Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Institutsleitung
Prof. Dr.-Ing. K. D. Müller-Glaser
Prof. Dr.-Ing. J. Becker
Prof. Dr. rer. nat. W. Stork

Übung03: Informationstechnik (IT)

Tobias Schwalb & Timo Sandmann & Stephan Werner

Teil1: Besprechung der Übungsaufgaben 2.01-2.06

Inhalt: Übung03 – Teil 1

Karlsruher Institut für Technologie

- Aufg. 2.01: Verständnisfragen
- Aufg. 2.02: Schleifenarten
- Aufg. 2.03: Mehrfachentscheidungen
- Aufg. 2.04: Anwendung von Kontrollstrukturen
- Aufg. 2.05: Arrays und Schleifen
- Aufg. 2.06: Mehrdimensionale Arrays und Schleifen

Aufg. 2.01: Verständnisfragen Lsg. (1)

a) In C++ ist der Index des **ersten** Arrayelements 0.

b) In C++ ist der Index des **letzten** Arrayelements N - 1.

c) Wenn zur Laufzeit eines Programms der für einen Array verwendete Index den zulässigen Bereich verlässt, gibt der Compiler eine Fehlermeldung aus. Falsch

d) Gegeben sei das folgende Array: `int myArray[] = { 0, 10, 20 };`
Dann adressiert der Ausdruck: **`myArray + 2`**

i. das dritte Arrayelement. Richtig
ii. das zweite Arrayelement. Falsch
iii. die Speicherstelle, die sich 2 Bytes entfernt vom Anfang des Arrays befindet. Falsch

Aufg. 2.01: Verständnisfragen Lsg. (2)

e) Benennen Sie alle Fehler im folgenden Codeausschnitt:

```
float index = 0;
float[5] myArray =
       {0, 1, 2, 3, 4, 5};
myArray[index] = 10;
```

```
int index = 0;
float myArray[6] =
       {0, 1, 2, 3, 4, 5};
myArray[index] = 10;
```

f) Nach Ausführung der folgenden Anweisungen speichert die Variable **`x`** den Wert: 10

```
int x = 1, y = 10;
if( y > 0 ) {
  if( y < 5 ) {
    x = 5;
  } else {
    x = 10;
  }
}
```

Aufg. 2.01: Verständnisfragen Lsg. (3)

g) Nach Ausführung der folgenden Schleife speichert die Variable **x** den folgenden Wert:

i. 1 Falsch ii. 2 Richtig iii. 3 Falsch

```
int x = 0, y = 1;
while( ++y < 3 ) {
  x += y;
}
```

h) Nach Ausführung der folgenden Schleife speichert die Variable **x** den folgenden Wert:

i. 1 Falsch ii. 2 Falsch iii. 3 Richtig

```
int x = 0, y = 1;
do {
  x += y;
} while( ++y < 3 );
```


Aufg. 2.01: Verständnisfragen Lsg. (4)

i) Geben Sie eine Möglichkeit an, eine **for**-Schleife zu unterbrechen:

```
break;
```

Aufg. 2.02: Schleifenarten Lsg.

- Erklären Sie kurz den Unterschied zwischen einer **`while`**-Schleife, einer **`do while`**-Schleife und einer **`for`**-Schleife:
  - **`while`**: Die Bedingung wird jeweils **vor** Abarbeitung des Schleifenrumpfs überprüft.
  - **`do while`**: Die Bedingung wird **nach** Abarbeitung des Schleifenrumpfs überprüft. Dadurch wird die Schleife mindestens einmal ausgeführt.
  - **`for`**: Ausführung des Schleifenrumpfs, wenn die Bedingung jeweils **vor** Abarbeitung des Schleifenrumpfs erfüllt ist. Zusätzlich ist die Möglichkeit gegeben, jeweils Anweisungen anzugeben, die einmalig **vor der Schleife** bzw. **nach jedem Durchlauf** ausgeführt werden.

Aufg. 2.03: Mehrfachentscheidungen Lsg.

Transformieren Sie den folgenden Codeausschnitt unter Anwendung einer **`switch`**-Anweisung und vermeiden Sie dabei redundanten Conde zur Ausgabe:

```
unsigned int number;
cout << "Geben Sie eine Zahl zwischen 1 und 5 ein: ";
cin >> number;

if( number == 0 ) {
  cout << "Eingabe zu klein!\n";
} else if( number == 1 ) {
  cout << "Eingabe ok und ungerade\n";
} else if( number == 2 ) {
  cout << "Eingabe ok und gerade\n";
} else if( number == 3 ) {
  cout << "Eingabe ok und ungerade\n";
} else if( number == 4 ) {
  cout << "Eingabe ok und gerade\n";
} else if( number == 5 ) {
  cout << "Eingabe ok und ungerade\n";
} else {
  cout << "Eingabe zu gross!\n";
}
```

```
switch( number ) {
case 0:
  cout << "Eingabe zu klein!\n";
  break;
case 1:
case 3:
case 5:
  cout << "Eingabe ok und ungerade\n";
  break;
case 2:
case 4:
  cout << "Eingabe ok und gerade\n";
  break;
default:
  cout << "Eingabe zu gross!\n";
}
```

Aufg. 2.04: Anwendung von Kontrollstrukturen

- Schreiben Sie die erforderlichen Anweisungen, um
  a) zwei Gleitpunktzahlen im Dialog einzulesen und die größere Zahl von der Kleineren zu subtrahieren und das Ergebnis auszugeben.

  b) Gleitpunktzahlen einzulesen und aufzusummieren, bis ihre Summe 100 überschreitet.

  c) positive Ganzahlen solange einzulesen, bis eine negative Zahl eingegeben wird.

  d) den Anwender aufzufordern, drei verschiedene Ganzzahlen einzugeben. Die Aufforderung wird wiederholt, solange zwei von drei Zahlen übereinstimmen.

- Hinweis: Bitte beachten Sie, dass Sie auch Schleifen nur mit Bedingungen und ohne Anweisungen schreiben können.

Aufg. 2.04: Kontrollstrukturen Lsg. (1)

a) zwei Gleitpunktzahlen im Dialog einzulesen und die größere Zahl von der Kleineren zu subtrahieren und das Ergebnis auszugeben.

```
double zahl1; double zahl2;
cout << "Bitte geben Sie die erste Zahl ein:";
cin >> zahl1;
cout << "Bitte geben Sie die zweite Zahl ein:";
cin >> zahl2;
if( zahl1 < zahl2 ) {
  cout << endl << "Ergebnis: " << zahl1 - zahl2 << endl;
} else {
  cout << endl << "Ergebnis: " << zahl2 - zahl1 << endl;
}
```

Aufg. 2.04: Kontrollstrukturen Lsg. (2)

b) Gleitpunktzahlen einzulesen und aufzusummieren, bis ihre Summe 100 überschreitet.

```
double x = 0.0; double sum = 0.0;
while( cin >> x && ( sum += x ) < 100.0 )
    ;
```

Alternative Lösung:

```
double x = 0.0;
double sum = 0.0;

while( sum <= 100.0 ) {
  cin >> x;
  sum = sum + x;
}
```

c) positive Ganzahlen solange einzulesen, bis eine negative Zahl eingegeben wird.

```
double n = 0;
while( cin >> n && n >= 0 )
    ;
```

cin hat einen Rückgabewert, welcher angibt, ob das Einlesen erfolgreich war bzw. ob der eingelesene Wert zum Variablentyp passt

Aufg. 2.04: Kontrollstrukturen Lsg. (3)

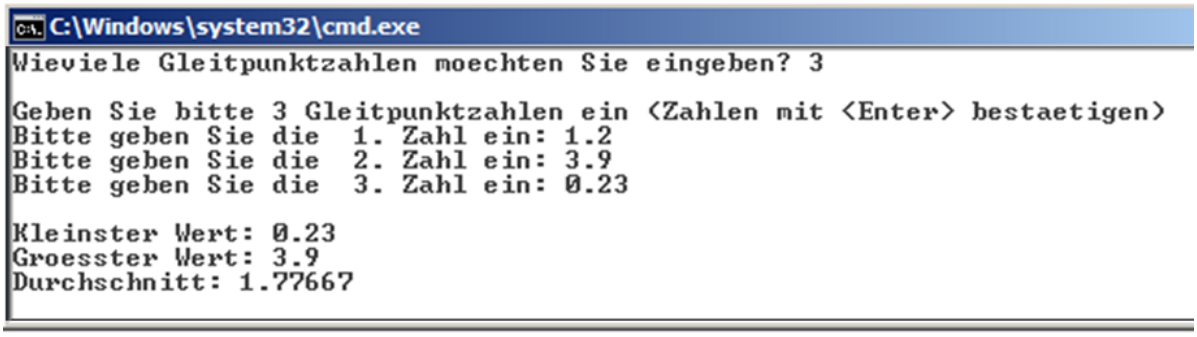
d) den Anwender aufzufordern, drei verschiedene Ganzzahlen einzugeben. Die Aufforderung wird wiederholt, solange zwei von drei Zahlen übereinstimmen.

```
int n1 = 1; int n2 = 2; int n3 = 3;
cout << "Geben Sie drei verschiedene Ganzzahlen ein. " << endl;
cout << "Die erste Ganzzahl:    ";
cin >> n1;
do {
  cout << "\nDie zweite Ganzzahl: ";
  cin >> n2;
} while( n1 == n2 );
do {
  cout << "\nDie dritte Ganzzahl:  ";
  cin >> n3;
} while( n1 == n3 || n2 == n3 );
cout << "Ihre Eingabe: " << n1 << "   " << n2 << "   " << n3 << endl;
```

Aufg. 2.05: Arrays und Schleifen

- Erstellen Sie ein C++ Programm, das im Dialog bis zu 50 Gleitpunktzahlen in ein Array mit Elementen von Typ double einliest. Das Programm sucht anschließend den größten und kleinsten Wert und berechnet den Durchschnitt aller Arrayelement. Die Ergebnisse werden auf dem Bildschirm ausgegeben.

- Beispielausgabe:



Aufg. 2.05: Arrays und Schleifen Lsg. (1)

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

const int max_anzahl = 50;

int main() {
  double werte[max_anzahl];
  int iMin = 0, iMax = 0, anzahl = 0;
  double sum = 0.0;

  cout << "Wieviele Gleitpunktzahlen moechten Sie eingeben? ";
  cin >> anzahl;

  if( anzahl > 50 ) {
    cout << "Anzahl zu gross - maximale Anzahl: " << max_anzahl;
    return 1;
  }

  cout << endl << "Geben Sie bitte " << anzahl << " Gleitpunktzahlen ein
                  (Zahlen mit <Enter> bestaetigen)" << endl;
```

Globale Konstante zur Angabe der maximalen Speicherplätze im Array

Indizes, Zähler

Überprüfung der Einhaltung der Grenze

Aufg. 2.05: Arrays und Schleifen Lsg. (2)

```
  for( int i = 0; i < anzahl; i++ ) {
    cout << "Bitte geben Sie die " << setw( 2 ) << i + 1 << ". Zahl ein: ";
    cin >> werte[i];
    sum += werte[i];
  }

  for( int i = 1; i < anzahl; ++i ) {
    if( werte[i] > werte[iMax] ) {
      iMax = i;
    }
    if( werte[i] < werte[iMin] ) {
      iMin = i;
    }
  }

  cout << endl << "Kleinster Wert:   " << werte[iMin]
       << "\nGroesster Wert:   " << werte[iMax]
       << "\nDurchschnitt:     " << sum / anzahl << endl;

  return 0;
}
```

Liest die Zahlen ein und berechnet die Summe

Ermittelt das Minimum und das Maximum

Ausgabe der Ergebnisse

Aufg. 2.06: Mehrdimensionale Arrays & Schleifen

- Erstellen Sie ein C++ Programm, das im Dialog eine 3x3 Matrix einliest, ihre Determinante berechnet und das Ergebnis auf dem Bildschirm ausgibt. Verwenden Sie zur internen Speicherung der Matrix ein mehrdimensionales Array.

- Beispielausgabe:

```
C:\Windows\system32\cmd.exe
Zeile 1, Spalte 1: 11
Zeile 1, Spalte 2: 12
Zeile 1, Spalte 3: -13
Zeile 2, Spalte 1: 21
Zeile 2, Spalte 2: -22
Zeile 2, Spalte 3: 23
Zeile 3, Spalte 1: -31
Zeile 3, Spalte 2: 32
Zeile 3, Spalte 3: 33

Eingegebene Matrix:
11      12      -13
21      -22     23
-31     32      33

Die Determinante betraegt: -32824
```

Aufg. 2.06: Mehrdim. Arrays & Schleifen Lsg. (1)

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

int main() {
  double matrix[4][4] = { { 0 } };
  /* Einlesen der Matrix */
  for( int zeile = 1; zeile <= 3; ++zeile ) {
    for( int spalte = 1; spalte <= 3; ++spalte ) {
      cout << "Zeile " << zeile << ", Spalte " << spalte << ": " ;
      cin >> matrix[zeile][spalte];
    }
  }
  /* optionale Ausgabe der Matrix, hilfreich zum Ueberpruefen mit
   * http://www.arndt-bruenner.de/mathe/determinanten.htm */
  cout << "\nEingegebene Matrix:\n";
  for( int zeile = 1; zeile <= 3; ++zeile ) {
    for( int spalte = 1; spalte <= 3; ++spalte ) {
      cout << matrix[zeile][spalte] << "\t";
    }
    cout << "\n";
  }
```

2-dim. Array zur Speicherung der eingegebenen Matrix

Speichern der Daten im Array ab Index 1

Aufg. 2.06: Mehrdim. Arrays & Schleifen Lsg. (2)

```
  /* Berechnung */
  double determinante = matrix[1][1] * matrix[2][2] * matrix[3][3];
  determinante += matrix[1][2] * matrix[2][3] * matrix[3][1];
  determinante += matrix[1][3] * matrix[2][1] * matrix[3][2];
  determinante -= matrix[1][3] * matrix[2][2] * matrix[3][1];
  determinante -= matrix[1][2] * matrix[2][1] * matrix[3][3];
  determinante -= matrix[1][1] * matrix[2][3] * matrix[3][2];

  /* Ausgabe */
  cout << "\nDie Determinante betraegt: " << determinante << "\n\n";

  return 0;
}
```