C Programmier-übungen

1

Kontrollstrukturen & Algorithmenentwicklung

- Summe 1..100
- Zinseszinzberechnung
- Programm 1x1.c
- 1 x 1 (mit for-Schleife)
- Bestimmung von Pi mittels Monte-Carlo Methode
- Annuitätentilgung
- Ostertagsberechnung, Codierung
- Zahlengeneratoren
- Skalieren von Zahlen
- Mondlandung
- sin-/cos-Tabelle
- Zahlentest
- Vokale im Text zählen
- Primzahltest
- Reihenentwicklung (sin)
- Reihenentwicklung (cos)

Summe 1..100

- Erstellen Sie ein Programm, welches die Zahlen von 1 bis 100 aufsummiert und diese Gesamtsumme dann ausgibt.
  - **Verwenden Sie hierfür eins der Schleifenkonstrukte von C**
    (Wir nehmen 'mal an, wir haben vergessen, daß es hierfür seit Gauß eine Formel gibt: n*(n+1)/2)

```
#include <iostream>
using namespace std;

void main()
{
   int    summe;
   summe =    1+ 2+ 3+ 4+ 5+ 6+ 7+ 8+ 9+10+
             11+12+13+14+15+16+17+18+19+20+
                    ...
             91+92+93+94+95+96+97+98+99+100;

   cout << "Gesamtsumme: " << summe;
}
```

```
#include <iostream>
using namespace std;

void main()
{
   int    summe = 0;
   int    i;

   for (i=1; i<= 100; i=i+1)
      summe = summe + i;

    cout << "Gesamtsumme: " << summe;
}
```


Zinseszinzberechnung

- Erstellen Sie ein Programm, welches nach Eingabe des zu verzinsenden Betrags, des Zinssatzes und der Laufzeit den sich ergebenden Betrag ausgibt

```
void main()
{
    double  Betrag, Zins;
    int     i, Laufzeit;

    cout << "Eingabe von Betrag, Zinssatz und Laufzeit: ";
    cin >> Betrag >> Zins >> Laufzeit);

    Zins = 1.0 + Zins/100.0;

    for (i=1; i <= Laufzeit; i++)
        Betrag = Betrag*Zins;

    cout << "Betrag nach %d Jahren: " << Laufzeit
         << Betrag;
}
```

Programm 1x1.c

- Schreiben Sie ein Programm, welches alle Kombinationen des kleinen 1x1 auf dem Bildschirm ausgibt

```
#include <iostream>
using namespace std;

void main()
{
   int x, y;

   x = 1;
   while (x <= 9) {
      y = 1;
      while (y <= 9) {
         cout << x << "x" << y << "="
              << setw(2) << x*y);
         y = y+1;
      }
      cout << endl;
      x = x+1;
   }
}
```

```
1x1= 1  1x2= 2  1x3= 3  1x4= 4  1x5= 5  1x6= 6  1x7= 7  1x8= 8  1x9= 9
2x1= 2  2x2= 4  2x3= 6  2x4= 8  2x5=10  2x6=12  2x7=14  2x8=16  2x9=18
3x1= 3  3x2= 6  3x3= 9  3x4=12  3x5=15  3x6=18  3x7=21  3x8=24  3x9=27
4x1= 4  4x2= 8  4x3=12  4x4=16  4x5=20  4x6=24  4x7=28  4x8=32  4x9=36
5x1= 5  5x2=10  5x3=15  5x4=20  5x5=25  5x6=30  5x7=35  5x8=40  5x9=45
6x1= 6  6x2=12  6x3=18  6x4=24  6x5=30  6x6=36  6x7=42  6x8=48  6x9=54
7x1= 7  7x2=14  7x3=21  7x4=28  7x5=35  7x6=42  7x7=49  7x8=56  7x9=63
8x1= 8  8x2=16  8x3=24  8x4=32  8x5=40  8x6=48  8x7=56  8x8=64  8x9=72
9x1= 9  9x2=18  9x3=27  9x4=36  9x5=45  9x6=54  9x7=63  9x8=72  9x9=81
```

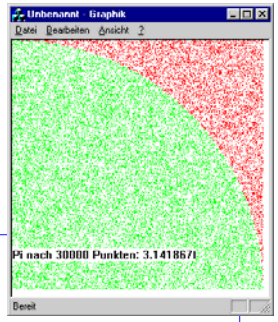

1 x 1 (mit for-Schleife)

```
#include <iostream>

void main()
{
   int x, y;

   x = 1;
   while (x <= 9) {
      y = 1;
      while (y <= 9) {
         cout << x << "x" << y << "="
              << setw(2) << x*y);
         y = y+1;
      }
      cout << endl;
      x = x+1;
   }
}
```

```
void main()
{
   int  x, y;

   for (x=1; x<=9; x=x+1) {
      for (y=1; y<=9; y=y+1)
         cout << x << "x" << y << "="
              << setw(2) << x*y);
      cout << endl;
   }
}
```

Bestimmung von Pi mittels Monte-Carlo Methode

- Würfeln Sie zwei Zahlen x und y im Bereich von [0.0..1.0]
- (x,y) = Koordinate im Einheitsquadrat
- Test: Liegen (x,y) auch im Einheitskreis?
  (Test mittels Abstandsbestimmung~Pythagoras)
- Verhältnis der Punkte innerhalb zur Gesamtanzahl = PI/4




```
#include <iostream>
#include <stdlib.h>
using namespace std;
void main()
{
   int    in = 0, total = 0;
   int    i;
   double x,y;

   for (i=0; i<500000; i++)
   {
      x = (double) rand() / (double) RAND_MAX;
      y = (double) rand() / (double) RAND_MAX;
      if (x*x + y*y <= 1.0)
         in = in +1;
      total = total + 1;
   }

   cout << "Pi nach " << i << " Punkten: "
        << 4.0*in / total;
}
```

```
Pi nach 500000 Punkten: 3.141160
```


Annuitätentilgung

- Einem Bankkunden soll ein Tilgungplan für ein beantragtes Darlehen erstellt werden. Es wurde eine Annuitätentilgung vereinbart. Weitere Darlehenskenngrößen:

  Darlehen: 100.000 €, Zins: 5% p.a. (eff), Tilgung: 3% p.a.
  (Bitte lassen Sie diese Größen über die Tastatur abfragen!)

- Geben Sie bitte eine Tabelle aus, in der jeweils in einer Zeile die folgenden Daten stehen, bis das Darlehen vollständig zurückgezahlt wurde.

  Lfd. Jahr | Restdarlehen | Zinsen im Jahr | Tilgung im Jahr

- Bitte berechnen Sie auch den Gesamtbetrag, der zurückbezahlt werden muß (also Darlehen+Zinsen) und die monatliche Belastung

  Hilfestellung (Annuitätentilgung)

  - **Annuität (Jahresrate) = Darlehen * (Zins+Tilgung) / 100**
    (Diese Rate bleibt über die gesamte Laufzeit konstant!)
  - **Zinsen im lfd. Jahr = Restdarlehen * Zins/100**
  - **Tilgung im lfd. Jahr = Annuität – Zinsen pro Jahr**

- Bitte probieren Sie aus, welche Auswirkungen eine Variierung der Tilgungsrate auf 1% bzw. 5% auf Laufzeit und Gesamtrückzahlungsbetrag hätte.

Annuitätentilgung

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
   double    darlehen;          // Darlehensbetrag
   double    zins;         // Zinssatz in %, z.B. 5%
   double  tilgung;        // Tilgungssatz in %

   double  jrate;          // Jaehrliche Tilgungsrate
   double  ZinsenImJahr; // Im Jahr zu zahlende Zinsen
   double  summezins = 0;  // Sum.d.Zinszahlungen
   int     lfdJahr = 1;

   cout << "Annuitaetentilgung" << endl;
   cout << "Darlehensbetrag:  ");
   cin >> darlehen;
   cout << "Zins:             ";
   cin >> zins;
   cout << "Tilgung:          ";
   cin >> tilgung;

   jrate = darlehen * (zins+tilgung)/100.0;

   while (darlehen > 0)
   {   // Solange ein Restdarlehen vorhanden ...
       ZinsenImJahr = darlehen*zins/100.0;
       summezins = summezins + ZinsenImJahr;
       tilgung = jrate-ZinsenImJahr;
       cout << setw(2) << lfdJahr << ":  " << setw(10)
           << darlehen << " | " << setw(8) ZinsenImJahr
           << " | << setw(8) << tilgung;
       lfdJahr  = lfdJahr + 1;
       darlehen = darlehen - tilgung;
   }
    cout << "\nSumme der Zinsen:     " << setw(10)
          <<  summezins);
    cout << "Monatliche Belastung:   " << setw(10)
          << jrate/12.0);
}
```


Ostertagsberechnung, Codierung

http://www.ptb.de/deutsch/org/4/43/432/oste.htm

Wann ist Ostern? (Ausdruck obiger Web-Seite)

- Ostern wird nach langer christlicher Tradition am ersten Sonntag nach dem ersten Vollmond im Frühling (auf der nördlichen Halbkugel) gefeiert.

  Die Bestimmung des kalendarischen Datums ist Bestandteil der grundlegenden Arbeit von Christopher Clavius zur Kalenderreform Papst Gregors XIII gewesen. Carl Friedrich Gauß (1777 - 1855) hat für die Datumsberechnung eine Vorschrift angegeben, die hier jedoch in der modifizierten Form von Dr. Heiner Lichtenberg, Bonn (H. Lichtenberg, *Zur Interpretation der Gaußschen Osterformel und ihrer Ausnahmeregeln*, Historia Mathematica 24 (1997), S. 441 - 444) angegeben wird.

- In den nachfolgenden Gleichungen bedeuten INT(a/b) der ganzzahlige Teil des Quotienten a/b und MOD (a,b) der nicht-negative Rest den a beim Teilen durch b läßt. Zu berechnen ist für die Jahreszahl X :

```
1. K = INT (X / 100);
2. M = 15 + INT ((3 · K + 3) / 4) -
       INT ((8 · K + 13) / 25);
3. S = 2 - INT ((3 · K + 3) / 4);
4. A = MOD (X,19);
5. D = MOD (19 · A + M, 30);
6. R = INT (D / 29) +
       (INT(D/28) - INT(D/29)) · INT (A / 11);
7. OG = 21 + D - R;
8. SZ =  7 - MOD (X + INT (X / 4) + S, 7);
9. OE =  7 - MOD (OG - SZ, 7);
```

- Dann ist OS = OG + OE das Datum des Ostersonntags, als Datum im Monat März dargestellt. Der 32. März entspricht dem 1. April usw.
- Liegt der Ostertermin (Os) erst einmal fest, so berechnen sich daraus weitere besondere Kalenderdaten, und zwar
  - Os-46: Aschermittwoch, Os+49: Pfingstsonntag, Os+39: Christi Himmelfahrt, Os+60: Fronleichnam

Ostertagsberechnung, Codierung (2)

```
1. K = INT (X / 100);
2. M = 15 + INT ((3 · K + 3) / 4) -
       INT ((8 · K + 13) / 25);
3. S = 2 - INT ((3 · K + 3) / 4);
4. A = MOD (X,19);
5. D = MOD (19 · A + M, 30);
6. R = INT (D / 29) +
       (INT(D/28) - INT(D/29)) · INT (A / 11);
7. OG = 21 + D - R;
8. SZ =  7 - MOD (X + INT (X / 4) + S, 7);
9. OE =  7 - MOD (OG - SZ, 7);
```

```
#include <iostream>
using namespace std;
void main()
{
   int X, K, M, S, A, D, R, OG, SZ, OE, OS;
   cout << "Bitte Jahr angeben: ";
   cin >> X;
   K = X / 100;
   M = 15 + (3 * K + 3) / 4 -  (8 * K + 13) / 25;
   S = 2 - (3 * K + 3) / 4;
   A = X%19;
   D = (19 * A + M) % 30;
   R = D/29 + (D/28 - D/29) * (A / 11);
   OG = 21 + D - R;
   SZ =  7 - (X + X/4 + S) % 7;
   OE =  7 - (OG - SZ) % 7;
   if (OG+OE < 32)
       cout <<"Ostern ist im Jahr " << X " am " << OG+OE
            << ". Maerz" << endl;
   else    cout <<"Ostern ist im Jahr " << X " am "
               << OG+OE-31 << ". April" << endl;
}
```

- Erweiterung: Lassen Sie die Ostertermine für die Jahre von 1990-2010 ausgeben

Zahlengeneratoren

- Geben Sie kleine Programmfragmente an, die immer wiederkehrend folgende Zahlenfolgen bilden
  - +1, -1, +1, -1, ...
  - 0, 1, 0, 1, ...
  - 0, 1, 2, 3, 0, 1, 2, 3, ...
  - 3, 4, 5, 6, 3, 4, 5, 6, ...

```
int a = 1;
while (...)
   a = -a;
```

```
int b=0;
while (...)
   b = 1-b;
```

```
int c=0;
while (...)
   c = (c+1) % 4;
```

```
int d=3;
while (...)
   if (d == 6)      bzw. d = (d!=6) ? 3 : d+1;
      d = 3;
   else
      d = d+1;
```

Skalieren von Zahlen

- Sie möchten gerne eine (Bibliotheks-)funktion verwenden, die Gleitkommawerte berechnet. Leider stimmt das Zahlenintervall, welches die Funktion liefert, für Ihre Anwendung nicht ganz, Sie müssen konvertieren

```
[0.0 .. 1.0]    -->   [2.0 .. 3.0]
[0.0 .. 1.0]    -->   [0.0 .. 10.0]
[0.0 .. 1.0]    -->   [5.0 .. 10.0]
[2.0 .. 8.0]    -->   [-2.0 .. +2]
[0.0 .. 360.0]  -->   [0.0 .. 2*PI]
[0.0 .. 2*PI]   -->   [0.0 .. 360.0]
[-PI .. +PI]    -->   [-180.0 .. +180.0]
```

Geben Sie bitte jeweils die Abbildungsvorschrift an

```
[0.0 .. 1.0]    -->   [2.0 .. 3.0]
  ➢    f = f+2.0;
```

```
[0.0 .. 1.0]    -->   [0.0 .. 10.0]
  ➢    f = 10.0*f;
```

```
[0.0 .. 1.0]    -->   [5.0 .. 10.0]
  ➢    f = 5.0*f + 5.0;
```

```
[2.0 .. 8.0]    -->   [-2.0 .. +2]
  ➢    f = (f-2.0) / 6.0 * 4.0 - 2.0;
```

```
[0.0 .. 360.0]  -->   [0.0 .. 2.0*PI]
  ➢    f = f/360.0 * 2.0*PI
```

```
[0.0 .. 2.0*PI] -->   [0.0 .. 360.0]
  ➢    f = f/(2.0*PI) * 360.0
```

```
[-PI .. +PI]    -->   [-180.0 .. +180.0]
  ➢    f = f/PI * 180.0
```


Mondlandung

- Entwickeln Sie eine einfache Simulation einer Mondlandung. Nehmen Sie an, Ihre Raumfähre befindet sich in 50.000m Höhe über der Mondoberfläche und hat eine Abwärtsgeschwindigkeit von derzeit 3600km/Std = 1000 m/s.
- Wenn Sie keinen Schub geben, so werden Sie mit $1.63 m/s^2$ vom Mond angezogen. Die Antriebe Ihrer Raumfähre vermögen bei vollem Schub jedoch eine Gegenbeschleunigung von $12 m/s^2$ zu erzeugen und verbrauchen dabei pro Sekunde 100 Liter Treibstoff. Anfangs verfügen Sie über 10.000 Liter Treibstoff.
- Ihre Raumfähre kann eine Aufsetzgeschwindigkeit von 10m/s aushalten, ansonsten ist das Landemanöver gescheitert. Hierfür ist eine entsprechende Meldung auszugeben.
- **Einige Hilfestellungen**

  Es gelten folgende Formeln

  $v = a * t$ (v - Geschwindigkeit, a-Beschleunigung, t-Zeit)

  $s = v * t$ (s - zurückgelegter Weg bei konst. Geschwindigk.)

  $s = \frac{1}{2} a t^2$ (zurückgelegter Weg bei konst. Beschleunigung)

```
Beispiel:
 Hoehe: 33007   Geschwindigkeit: -866   Resttreibstoff: 10000
        Zeit Aktion (0-Fallen, 1-Schub) 10 0
 Hoehe: 24267   Geschwindigkeit: -882   Resttreibstoff: 10000
        Zeit Aktion (0-Fallen, 1-Schub) 10 0
 Hoehe: 15363   Geschwindigkeit: -899   Resttreibstoff: 10000
        Zeit Aktion (0-Fallen, 1-Schub) 20 1


---------------
Aufsetz-/Aufprallgeschwindigkeit: -658.533333
So ein Bruchpilot... Noch einmal ueben
```

Mondlandung

```
#include <iostream>
void main()
{
   double  zeit  =  1.0;       // Zeitaufloesung
   double hoehe  =  50000.0;   // Höhe über Grund
   double v      = - 1000.0;   // Geschwindigkeit
   double a      =  0.0;       // Beschleunigung
   double treibstoff = 10000.0; // Treibstoffvorrat
   int    schub  = 0;           // Treibstoffstoss?
   while (hoehe > 0.0)   {
      cout <<"\nH: " hoehe << "  v: " << v << " Tank: "
           << treibstoff;
      schub = 0;
      if (treibstoff > 0) {
         cout <<"\n\tZeit Aktion (0-Fallen, 1-Schub) ";
         cin >> zeit >> schub;
      }
      else
         cout << endl;
      if (schub) {
         if (treibstoff-zeit*100 < 0)
            zeit = treibstoff/100.0;
         treibstoff = treibstoff - zeit*100.0;
         a = 12.0;
      }
      else
         a = - 1.63;
      hoehe = hoehe + v*zeit + 0.5*a*zeit*zeit;
      v     = v + a * zeit;
   }
   cout <<"\n\nAufsetzgeschwindigkeit: " << v
        << endl;;
   if (v < -10.0)
      cout << "So ein Bruchpilot... " << endl;;
   else
      cout << "Herzlichen Glueckwunsch!" << endl;
}
```


sin-/cos-Tabelle

- Erstellen Sie ein Programm, welches für den Wertebereich [0, ... 360 Grad] in der Schrittweite 10 Grad eine Tabelle von sin- und cos-Werten mit jeweils 3-stelliger Genauigkeit nach dem Komma ausgibt. Die Ausgabe soll folgendes Format haben:
  wert | sin(wert) | cos(wert)
- Hinweis: Für die Verwendung von sin und cos wird die Include-Datei math.h benötigt. sin und cos erwarten als Argument einen Winkelwert im Bogenmaß, also [0, .. 2*Pi]

```
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

void main()
{
   const double Pi=3.1415;

   cout << setprecision(4);
   for (double x=0.0; x<= 360.0; x=x+10.0)
   {
      cout << x
         << " | sin=" << sin(x/360.0*2.0*Pi)
         << " | cos=" << cos(x/360.0*2.0*Pi)
         << endl;
   }
}
```

Zahlentest

- Erstellen Sie ein Programm, welches alle dreistelligen Zahlen ausgibt, die durch alle ihre Ziffern teilbar sind (Beispiel 126 ist durch 1, 2 und 6 teilbar)

```
void main()
{
   int x,y,z, zahl;

   for (x=1; x<10; x=x+1)
   {
      for (y=1; y<10; y= y+1)
      {
         for (z=1; z<10; z=z+1)
         {
            zahl = 100*x + 10*y+ z;
            if (zahl % x == 0 &&
                zahl % y == 0 &&
                zahl % z == 0    )
               cout << zahl << " ";
         }
      }
   }
   cout << endl;
}
```