Anmerkungen

- Sichern Sie ihre Berechnungen, so weit wie möglich gegen fehlerhafte Eingaben ab.
- Verwenden Sie sprechende Variablennamen und formatieren Sie Ihren Programmtext ordentlich.
- Erstellen Sie Inputprompts und Ausgaben, sodass das Programm für BenutzerInnen angenehm zu bedienen ist.
- Schreiben Sie Ihre Programm so, dass mehrere Berechnungen ausgeführt werden können, ohne das Programm immer wieder neu starten zu müssen.
- Testen Sie Ihre Programme und unterziehen Sie die Ergebnisse einer Prüfung (Schätzung des erwarteten Ergebnisses, bzw. im Notfall mit dem Taschenrechner prüfen).
- Mit * markierte Beispiele sind etwas anspruchsvoller
- Die Aufgaben 2 bis 19 behandeln Ausdrücke und Typumwandlungen, die Aufgaben 20 bis31 Schleifen.

Aufgabe 1 *

Lösen Sie die Rätselaufgabe aus der Übungseinheit so vollständig wie möglich.

Aufgabe 2

Schreiben Sie ein Programm, das zwei Zahlen m und n einliest und die n-te Ziffer der Zahl m ausgibt.
Z.B.: m=1358 n=2 Ausgabe: 5 (Die Ziffern werden ausgehend vom Dezimalpunkt gezählt).

Aufgabe 3

Schreiben Sie ein Programm, das eine dreistellige Zahl einliest und mit Hilfe der Ziffernsumme überprüft, ob die eingegebene Zahl durch 3 teilbar ist.

Aufgabe 4

Schreiben Sie ein Programm, das eine dreistellige Zahl einliest und die Zahl ausgibt, die durch Umkehrung der Ziffernfolge entsteht.
Z.B.: Eingabe: 123 Ausgabe: 321

Aufgabe 5

m Arbeiter können x Stück eines Produktes in einer Stunde fertigen. Wieviel Stück können n Arbeiter in der gleichen Zeit herstellen?

Für diese einfache Schlussechnung hat ein (unbedarfter) Programmierer folgendes Programm geschrieben:

```
#include <iostream>

using namespace std;

main() {

  int Arbeiter, Stueck, ArbeiterFrage;

  cin >> Arbeiter >> Stueck >> ArbeiterFrage;

  cout << Stueck/Arbeiter*ArbeiterFrage;

}
```

Nun erhält er aber für m=10 x=5 und n=2 die (mathematisch) falsche Antwort 0. Schlagen Sie zwei verschiedene Möglichkeiten vor, den auszugebenden **Ausdruck** so zu verändern, dass die Reihenfolge der Operationen unverändert bleibt, das Programm aber (für beliebige Eingaben) das korrekte Ergebnis liefert. Beweisen Sie das, indem Sie ein entsprechend verbessertes Programm implementieren.

Aufgabe 6

Lesen Sie die Koordinaten zweier Punkte im R^3 ein und berechnen Sie den Abstand der beiden Punkte voneinander.

Aufgabe 7

Berechnen Sie ax^2+bx+c mit Hilfe des Horner-Schemas für beliebige Eingaben von a,b,c und x.

Aufgabe 8

Schreiben Sie ein Programm zur Umrechnung von Schilling in Dollar und umgekehrt. Geben Sie das Ergebnis formatiert in Euro und Cent bzw. Dollar und Cent aus. (Betragsteile kleiner als ein Cent können Sie ignorieren.

Aufgabe 9

Schreiben Sie ein Programm zur Umrechnung von Sekunden in die entsprechende Anzahl von Jahren, Tagen, Stunden, Minuten und Sekunden und umgekehrt. (Gehen Sie davon aus, dass ein Jahr immer 365 Tage hat. Schaltjahre bleiben also unberücksichtigt.

Aufgabe 10

Eine etwas verbissene Roulettespielerin setzt immer einen fixen Betrag auf die Zahl 0. Das macht sie so lange, bis diese Zahl endlich geworfen wird. Schreiben Sie ein Programm, das den Betrag und die Anzahl der benötigten Versuche einliest und daraus den Gewinn bzw. den Verlust der Roulettespielerin berechnet.

Aufgabe 11

Ein Bierversand verkauft Fässer mit jeweils n Liter Fassungsvermögen. Die Kunden können beliebige Mengen in ganzen Litern bestellen. Schreiben Sie ein Programm, das n und eine Bestellmenge einliest und die Anzahl der benötigten Fässer berechnet. (Verwendung einer if-Anweisung ist erlaubt.

Aufgabe 12

Lesen Sie eine double Zahl ein und bestimmen Sie, ob der Benutzer eine ganze Zahl eingegeben hat oder nicht.

z.B. Eingabe: 3.7 Ausgabe: Keine ganze Zahl. Eingabe: 5 Ausgabe: Eine ganze Zahl.

Aufgabe 13

Lesen Sie zwei ganze Zahlen ein und ermitteln Sie, ob die zuletzt eingegebene Zahl ein Teiler der zuerst eingegebenen Zahl ist.

z.B. Eingabe: 7 3 Ausgabe: 3 teilt 7 nicht. Eingabe: 15 5 Ausgabe: 5 teilt 15.

Aufgabe 14 *

Schreiben Sie einen **Ausdruck** (also keine Anweisung oder Ähnliches), der eine positive Zahl x kaufmännisch korrekt rundet (also aufrunden ab 0.5 und abrunden darunter).

Aufgabe 15 *

Berechnen Sie den Ausdruck 2147483647+1 jeweils unter Verwendung der Datentypen int, unsigned, long und double. Vergleichen und interpretieren Sie die Ergebnisse.

Aufgabe 16 *

Ein Bierversand verkauft Fässer mit jeweils n Liter Fassungsvermögen. Die Kunden können beliebige Mengen in ganzen Litern bestellen. Schreiben Sie ein Programm, das n und eine Bestellmenge einliest und die Anzahl der benötigten Fässer berechnet. (Verwendung einer anderen Anweisung (if, switch, Schleife) oder einer bedingten Auswertung ist nicht erlaubt.

Aufgabe 17

Sortieren Sie 3 eingelesene double Zahlen ohne Verwendung von logischen Operatoren.

Aufgabe 18

Schreiben Sie ein Programm, das eine vierstellige ganze Zahl einliest und ihre sprachliche Repräsentation ausgibt.

z.B. Eingabe 1723

Ausgabe eins-sieben-zwei-drei.

Aufgabe 19

Schreiben Sie ein Programm zur Lösung der quadratischen Gleichung $x^2+px+q=0$ (Achtung: eventuelle komplexe Lösungen beachten)

Anmerkung: In der Klassenbibliothek cmath befindet sich die Funktion sqrt(), die zur Berechnung der Quadratwurzel dient. Der übergebene Wert sollte nicht kleiner als 0 (Null) sein, sonst wird ein Fehler ausgelöst.

Aufgabe 20

Berechnen Sie die Fakultät von n (n!)mit Hilfe einer Schleife.

Aufgabe 21

Erstellen Sie die Additionstabelle und die Multiplikationstabelle modulo n. Die Ausgabe für n=4 könnte zum Beispiel so aussehen:

Addition:

0 1 2 3

1 2 3 0

2 3 0 1

3 0 1 2

Multiplikation:

0 0 0 0

0 1 2 3

0 2 0 2

0 3 2 1

Aufgabe 22

Lesen Sie 2 Intervallgrenzen und danach beliebig viele Zahlen (Abschluss mit EOF) ein, und geben Sie aus, wie viele Zahlen kleiner als die untere Intervallgrenze, wie viele im Intervall und wie viele größer als die obere Intervallgrenze waren.

Aufgabe 23

Überprüfen Sie, ob die Folge

$n_{i+1} = n_i/2$ für gerade n_i

$n_{i+1} = 3\ n_i + 1$ für ungerade n_i

für unterschiedliche (beliebige) positive Startwerte n_1 immer den Wert 1 erreicht.

Aufgabe 24

Bestimmen Sie den Wert

$$\sum_{i=1}^{n} i^i$$

(also $1^1+2^2+3^3+...+n^n$ für eine beliebige natürliche Zahl n).

Die Verwendung von mathematischen Funktionen (wie z.B. pow aus cmath) ist nicht erlaubt.

Aufgabe 25

Lesen Sie beliebig viele Zahlen (Abschluss mit EOF) ein, und geben Sie das Maximum, das Minimum und den Mittelwert der eingegebenen Werte aus.

Aufgabe 26

Schreiben Sie ein Programm, das einen ganz simplen Taschenrechner simuliert. Eingegeben wird ein mathematischer Ausdruck, der Zahlen und die binären Operatoren +,-,* und / enthält. Der Ausdruck wird durch ein Rufzeichen abgeschlossen. Das Programm soll das Ergebnis des Ausdrucks berechnen, wobei die Operationen strikt von links nach rechts durchzuführen sind. (Das heißt, es gilt **nicht** Punkt- vor Strichrechnung!)

z.B. Eingabe: 2+3-4/2*3-1! Ausgabe: 0.5

Aufgabe 27

Schreiben Sie ein Programm, das eine natürliche Zahl einliest und die größte Zweierpotenz ausgibt, die die eingelesene Zahl teilt.

Eingabe	Ausgabe
17	1
48	16
1024	1024

Aufgabe 28

Eine natürliche Zahl heißt perfekt, wenn die Summe ihrer echten Teiler (das sind alle Teiler, die kleiner als die Zahl selbst sind) gleich groß ist, wie die Zahl. Zum Beispiel ist 6 eine perfekte Zahl, da 6=1*2*3 und 1+2+3=6. Falls die Summe der echten Teiler kleiner ist als die Zahl, heißt die Zahl defizient. Falls die Summe der echten Teiler größer ist als die Zahl, heißt die Zahl abundant. Schreiben Sie ein Programm, das eine natürliche Zahl einliest und ausgibt, ob die eingelesene Zahl perfekt, defizient oder abundant ist.

Eingabe	Ausgabe	Begründung
5	defizient	1 ist einziger echter Teiler
6	perfekt	1+2+3=6
12	abundant	1+2+3+4+6=16>12

Aufgabe 29

Eine natürliche Zahl heißt fast perfekt, wen die Summe ihrer echten Teiler (das sind alle Teiler, die kleiner als die Zahl selbst sind) um eins kleiner ist, als die Zahl. Zum Beispiel ist 4 fast perfekt, da 4=1*2*2 und 1+2=3=4-1. Ermitteln Sie alle perfekten Zahlen im Bereich 0 bis 10^7.

Aufgabe 30

Schreiben Sie ein Programm, das eine ganze Zahl einliest und die Anzahl der Inversionen in der eingegebenen Zahl ausgibt. Als Inversionen für eine bestimmte Ziffer (Ausgangsziffer) der Zahl bezeichnen wir alle Ziffern, die in der Zahl nach der Ausgangsziffer stehen und kleiner sind als diese. Zum Beispiel:

Eingabe: 53278 Ausgabe: 3

Da 3 hinter 5 steht, 2 hinter 5 steht und 2 hinter 3 steht.

Aufgabe 31 *

Wie Beispiel 18, allerdings für beliebige reelle Zahlen.

z.B. Eingabe 17.23

Ausgabe: eins-sieben-komma-zwei-drei.