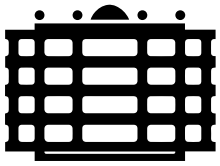


TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

Fakultätsrechen-
und Informationszentrum

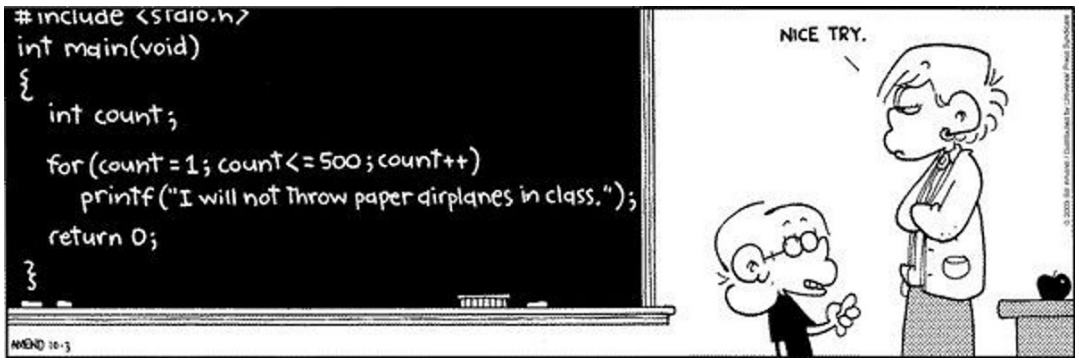
Übungsaufgaben
zur LV Grundlagen der Informatik I

Dieses Dokument wurde erstellt mit

LaTeX

Ausgabe vom 21. September 2018

Inhaltsverzeichnis



Ein besonderer Dank gilt allen an der Erstellung der Aufgaben beteiligten Personen.

Einige Musterlösungen sind online verfügbar:
`http://www.tu-chemnitz.de/informatik/friz/Grundl-Inf`

1 Grundlagen

1.1 Was ist ein Zahlensystem? Welche Zahlensysteme gibt es? Welche werden in den Rechnersystemen verwendet und warum?

1.2 Konvertieren Sie folgende Zahlen in Dezimalschreibweise: 1011_2, 101010_2, $1FF_{16}$, $A7_{16}$

1.3 Konvertieren Sie folgende Zahlen in Binärschreibweise: 42_{10}, 324_{10}, $BC5_{16}$, $7D_{16}$

1.4 Konvertieren Sie folgende Zahlen in Hexadezimalschreibweise: 100011_2, 11011_2, 255_{10}, 256_{10}

1.5 Addieren Sie folgende Zahlen in der Binärdarstellung: $11101_2 + 1110_2$ und $101001_2 + 11001_2$

1.6 Welcher Bereich von natürlichen Zahlen kann mit 8 Bit dargestellt werden? Was passiert, wenn man zu der größten mit 8 Bit darstellbaren Zahl noch 1 dazu addiert (es stehen weiterhin nur 8 Bit zur Verfügung)?

1.7 Überlegen Sie sich, im Bezug auf Aufgabe 1.6, eine Möglichkeit negative Zahlen darzustellen.

2 Einfache Berechnungen

Mit dem Lösen der folgenden Aufgaben erarbeiten Sie sich ein grundlegendes Verständnis für den Aufbau von C++ Programmen.

Im ersten Schritt sollten Sie sich mit dem Programm *HalloWelt* auseinandersetzen. Tipp: Eine Google-Suche danach findet viele Einstiegsmöglichkeiten in die Welt der Programmierung.

Darauf aufbauend erweitern Sie Ihr Wissen mit den Aufgaben 2.1 bis 2.6 in kleinen Schritten, bis Sie in der Lage sind, Programme zu schreiben, die einfache Berechnungen durchführen.

2.1 Schreiben Sie ein C/C++ Programm, die vollständige Adresse von P. Sherman wohl angeordnet auf den Bildschirm ausgibt!
Musterlösung: ausgabe.cpp

2.2 Schreiben Sie ein Programm, das den Radius eines Kreises einliest und dessen Umfang sowie den Flächeninhalt ausgibt.
Musterlösung: kreis.cpp

2.3 Schreiben Sie ein Programm, das zwei ganze Zahlen a und b einliest und das Ergebnis der Division a/b sowie den Divisionsrest ausgibt. Testen Sie ihr Programm mindestens mit folgenden Eingabewerten für die Zahl b: 2, 7 und 0.
Musterlösung: division.cpp

2.4 Schreiben Sie ein Programm, das einen Nettobetrag und einen Mehrwertsteuersatz vom Nutzer abfragt und daraus den Bruttobetrag und die Mehrwertsteuer errechnet. Geben Sie die Ergebnisse auf den Bildschirm aus.
Musterlösung: steuer.cpp

2.5 Die Formel für die Umrechnung einer Temperatur in Grad Fahrenheit nach Grad Celsius lautet:

$$T_C = \frac{5}{9} \cdot (T_F - 32)$$

Beispiel: 37°C sind etwa 100°F. Schreiben Sie ein Programm, das eine Temperatur in Grad Fahrenheit einliest und in Grad Celsius umrechnet.
Musterlösung: temperatur.cpp

2.6 Schreiben Sie ein Programm, das ein Datum als achtstellige Zahl der Form `TTMMJJJJ` als ganze Zahl einliest (Sie müssen nicht überprüfen, ob die Eingabe korrekt ist). Das Programm soll das Datum in der Form `TT.MM.JJJJ` ausgeben.
Musterlösung: datum.cpp

Die folgenden Aufgaben dienen Ihnen zur Festigung ihres Wissensstandes.

2.7 Schreiben Sie ein Programm, das füreine vom Nutzer einzugebende Anzahl der Sekunden die Zeit im Format Stunden:Minuten:Sekunden ausgibt.
Musterlösung: zeitumrechnung.cpp

2.8 Schreiben Sie ein Programm, das die zurückgelegte Strecke und die Geschwindigkeit nach einer bestimmten Zeit im freien Fall berechnet. Die Zeitdauer ist vom Nutzer zu bestimmen.
Musterlösung: freierfall.cpp

2.9 Ermitteln Sie mit einem Programm den Speicherbedarf der Datentypen `char`, `int`, `long`, `float` und `double`. Verwenden Sie dazu den Operator `sizeof()`.
Musterlösung: speicherplatz.cpp

2.10 Schreiben Sie ein Programm, welches zwei eingelesene Integer bitweise mit `und`, `oder` und `exklusiv-oder` verknüpft.
Musterlösung: bitweise.cpp

2.11 Schreiben Sie ein C++ Programm, welches zwei Zahlen vom Benutzer einliest und diese dividiert. Das Ergebnis soll korrekt auf Ganzzahlen gerundet werden. (D.h. bis $x.499999$ abgerundet, ab $x.5$ aufgerundet werden.)
Hinweis: Die Lösung kann nur mit Konvertierung der Zahlen zwischen **int** und **double** erfolgen (natürlich auch andere Varianten).
Musterlösung: divround.cpp

2.12 ***Zur Prüfungsvorbereitung!***
Schreiben Sie ein Programm, das die Blutalkoholkonzentration annähernd berechnet. Nutzen Sie folgende Formeln:

$$BAK = \frac{A}{m \cdot 0.62}$$

$$A = V \cdot \frac{e}{100} \cdot 0.8$$

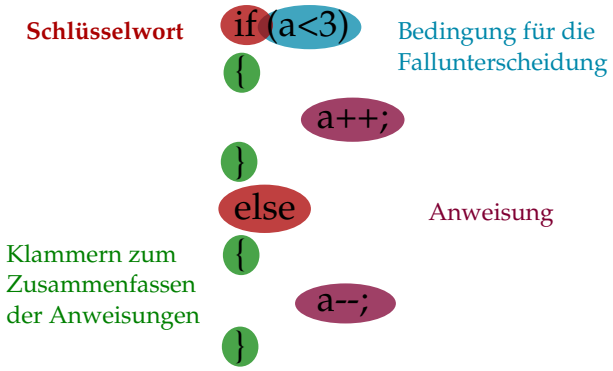
BAK = Massenanteil des Alkohols im Körper in ‰
A = aufgenommene Masse des Alkohols in Gramm (g)
m = Masse der Person in Kilogramm (kg)
V = Volumen des Getränkes in ml
e = Alkoholvolumenanteil in %
Musterlösung: blutalkohol.cpp

3 Alternativen und logische Verknüpfungen

Die Kontrollstruktur dient dazu, abhängig von Bedingungen unterschiedlichen Quellcode auszuführen bzw. wegzulassen.

Wie funktionierts?

Falls die Variable a in der folgenden Abbildung einen Wert enthält, der kleiner als 3 ist (z. B. 2), wird dem Ausdruck der Wahrheitswert true (wahr) zugewiesen. Als Folge führt das Programm alle Anweisungen aus, die in den geschweiften Klammern unter dem Schlüsselwort if stehen. Ansonsten (z. B. bei a=8) bekommt der Ausdruck den Wahrheitswert false (falsch) und alle Anweisungen in den Klammern nach else werden ausgeführt. Oder kurz und einfach: wenn → dann, sonst.



3.1 Schreiben Sie ein Programm, das eine Zahl x einliest und deren absoluten Betrag ausgibt.
Musterlösung: betrag.cpp

3.2 Schreiben Sie ein Programm, das dem Nutzer im Krankheitsfall gute Besserung wünscht.
Musterlösung: gutebesserung.cpp

3.3 Schreiben Sie ein Programm, das drei Zahlen einliest und die größte wieder ausgibt.
Musterlösung: groesstezahl.cpp

3.4 Schreiben Sie ein Programm, das drei Zahlen einliest und aufsteigend sortiert wieder ausgibt.
Musterlösung: sortieren.cpp

3.5 Schreiben Sie ein Programm, das zwei Zahlen a und b einliest und ausgibt, ob a ein Teiler von b ist. Verhindern Sie die Division durch 0!
Musterlösung: teiler.cpp

3.6 Schreiben Sie ein Programm, das zwei Zahlen einliest und bestimmt, ob mindestens eine davon durch 2 teilbar ist.
Musterlösung: gerade.cpp

3.7 Ein Unternehmen zahlt Boni für eine lange Betriebszugehörigkeit. Schreiben Sie ein Programm, das das monatliche Grundgehalt sowie die Dauer der Betriebszugehörigkeit abfragt und die Auszahlung berechnet. Ein Mitarbeiter bekommt ab dem 5. (bzw. 10., 15.) Jahr einen Bonus von 3% (8%, 18%).
Musterlösung: boni.cpp

3.8 Erstellen Sie einen minimalen Taschenrechner. Vom Nutzer werden folgende drei Eingaben verlangt: `operand1 operator operand2` (Beispiel für eine Eingabe: 1.5 * 120). Die Operanden sind Fließkommazahlen, der Operator ein Zeichen. Gehen Sie davon aus, dass der Operator eines der Zeichen `+, -, *, /` ist. Das Programm soll das Ergebnis der Rechenaufgabe ausrechnen und ausgeben.
Musterlösung: rechner.cpp

3.9 Erweitern Sie das Programm aus Aufgabe 2.6 so, dass der Monat nicht als Zahl sondern als Wort ausgegeben wird. Prüfen Sie außerdem, ob die Eingabe ein gültiges Datum ist. Gehen Sie davon aus, dass das Jahr kein Schaltjahr ist.
Musterlösung: datum2.cpp

3.10 Schreiben Sie ein Programm, das eine Jahreszahl einliest und ausgibt, ob es sich bei dem Jahr um ein Schaltjahr handelt. Es gelten folgende Regeln:

- Ein Jahr ist ein Schaltjahr wenn die Jahreszahl durch 4 teilbar ist.
- Ausnahme 1: Ein Jahr ist kein Schaltjahr wenn die Jahreszahl auch durch 100 teilbar ist.
- Ausnahme 2: Ein Jahr ist ein Schaltjahr wenn die Jahreszahl auch durch 400 teilbar ist.

Musterlösung: schaltjahr.cpp

3.11 Schreiben Sie ein Programm, das eine Zeitangabe der Form `HH:MM` in eine umgangssprachliche Form umwandelt (Beispiel: 10:15 in „viertel 11“). Lesen Sie die Zeit als ganze Zahl der Form `hhmm` ein. Sie können davon ausgehen, dass die Minutenzahl einen der Werte 0, 15, 30, 45 annimmt.
Musterlösung: uhrzeit.cpp

3.12 Schreiben Sie ein Programm, das zu den gegebenen Werten a, b und c die lineare Gleichung $a \cdot x + b = c$ löst.
Musterlösung: lingleichung.cpp

3.13 Schreiben Sie ein Programm, welches die quadratische Gleichung $ax^2+bx+c=0$ löst. Dabei soll unterschieden werden, ob es eine, zwei oder keine Lösung gibt. Die Lösungformel für quadratische Gleichungen lautet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Musterlösung: quadgleichung.cpp

3.14 Ostern ist ja bekanntlich jedes Jahr an einem anderen Tag, welcher sich nach dem ersten Vollmond des Frühlings richtet. Leider ist es für die Kalenderberechung mehr als ineffizient die Mondumläufe zu berechnen. Der Mathematiker Carl Friedrich Gauß veröffentlichte eine Formel um Anhand der Jahreszah das Datum des Ostersonntages direkt zu bestimmen. Belesen Sie sich, z.B. in der Wikipedia, über diese Formel und erstellen Sie ein C++ Programm, welches für ein eingegebenes Jahr das Datum des Ostersonntages ermittelt.
Musterlösung: ostern.cpp

3.15 Erstellen Sie ein Programm, das zu einem eingegebene Datum (Tag, Monat, Jahr) berechnet, um den wievielten Tag des Jahres es sich handelt.
Hinweis: Beachten Sie die Schaltjahre.
Musterlösung: tagdesjahres.cpp

4 Schleifen

Die for-Schleife: Die drei Ausdrücke innerhalb der runden Klammern nach dem Schlüsselwort for bestimmen den Ablauf der Schleife:

1. Der Beginn
   Der Ausdruck weist der Schleife eine Zählvariable zu. Sie heißt üblicherweise i und bekommt an dieser Stelle ihren initialen Wert zugewiesen, falls sie noch keinen besitzt.

2. Die Bedingung
   Vor jedem Schleifendurchlauf wird geprüft, welchen Wahrheitswert der Ausdruck besitzt:
   - true - die Anweisungen in den geschweiften Klammern werden ein weiteres Mal abgearbeitet
   - false - die Schleife wird beendet

3. Die Iteration
   Der Ausdruck bestimmt, wie die Zählvariable verändert wird.

Bsp: for (int i=1; i<10; i++)

Die while-Schleife: Nach dem Schlüsselwort while steht in runden Klammern die Bedingung für die Ablaufsteuerung der Schleife. Vor jedem Schleifendurchlauf wird geprüft, welchen Wahrheitswert sie besitzt:

- true - die Anweisungen in den geschweiften Klammern werden ein weiteres Mal abgearbeitet
- false - die Schleife wird beendet

Falls hier eine Zählvariable benötigt wird, initialisiert der Programmierer sie vor der Schleife und legt ihre Veränderung in der Schleife fest. Bsp: int i=1; while (i<10)

4.1 Schreiben Sie ein Programm, das die Quadratzahlen von $1^2 = 1$ bis $20^2 = 400$ ausgibt.
Musterlösung: quadratzahlen.cpp

4.2 Schreiben Sie ein Programm, das die ersten geraden N natürlichen Zahlen addiert und die Summe auf den Bildschirm ausgibt.
Musterlösung: summe.cpp

4.3 Schreiben Sie ein Programm, das eine vom Nutzer zu bestimmende Anzahl an Messwerten einliest und deren Durchschnitt, sowie das Maximum ausgibt.
Musterlösung: messwerte.cpp

4.4 Programmieren sie folgende Variante von Aufgabe 4.3: Es werden solange Messwerte eingelesen, bis der Nutzer zum ersten mal eine 0 eingibt (die 0 selbst gilt nicht mehr als Bestandteil der Messreihe).
Musterlösung: messwerte2.cpp

4.5 Schreiben Sie ein Programm, das in einer gegebenen Zahlenreihe aus N Elementen, die Anzahl von Elementen ermittelt, die nicht 0 sind.
Musterlösung: zahlenreihe.cpp

4.6 Schreiben Sie ein Programm, das eine natürliche Zahl einliest und deren Quersumme ausgibt.
Musterlösung: quersumme.cpp

4.7 Schreiben Sie ein Programm, das zwei natürliche Zahlen einliest und deren größten gemeinsamen Teiler ausgibt.
Musterlösung: ggt.cpp

4.8 Schreiben Sie ein Programm, das eine Zahl n einliest und $n!$ (Fakultät) ausgibt. Testen Sie ihr Programm mit den Werten $n = 10$ und $n = 20$.
Hinweis: $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$
Musterlösung: fakultaet.cpp

4.9 Schreiben Sie ein Programm, das eine Zahl n einliest und ausgibt, ob es sich um eine Primzahl handelt. Ein einfacher Primzahltest ist folgender: Man prüft für alle $k = 2 \ldots n-1$, ob k ein Teiler von n ist. Findet man einen Teiler k, so ist n offensichtlich keine Primzahl, findet man keinen Teiler, so ist n eine Primzahl.
Musterlösung: primzahl.cpp

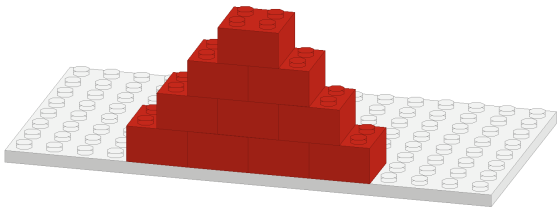
4.10 Auf einem Bankkonto wird ein Startbetrag K pro Jahr mit einem Zinssatz p verzinst. Schreiben Sie ein Programm, das Startbetrag, Zinssatz und einen Zielbetrag einliest und ausrechnet, nach wie vielen Jahren der Zielbetrag erreicht wird.
Musterlösung: zins.cpp

4.11 Bei einem Kinderspiel werden reihum die Zahlen von 1 bis 100 genannt. Allerdings werden Zahlen, die die Ziffer 7 enthalten oder durch 7 teilbar sind ausgelassen. Wer aus Versehen eine solche Zahl nennt, muss einen Pfand zahlen. Schreiben Sie ein Programm, das alle nach diesen Regeln zulässigen Zahlen ausgibt.
Musterlösung: spiel.cpp

4.12 Schreiben Sie ein Programm, das eine Zahl n einliest und deren Primfaktorzerlegung ausgibt.
Musterlösung: primzerlegung.cpp

4.13 Schreiben Sie ein Programm, das die Summe der ersten N Glieder der Zahlenfolge $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$ ausrechnet. Modifizieren Sie Ihr Programm so, dass die Summierung auch dann abbricht, wenn die Differenz zwischen zwei benachbarten Gliedern kleiner oder gleich als ein gegebenes Epsilon ist.
Musterlösung: harmonreihe.cpp

4.14 Schreiben Sie ein Programm, das berechnet, wie viele Legosteine zum Bau der folgenden Treppe mit der zurvor eingegebenen Höhe h notwendig sind:



Treppe der Höhe $h = 4$

Musterlösung: lego.cpp

4.15 Schreiben Sie ein Programm, welches zu einer eingegebenen Dezimalzahl deren Binärdarstellung berechnet.
Musterlösung: binaerzahl.cpp

4.16 ***Zur Prüfungsvorbereitung!***
Erweitern Sie ihr Programm aus Aufgabe 2.12 so, dass eine Tabelle auf den Bildschirm ausgegeben wird, die die jeweiligen Promillewerte und die Anzahl der getrunkenen Biere (bis 10) enthält. Gehen Sie davon aus, dass in einer Stunde ein Bier getrunken wird und der Körper 0.15 Promille Alkohol pro Stunde abbaut.
Musterlösung: bakbier.cpp

5 Felder und Matrizen

kurz und knapp

- dienen der Speicherung von Datenmengen gleichen Datentyps
- Zugriff auf einzelne Elemente erfolgt mittels einem Index
- Indexzählung beginnt in C/C++ mit 0 (null)

Wie funktionierts?
Feld deklarieren: Datentyp Feldname [Feldgröße];
Bsp: int a[5];

5.1 Schreiben Sie ein Programm, das 20 Messwerte einliest und in einem Feld speichert. Berechnet werden sollen:

- wie viele Zahlen über dem Durchschnitt der Messwerte liegen
- die Zahl, die zum Durchschnitt am nächsten liegt
- der größte Messwert und seine Position (Index) im Feld

Musterlösung: messwerte3.cpp

5.2 Schreiben Sie ein Programm, das prüft, ob es sich bei einem eingegebenem Wort (max. 30 Zeichen) um ein Palindrom handelt. Eine Zeichenkette heißt Palindrom, wenn sie von vorn sowie von hinten gelesen das gleiche ergibt.
Musterlösung: palindrom.cpp

5.3 Gegeben sei ein Feld von 10 ganzen Zahlen. Schreiben Sie ein Programm, das prüft, ob einige Zahlen in der Eingabe mehrfach vorkommen. Für diese soll die Anzahl des Auftretens im Feld ausgegeben werden. Ein Beispiel:

```
Eingabe: 5 5 6 7 5 9 4 2 2 7
Ausgabe: 3mal 5 , 2mal 7 , 2mal 2
```

Musterlösung: mehrfach.cpp

5.4 Programmieren Sie das *Sieb des Eratosthenes*(Primzahltafel)
Funktionsweise: Zunächst werden alle Zahlen 2, 3, 4,... bis zu einem Maximalwert S aufgeschrieben. Zunächst sind all die Zahlen potentielle Primzahlen. Im Folgenden werden alle Zahlen gekennzeichnet, die keine Primzahlen sind. Dazu wird vom Algorithmus jedes Vielfache der kleinsten nicht gekennzeichneten Zahl gekennzeichnet.
Das Verfahren beginnt also damit, die Vielfachen $4, 6, 8, \ldots$ der kleinsten Primzahl 2 zu streichen. Die nächste nicht gekennzeichnete Zahl ist die nächst größere Primzahl, die 3. Anschließend werden deren Vielfache $6, 9, 12, 15, \ldots$ durchgestrichen, usw.
Die obere Schranke S soll im Programm vor dem kompilieren festgelegt werden. Geben Sie alle Primzahlen bis zu der vorher gesetzten Schranke S aus. Jede Zahl aus dem Intervall einzeln auf die Primzahleigenschaft zu testen, ist nicht erwünscht und erforderlich. Überlegen Sie, an welchen Stellen man das Verfahren noch beschleunigen kann.
Musterlösung: eratosthenes.cpp

5.5 ***Zur Prüfungsvorbereitung!***
Gegeben ist eine Menge ganzer Zahlen (vorliegend in einem Feld). Gesucht ist die maximale Summe, die beim Aufaddieren hintereinanderfolgender Elemente entsteht. Des weiteren wird der Start- und der End-Index der Summe gesucht. Die Länge des Feldes sei n. Ein Beispiel:

Feld-Index:	0	1	2	3	4	5	6	7	8	9
Wert:	-1	3	4	-2	5	1	-9	4	2	-2

Die maximale zusammenhängende Summe wäre 11. Sie beginnt an Position 1 und endet mit der Position 5. Programmieren Sie ein C++-Programm, welches vom Nutzer die Anzahl der Zahlen(maximal 1000) erfragt, alle der Reihe nach einliest und die maximale Summe nach obigen Prinzip bestimmt.
Musterlösungen: maxsum1.cpp maxsum2.cpp maxsum3.cpp

5.6 ***Zur Prüfungsvorbereitung!***
Schreiben Sie ein Programm, das aus einem Feld mit ganzen Zahlen alle mehrfach vorkommenden Zahlen mit Ausnahme des ersten Auftretens entfernt. Die übrigen Zahlen sollen ihre Reihenfolge beibehalten und so zusammengeschoben werden, dass dazwischen keine Lücken entstehen. Am Ende des Felds entsteht eventuell ein Bereich ohne sinnvolle Daten. Ein Beispiel:

```
Eingabe: 2 5 2 3 3 6 3 7 4 5
Ausgabe: 2 5 3 6 7 4
```

Musterlösung: einzel.cpp

5.7 Gegeben seien zwei Integermatrizen A und B (je 5×3). Implementieren Sie ein C++-Programm mit folgendem Ablauf:

I. Einlesen der Matrix A

II. Ausgabe der Spaltensummen der Matrix A

III. A elementweise Quadrieren und das jeweilige Ergebnis an gleicher Position in B speichern

IV. Erzeugen einer Matrix C, deren Inhalt sich aus der Addition von A und B ergibt

V. Ausgabe der Matrix C in Matrixform (1 Matrixzeile entspricht einer Bildschirmzeile; Elemente werden durch Tabulatoren (`\t`) getrennt.)

Musterlösung: matrix.cpp

5.8 Erzeugen Sie eine Matrix mit 6 Zeilen und 15 Spalten. Lassen Sie den Inhalt der ersten Spalte vom Nutzer einlesen. Füllen Sie dann die restlichen Elemente der Matrix nach folgender Formel:

$$rm_{ij} = \begin{cases} rm_{i,j-1} + rm_{i+1,j-1} & \text{für i} = 1 \\ rm_{i,j-1} + rm_{i-1,j-1} & \text{für i} = 6 \\ rm_{i-1,j-1} - rm_{i,j-1} - rm_{i+1,j-1} & \text{sonst} \end{cases}$$

Geben Sie die Matrix zur Ansicht wohl formatiert aus und berechnen Sie anschließend welche Spalte der Matrix die größte Summe besitzt. Teilen Sie dem Nutzer auch die Summe selbst mit.
Musterlösung: rm.cpp

5.9 ***Zur Prüfungsvorbereitung!***
Prüfen Sie, ob ein vorgegebenes Sudoku richtig gelöst ist. Geben Sie im Fehlerfall die Position des ersten gefundenen Fehlers aus!

Musterlösung: sudoku.cpp

Tabelle 1: Sudoku - richtig gelöst

7	6	5	4	1	2	9	3	8
1	8	2	9	5	3	4	7	6
3	9	4	7	6	8	1	5	2
8	1	6	3	2	9	5	4	7
9	4	3	8	7	5	6	2	1
2	5	7	6	4	1	8	9	3
4	2	9	1	8	7	3	6	5
5	3	8	2	9	6	7	1	4
6	7	1	5	3	4	2	8	9

Tabelle 2: Sudoku - falsch gelöst

7	6	5	4	1	2	9	3	8
1	8	2	9	5	3	4	8	6
3	9	4	7	6	8	1	5	2
8	1	6	3	2	9	5	4	7
9	4	3	8	7	5	6	2	1
2	5	7	6	4	1	8	9	3
4	2	9	1	8	2	3	6	5
5	3	8	2	9	6	7	1	4
6	7	1	5	3	4	2	8	9

6 Strukturen

kurz und knapp

- dienen der Speicherung von Datenmengen unterschiedlichen Typs
- Sinn der Datenverknüpfung ergibt sich aus dem Kontext der Aufgabe

Wie funktionierts?

- Struktur deklarieren:

```
struct name
{
        datentyp element1;
        datentyp element2;
};
```

- Bsp.: Weihnachten
  Der Weihnachtsmann teilt jedes Jahr unzählige Geschenke an Kinder aus. Damit jedes Kind auch das richtige Geschenk bekommt, benötigt er ein paar Informationen über die Kinder: Geburtsjahr sowie -land und natürlich, ob das Kind lieb war. Um den Überblick zu behalten, hat der DV-Wichtel eine Datenstruktur angelegt:

```
struct kind
{
        int gebjahr;
        char land[4];
        bool lieb;
};
```

- Verwendung:
  Der Name der Struktur ist nun als neuer Datentyp zu verwenden. In jeder Variable vom Typ kind können nun die drei Informationen gespeichert werden. Der Zugriff auf die Elemente erfolgt mit dem Punktoperator: variablenname.elementname

```
int main()
{
        kind pippilotta;
        pippilotta.gebjahr = 1941;
        pippilotta.land = "S";
        pippilotta.lieb = true;
};
```

- Feld einer Struktur:

```
int main()
{
        kind pippilotta[5];
        pippilotta[0].gebjahr = 2007;
        pippilotta[2].land = "GER";
        pippilotta[3].lieb = true;
};
```

6.1 Entwerfen Sie eine Datenstruktur, die die Koordinaten eines Punktes in der Ebene speichern kann. Schreiben Sie ein Programm, das die Koordinaten zweiter Punkte einliest und deren Abstand berechnet.
Musterlösung: abstand.cpp

6.2 Schreiben Sie ein Programm, das die Koordinaten von 10 Punkten in der Ebene(siehe Aufgabe 6.1) einliest und in einem Feld speichert. Anschließend sollen die Punkte bestimmt werden, die dem Ursprung am nächsten liegen bzw. am weitesten entfernt liegen (wenn dabei mehrere Punkte gleich weit entfernt sind, soll einer davon angegeben werden).
Musterlösung: punkte.cpp

6.3 Entwerfen Sie eine Datenstruktur, die Ort und Größe eines Kreises speichern kann. Schreiben Sie ein Programm, das einen Punkt und einen Kreis einliest und ausgibt, ob der Punkt im Kreis liegt oder außerhalb. Nutzen Sie für den Punkt die Datenstruktur aus Aufgabe 6.1.
Musterlösung: kreis2.cpp

6.4 Schreiben Sie ein Programm für die Verwaltung von Einschreibungen für eine Lehrveranstaltung. Erstellen Sie zunächst eine Datenstruktur, die die benötigten Daten eines Studenten aufnehmen kann. Für die Lehrveranstaltung können maximal 30 Studenten eingeschrieben sein. Das Programm soll wie folgt ablaufen: Zuerst werden die Daten eines Studenten eingelesen. Anschließend wird gefragt, ob noch ein weiterer Datensatz folgt. Falls der Nutzer dies bestätigt, kann er einen weiteren Datensatz eingeben und wird danach erneut gefragt, ob er noch ein weiterer Datensatz eingelesen werden soll. Sobald der Nutzer eingibt, dass keine Daten mehr folgen oder die maximale Teilnehmerzahl erreicht ist, werden alle eingetragenen Studenten auf dem Bildschirm ausgegeben und das Programm anschließend beendet.
Musterlösung: einschreibung.cpp

6.5 Schreiben Sie ein Programm für die Personalverwaltung eines kleinen Betriebs. Für jeden Mitarbeiter werden Name, Alter und Jahresgehalt gespeichert. Der Betrieb kann bis zu 30 Mitarbeiter haben. Nach dem Starten soll das Programm den Nutzer folgendes Menü anbieten:

```
1 - alle Mitarbeiter anzeigen
 (Gibt Name, Alter und Gehalt aller Mitarbeiter aus)
2 - durchschnittliches Alter ausgeben
3 - Mitarbeiter einstellen
 (Liest die Daten für einen neuen Mitarbeiter ein
  und fügt ihn dem Personal hinzu)
4 - Gehaltserhöhung
 (Erhöht das Gehalt aller Mitarbeiter um 10%)
5 - Topverdiener anzeigen
 (Gibt alle Mitarbeiter aus, deren Gehalt
  mindestens 20% über dem Durchschnitt liegt)
6 - Mitarbeiter entlassen
 (Liest einen Name ein, wenn ein Mitarbeiter mit diesem
  Name existiert, wird er aus der Personalliste entfernt)
x - Programm beenden
```

Der Nutzer kann jetzt einen Menüpunkt auswählen, indem er '1' - '6' oder 'x' eingibt. Wenn 'x' eingegeben wurde, wird das Programm beenden, andernfalls wird der entsprechende Menüpunkt ausgeführt und anschließend wieder das Menü angezeigt.
Musterlösung: personal.cpp

6.6 Schreiben Sie ein Programm, das die Möglichkeit bietet, die Daten (Name, Wohnort, Telefonnummer) von N Personen einzulesen sowie die Möglichkeit nach einer bestimmten Person zu suchen. Dabei soll der Nutzer zuerst das Suchkriterium wählen und danach das Suchwort bzw. die Nummer eingeben.
Musterlösung: telefonbuch.cpp

6.7 Ein Farbbild sei gegeben durch eine Tabelle mit 10×10 Einträgen. Jeder Eintrag enthält die Farbinformation (Rot-, Grün- und Blauanteil jeweils von 0 bis 255) für den entsprechenden Bildpunkt. Schreiben Sie ein Programm, das die Anzahl von komplett weißen (rot=255, grün=255, blau=255) und komplett schwarzen (rot=0, grün=0, blau=0) Punkten ermittelt. Die Farbinformation sei gegeben durch die folgende Struktur:

```
struct pix
{
        int r;
        int g;
        int b;
};
```

Musterlösung: farbbild.cpp

6.8 ***Zur Prüfungsvorbereitung!***
Schreiben Sie ein Programm, das die Eigenschaften Marke (char[ ]), Kilometerstand (int), Alter (int), PS (int) und Unfallwagen (bool) für 10 vorgegebene PKW mittels einer Datenstruktur speichert. Alle Informationen sollen in einer Tabelle wohlformatiert ausgegeben und um die pro Jahr gefahrenen Kilometer ergänzt werden. Geben Sie alle Tabelleneinträge geordnet nach dem Alter der Fahrzeuge aus. Ergänzen Sie anschließend eine Suche, ob die Lieblingsautomarke des Nutzers enthalten ist.

Hinweis: Sie erhalten eine Daten- und Programmvorlage unter `https://www.tu-chemnitz.de/informatik/friz/Grundl-Inf/Aufgaben/aufgaben.php`
Musterlösung: auto.cpp

7 Funktionen

kurz und knapp

Eine Funktion ist eine Zusammenfassung von Anweisungen. Durch die Verwendung von Funktionen wird Quellcode strukturiert und reduziert, da eine einmal geschriebene Funktion beliebig oft aufrufbar ist. Sie wird wie folgt vereinbart:

```
Datentyp des Rueckgabewertes Funktionsname (Parameterliste)
{
    Anweisung(en);
    return ... ;
}
```

Außerdem:

- soll die Funktion nichts zurückgeben, heißt der Datentyp void und das Argument nach return entfällt
- Parameter sind einzeln, durch ein Komma getrennt mit Datentyp aufzulisten

Wie funktionierts?

Im Beispiel addiert die Funktion add() die Parameter a und b und gibt das Ergebnis an die aufrufende main()-Funktion zurück. Es ist wichtig darauf zu achten, dass die Datentypen miteinander übereinstimmen: Die zu addierenden Zahlen a und b sind Integer. Folglich ist auch das Ergebnis der Addition ein Integer. Dem entsprechend lautet der Datentyp der Rückgabe int.

In der main()-Funktion müssen die Variablen (x,y), die als Parameter an add() übergeben werden ebenfalls den gleichen Datentyp besitzen. Soll der Rückgabewert von add() gespeichert werden, muss der Datentyp dieser Variable (erg) mit dem Datentyp der Rückgabe von add() übereinstimmen.

```
int add (int a, int b)
{
    return a+b;
}

int main ()
{
        int erg, x=8, y=2;
        cout << add(x,y); // in add: 8+2
        erg = add(y,x); // in add: 2+8
        cout << erg;
        return 0;
}
```

Die Arbeitsweise von Funktion ist immer mit einem kleinen Hauptprogramm zu demonstrieren.

7.1 Schreiben Sie eine Funktion `double sqr (double n)` zum Berechnen des Quadrats einer Fließkommazahl. Das Ergebnis soll an die aufrufende Funktion zurückgegeben werden.
Musterlösung: sqr.cpp

7.2 Schreiben Sie eine Funktion, welche das Vorzeichen einer übergebenen Zahl zurückgibt. Wenn die Zahl größer als 0 ist soll 1, wenn sie kleiner als 0 ist soll -1 und im Falle gleich 0 soll 0 zurückgegeben werden.
Musterlösung: sgn.cpp

7.3 Schreiben Sie eine Funktion `double min (double a, double b)`, die von zwei Fließkommazahlen a und b die kleinere zurückgibt.
Musterlösung: min.cpp

7.4 Schreiben Sie eine Funktion, die die Potenz a^b für eine Fließkommazahl a und eine natürliche Zahl b berechnet und zurückgibt.
Musterlösung: pow.cpp

7.5 Verwenden Sie die Funktion aus Aufgabe 7.4 um ein Polynom n-ten Grades $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots a_0$ an einer gegebenen Stelle x_0 auszuwerten. Die Koeffizienten $a_n \ldots a_0$ sind vom Benutzer abzufragen.
Musterlösung: polynom.cpp

7.6 Schreiben Sie eine Funktion, die die Fakultät $n!$ einer natürlichen Zahl berechnet (siehe Aufg. 4.8).
Musterlösung: fakultaet2.cpp

7.7 Schreiben Sie eine Funktion, die den Abstand zwischen zwei Punkten berechnet und zurückgibt. Verwenden Sie dabei die Funktion `sqr` aus Aufgabe 7.1.
Musterlösung: abstand2.cpp

7.8 Verwenden Sie die Fakultätsfunktion aus Aufgabe 7.6 um folgende Zahlenfolge zu berechnen:

$$-1 + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^N \cdot \frac{1}{N!}$$

Musterlösung: zahlenfolge2.cpp

7.9 Schreiben Sie eine Funktion, die die Reihenfolge der Ziffern einer ganzen Zahl umkehrt ($1234 \mapsto 4321$). Das Ergebnis soll ebenfalls als ganze Zahl zurückgegeben werden.
Musterlösung: zahlspiegeln.cpp

7.10 Schreiben Sie eine Funktion welche den Wert der e-Funktion $e^x = \exp(x)$ berechnet. Dies lässt sich zum Beispiel realisieren, indem man die unendliche Summe

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

nach einer vorher festgelegten Anzahl von Schritten abgebrochen wird.
Musterlösung: exp.cpp

7.11 Schreiben Sie ein Programm, das alle Primzahlen bis 1000 ausgibt. Entwerfen Sie dazu eine Funktion die zurückgibt, ob eine Zahl eine Primzahl ist.
Musterlösung: primzahlen.cpp

7.12 ***Für Fortgeschrittene!***
Schreiben Sie eine Funktion, die ein Feld a sowie zwei Indizes i, j als Parameter übernimmt und in dem durch die beiden Indizes i und j begrenzten Teilfeld das kleinste Element sucht. Die Funktion soll die Position, an der das kleinste Element gefunden wurde, zurückgeben.
Musterlösung: feld_min_index.cpp

7.13 ***Für Fortgeschrittene!***
Überlegen Sie sich auf Grundlage von Aufgabe 7.12 ein Verfahren, um ein Feld zu sortieren.

Musterlösung: selectionsort.cpp

7.14 Lösen Sie folgende Aufgaben unter Verwendung von Bibliotheksfunktionen:

- Ermitteln Sie die Länge einer eingegebenen Zeichenkette.
- Testen Sie die Zeichen einer Zeichenkette, ob Zahlen darin vorkommen.
- Alle Buchstaben eines eingegeben Wortes sollen in Großbuchstaben umgewandelt werden.
- Berechnen Sie x^y für vom Nutzer einzugebende Werte.

Hinweis: Nutzen Sie für eventuelle online-Recherchen `http://www.cplusplus.com/reference`
Musterlösung: bibo.cpp

8 Rekursion

Eine rekursive Funktion besteht damit immer aus zwei Teilen:

- Lösung des Elementarproblems
- Veränderung des Problems

8.1 Lösen Sie Aufgabe 4.14 mithilfe einer rekursiven Funktion.
Musterlösung: lego_rek.cpp

8.2 Schreiben Sie eine Funktion, welche die Summe aller Quadratzahlen von 1 bis n^2 rekursiv berechnet.
Musterlösung: summequad.cpp

8.3 Schreiben Sie eine rekursive Funktion zur Berechnung der Fakultät $n!$ einer natürlichen Zahl.
Musterlösung: fakultaet3.cpp

8.4 Schreiben Sie eine rekursive Funktion, welche die Quersumme einer natürlichen Zahl n berechnet.
Musterlösung: quersummerek.cpp

8.5 Schreiben Sie eine rekursive Funktion, welche die Potenz a^b einer reellen Zahl a und einer natürlichen Zahl b berechnet.
Musterlösung: powrek.cpp

8.6 Ein Bauer erhält eine junge Kuh. Ab ihrem 3.Lebensjahr bringt sie jedes Jahr ein weibliches Kalb zur Welt. Alle Nachkommen werfen ebenso ab dem 3.Lebensjahr jährlich genau 1 weibl. Kalb. Wieviele Kühe hat der Bauer nach n Jahren, wenn angenommen wird, dass alle Kühe ewig leben? Wieviele Kühe hat der Bauer nach n Jahren, wenn angenommen wird, dass eine Kuh k Jahre lebt?
Hinweis: Denken Sie hierbei an die Fibbonaccifolge.
Musterlösung: fibbonacci.cpp

8.7 Eine Rehpopulation in einem Erzgebirgswald wird am 1.1. des ersten Jahres gezählt. Es ergibt sich eine Zahl von 12 Rehen. Die Population ändert sich nach Beobachtungen von Naturfreunden wie folgt:

- Im Laufe eines Jahres verdreifacht sich die Zahl der Rehe durch Vermehrung.
- Zur Sicherung des Silversterbratens der umliegenden Dörfer werden am Jahresende 10 Rehe geschossen.
- Zum Schutz des Waldes vor Wildverbiss werden am Ende des Jahres zusätzlich 70% (nächste ganze Zahl, die kleiner oder gleich 70% der Anzahl ist) der verbliebenen Rehe geschossen, wenn deren Zahl größer als 40 ist.

Schreiben Sie eine rekursive Funktion, welche die Anzahl der lebenden Rehe zu Beginn des n-ten Jahren berechnet. Demonstrieren Sie die Verwendung dieser Funktion in einem kleinen Hauptprogramm, in dem Sie eine Tabelle der Populationsentwicklung über 10 Jahre erstellen.
Musterlösung: rehe.cpp

8.8 Gegeben sei die Ackermannfunktion:

$$\text{ack}(m,n) = \begin{cases} n+1 & falls\ m = 0 \\ \text{ack}(m-1,1) & falls\ n = 0, m > 0 \\ \text{ack}(m-1, ack(m, n-1)) & sonst \end{cases}$$

Implementieren Sie diese als rekursive C++-Funktion.
Hinweis: Verwenden Sie als Testwerte für m nur Werte bis einschließlich 3!
Musterlösung: ackermann.cpp

8.9 Die Stirling-Zahl zweiter Art $\text{S}(n,k)$ gibt die Anzahl der k-elementigen Partitionen einer n-elementigen Menge an. Einfach ausgedrückt: Alle Elemente (n) einer Menge müssen sich in eine bestimmte Anzahl von Guppen (k) aufteilen. Wie viele möglichkeiten gibt es?

Beispiel: Die folgende Menge M mit $n = 4$ Elementen kann in folgende 2-Partitionen ($k = 2$) zerlegt werden.

M = 2-Partitionen:$\{a,b,c,d\}$
$\{\{a,b\},\{c,d\}\}$
$\{\{a,c\},\{b,d\}\}$
$\{\{a,d\},\{b,c\}\}$
$\{\{a,b,c\},\{d\}\}$
$\{\{a,b,d\},\{c\}\}$
$\{\{a,c,d\},\{b\}\}$
$\{\{b,c,d\},\{a\}\}$
Die Anzahl der 2-Partitionen beträgt 7.

Eine Partition einer n-elementigen Menge kann höchstens n Blöcke enthalten: $\text{S}(n,k) = 0$ falls $k > n$

Außerdem gibt es nur eine 1-Partition von n und eine n-Partition von n: $\text{S}(n,1) = 1$ und $\text{S}(n,n) = 1$

Schreiben Sie ein Programm, das alle Stirling-Zahlen 2.Art $\text{S}(n,k)$ für $n <= 10$ ausgibt! Die Funktion $\text{S}(n,k)$ ist durch folgende Rekursionsformel definiert:

$$\text{S}(n,k) = \text{S}(n-1,k-1) + k \cdot \text{S}(n-1,k)$$

Musterlösung: stirling2.cpp

8.10 Nachfolgendes Lineal soll mit Hilfe eines Unterprogramms erzeugt werden. Dabei sind folgende Konventionen einzuhalten:

- Der mittlere Strich ist genau n Einheiten hoch.
- Der mittlere Teilstrich des (durch die erste Teilung entstandenen) linken Intervalls ist genau $n-1$ Einheiten hoch.
- Analoges gilt für das rechte Intervall.
- Der mittlere Teilstrich der dadurch entstehenden Intervalle ist genau n-2 Einheiten hoch.
- Die Teilung soll fortgesetzt werden, bis der Teilstrich nur noch eine Einheit hoch ist.

Beispiel für $n = 4$:

```
        |
   |    |    |
 | | | | | | | |
|||||||||||||||||
```

Formulieren Sie ein rekursives Unterprogramm zur Ausgabe dieses Lineals. Zur Vereinfachung kippen Sie das Lineal um -90 Grad.
Musterlösung: lineal.cpp

8.11 ***Für Fortgeschrittene!***
Schreiben Sie eine Funktion, welche alle Elemente eines übergebenen Feldes aufaddiert und diesen Wert zurück gibt.
Musterlösung: feld_sumrek.cpp

8.12 ***Für Studiengänge B_AP, B_MB, B_MM***
Erweitern Sie die Aufgabe 8.8 so, dass Sie die Anzahl der Rekursionsschritte angeben können. Wie oft wird die Funktion mit $m = 4$ und $n = 1$ aufgerufen?
Musterlösung: ackermann_anz.cpp

8.13 ***Zur Prüfungsvorbereitung!***
Binomialkoeffizienten $\binom{n}{k}$ (sprich n über k) sind wichtige mathematische Größen. Sie können auch rekursiv berechnet werden. Dafür gilt

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k} + \binom{n-1}{k-1} & \text{falls } n > k > 0 \\ 1 & n = k \text{ oder } k = 0 \end{cases}$$

Schreiben sie eine rekursive Funktion, welche die Binomialkoeffizienten für gegebene n und k berechnet.
Musterlösung: bino.cpp

9 Zur Prüfungsvorbereitung

Dieses Kapitel beinhaltet umfassende Aufgaben zur Vorbereitung auf die Klausur. Sie entsprechen in ihrem Umfang nicht den Klausuraufgaben.

9.1 Ein Unternehmen vertreibt verschiedene Getränke. Oft stellen Kunden Fragen zu den Produkten. Damit die E-Mail mit den Fragen an den richtigen Mitarbeiter gelangt, soll ein Programm erkennen, um welche Produktsorte es geht und einen Text in eine Datei speichern. Das von Ihnen zu entwerfende Programm soll also in einem Text (einzulesen aus einer Datei, Dateiname wird abgefragt) Schlagworte (siehe Tabelle) suchen und je nach Ergebnis folgenden Text generieren und speichern:
Hi <name>, du hast eine neue Nachricht zum Thema <buzzword>.

buzzword	**name**	**name outputfile**
Kaffee	Paul	new_mail_to_Paul.txt
Bohnen	Paul	new_mail_to_Paul.txt
Koffein	Paul	new_mail_to_Paul.txt
Wein	Peet	new_mail_to_Peet.txt
Sekt	Peet	new_mail_to_Peet.txt
Bier	Peet	new_mail_to_Peet.txt
Saft	Phil	new_mail_to_Phil.txt
Wasser	Phil	new_mail_to_Phil.txt
Schorle	Phil	new_mail_to_Phil.txt

Hinweis: Die Bibliothek <fstream> ermöglicht Ihnen die Dateiarbeit und <cstring> einen leichteren Umgang mit C-Zeichenketten.
Musterlösung: mail.cpp

9.2 In einer Spielshow bekommt ein Kandidat Fragen gestellt, die er richtig beantworten muss. Schreiben Sie ein Programm, das ein solches Spiel durchführen kann. Benutzen Sie zum Speichern der Fragen und Antworten folgende Struktur:

```
struct data
{
        char frage[2000];
        char antwort[4][100];
        int schwierigkeit;
        int id;
};
```

Aus dieser Struktur wird ein Feld fragen[1000] aufgebaut. Wenn weniger als 1000 Fragen enthalten sind, belegen Sie die Komponente id des ersten freien Elementes mit 0. Es gibt folgende vereinfachte Regeln:

- Der Kandidat muss insgesamt 6 Fragen beantworten. Die Fragen haben 3 Schwierigkeitsstufen. Er bekommt also zuerst 2 Fragen der Stufe 1, dann 2 Fragen der Stufe 2 und zuletzt 2 Fragen der Stufe 3 gestellt.

- Die Fragen werden zufällig aus einem Fragenpool ausgewählt.
- Beim Stellen der Fragen sind die Antwortmöglichkeiten in zufälliger Reihenfolge anzuzeigen.
- Fragen dürfen nicht doppelt gestellt werden.
- Beantwortet der Kandidat eine Frage falsch, ist das Spiel zu Ende.

Der Fragenpool und die dazugehörigen Antwortmöglichkeiten sind der Datei *data.txt* zu entnehmen. Die jeweils erste Antwort ist die richtige Antwort. Jede Zeile der Datei hat folgenden Aufbau:
ID Frage Antwort Antwort Antwort Antwort Schwierigkeitsstufe
In der letzten Zeile der Datei steht eine 0, die das Ende der Datei.
Erstellen Sie eine zweite Variante des Programms, in der die ID nicht in der Datei enthalten ist.
Hinweis: Die Bibliothek <fstream> ermöglicht Ihnen die Dateiarbeit.
<stdlib.h> beinhaltet Funktionen zum Ermitteln von Zufallszahlen.
Musterlösung: spielshow.cpp

9.3 An einer Prüfung nehmen Studierende (max. 400) verschiedener Studiengänge teil. Entwerfen Sie ein C++-Programm, das für eine absolvierte Prüfung den Anteil der Teilnehmer berechnet, die bestanden haben (ohne Unterscheidung der Studiengänge). Der Nutzer gibt die Gesamtpunktzahl ein. Eine Prüfung gilt als bestanden, wenn der Teilnehmer mindestens 40% der Gesamtpunktzahl erreicht. Alle nötigen Daten bezüglich der Studierenden stehen in einer Datei. Der erste Wert in der Datei gibt die Anzahl der Teilnehmer an. Fügen Sie eine Option in das Programm ein, mit der Nutzer die Anzahl der Teilnehmer eines Studienganges seiner Wahl erfährt.
Hinweis: Die Bibliothek <fstream> ermöglicht Ihnen die Dateiarbeit und <cstring> einen leichteren Umgang mit C-Zeichenketten.
Musterlösung: pruefung.cpp

9.4 Ein Magier benötigt zum Herstellen verzauberter Waffen und Rüstungen 3 verschiedene Materialien: feuerfeste Drachenhaut, goldene Federn eines Engels und magische Essenzen. Je nach Anzahl der eingesetzten verschiedenen Materialien entstehen beim Herstellen Gegenstände unterschiedlicher Wertigkeit:

- Stufe 1: normale Gegenstände, benötigt werden 2 Drachenhäute, 1 Feder und 1 magische Essenz
- Stufe 2: seltene Gegenstände, benötigt werden 5 Drachenhäute, 2 Federn und 3 magische Essenzen
- Stufe 3: legendäre Gegenstände, benötigt werden 8 Drachenhäute, 10 Federn und 8 magische Essenzen

Version 1:
Schreiben Sie eine Funktion, die rekursiv die Gesamtanzahl der Gegenstände berechnet und zurückgibt, deren Herstellung aus dem Inventar des Magiers möglich ist. Die Anzahl der Materialien im Inventar sind vom Nutzer einzulesen und der Funktion geeignet zu übergeben. Es sollen dabei immer möglichst viele Gegenstände der höheren Wertigkeit entstehen. Der Magier möchte nun seine Gegenstände verkaufen. Unabhängig von der Stufe der Gegenstände erhält er ein Gesamtangebot über den Preis zum Verkauf aller Gegenstände, die hergestellt wurden. Er entscheidet sich gegen den Verkauf, wenn für einen einzelnen Gegenstand durchschnittlich weniger als 128 Euro geboten werden. Schreiben Sie eine Funktion, die der aufrufenden Funktion mitteilt, ob der Magier seine Gegenstände verkauft. Das Preisangebot ist ein Parameter der Funktion.

Zeigen Sie eine geeignete Verwendung beider Funktionen in der main()-Funktion und geben Sie für die verschiedenen Ereignisse entsprechende Hinweistexte an den Nutzer aus.

Für Studiengänge B_ AP, B_ MB, D_ MB, B_ MM
Version 2:
Schreiben Sie eine Funktion, die **rekursiv** die Anzahl der Gegenstände berechnet, deren Herstellung aus dem Inventar des Magiers möglich ist. Die Anzahl der Materialien im Inventar sind vom Nutzer einzulesen und der Funktion geeignet zu übergeben. Es sollen dabei immer möglichst viele Gegenstände der höheren Wertigkeit entstehen. Geben Sie in der main()-Funktion aus, wie viele Gegenstände insgesamt und wie viele pro Stufe erzeugt werden und geben Sie für die verschiedenen Ereignisse entsprechende Hinweistexte an den Nutzer aus. Die Anzahl der insgesamt erzeugten Gegenstände soll von der Funktion zurück gegeben werden. Verwenden Sie zum Speichern der Anzahl der pro Stufe erzeugten Gegenstände folgende Struktur:

```
struct items
{
    int normal;     // Anzahl normaler Gegenstände
    int rare;       // Anzahl seltener Gegenstände
    int legend;     // Anzahl legendärer Gegenstände
};
```

Der Magier möchte nun seine Gegenstände verkaufen. Unabhängig von der Stufe der Gegenstände erhält er ein Gesamtangebot über den Preis zum Verkauf aller Gegenstände, die hergestellt wurden. Er entscheidet sich gegen den Verkauf, wenn für einen einzelnen Gegenstand durchschnittlich weniger als 128 Euro geboten werden. Geben Sie für die verschiedenen Ereignisse entsprechende Hinweistexte an den Nutzer aus.
Musterlösung: magisch1.cpp, magisch2.cpp