for(i= 0; i < 10000000; i++) {
  y= powi(base, n);
}
times(&(time1[1]));
printf("Ausgabe:\n");
printf("Funktion iii), n = %d\n",n);
erg1= time1[1].tms_stime - time1[0].tms_stime;
erg2= time1[1].tms_utime - time1[0].tms_utime;
printf("System: %f User: %f\n",
  ((double)erg1)/CLOCKS_PER_SEC, ((double)erg2)/CLOCKS_PER_SEC);

times(&(time1[0]));
for(i=0; i < 10000000; i++) {
  y= exp(n * log(base));
}
times(&(time1[1]));
printf("Ausgabe:\n");
printf("Funktion iv), n = %d\n",n);
erg1= time1[1].tms_stime - time1[0].tms_stime;
erg2= time1[1].tms_utime - time1[0].tms_utime;
printf("System: %f User: %f\n",
  ((double)erg1)/CLOCKS_PER_SEC, ((double)erg2)/CLOCKS_PER_SEC);
```

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

```
  return;
}

int main() {
  test_of_time(2);
  test_of_time(3);
  test_of_time(19);
  test_of_time(233);

  return EXIT_SUCCESS;
}
```