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

Übung04: Informationstechnik (IT)

Tobias Schwalb & Timo Sandmann & Stephan Werner

Institut für Technik der Informationsverarbeitung (ITIV)

Teil1: Besprechung der Übungsaufgaben 3.01-3.07

Inhalt: Übung04 – Teil 1

Karlsruher Institut für Technologie

- Aufg. 3.01: Verständnisfragen
- Aufg. 3.02: Funktionen – Deklaration, Prototypen, Aufruf
- Aufg. 3.03: Funktionen und Header-Dateien
- Aufg. 3.04: Programmstrukturen
- Aufg. 3.05: Gültigkeitsbereich
- Aufg. 3.06: Referenzen und Zeiger
- Aufg. 3.07: Arrays und Zeigerarithmetik

Aufg. 3.01: Verständnisfragen Lsg. (1)

a) Ein Funktionsaufruf ist ein Ausdruck, dessen Typ bestimmt ist durch
   i. die an die Funktion übergebenen Argumente. Falsch
   ii. die im Funktionskopf deklarierten Parameter. Falsch
   iii. den Return-Wert der Funktion. Richtig

b) Der Prototyp einer Funktion stellt dem Compiler Informationen über
   i. den Return-Typ der Funktion bereit. Richtig
   ii. die Namen der Parameter bereit. Falsch
   iii. den Typ jedes Parameters bereit. Richtig

c) Ein Compiler erkennt eine falsche Anzahl von Argumenten nicht. Falsch

Aufg. 3.01: Verständnisfragen Lsg. (2)

d) In C++ ist eine Standard-Header-Datei
   i. eine Objektdatei. Falsch
   ii. eine ausführbare Datei. Falsch
   iii. eine Textdatei. Richtig

e) Zur Ausführung der Anweisung `cout << "Himm…!" << endl;` genügt es, die Header-Datei `iostream` in Ihrem Programm zu inkludieren. Falsch

f) Eine außerhalb einer Funktion definierte Variable wird als global bezeichnet.

g) Eine Funktion kann innerhalb einer anderen Funktion definiert werden. Falsch

Aufg. 3.02: Funktionen - … Lsg. (1)

a) Bestimmen Sie die Fehler in folgenden Prototypen:

i.
```
double calculate double x, double y;
```
Lösung:
```
double calculate( double x, double y );
```

ii.
```
void myFunc( int n, m );
```
Lösung:
```
void myFunc( int n, int m );
```

iii.
```
int your-Func();
```
Lösung:
```
int your_Func();
```

iv.
```
Bool test( void );
```
Lösung:
```
bool test( void );
```


Aufg. 3.02: Funktionen - … Lsg. (2)

b) Welche der folgenden Funktionsaufrufe sind korrekt? Wenn der Funktionsaufruf nicht korrekt ist beschreiben Sie den Fehler.

i.
```
int max( int a, int b, int c );
int result = max( 7, 12 );
```
Lösung: Die Anzahl der Parameter im Prototyp stimmt nicht mit der Anzahl Argumente beim Aufruf der Funktion überein.

ii.
```
double square( double wert );
double x = 2.1;
cout << square( x );
```
Lösung: Richtig

Aufg. 3.02: Funktionen - … Lsg. (3)

b) Welche der folgenden Funktionsaufrufe sind korrekt? Wenn der Funktionsaufruf nicht korrekt ist beschreiben Sie den Fehler.

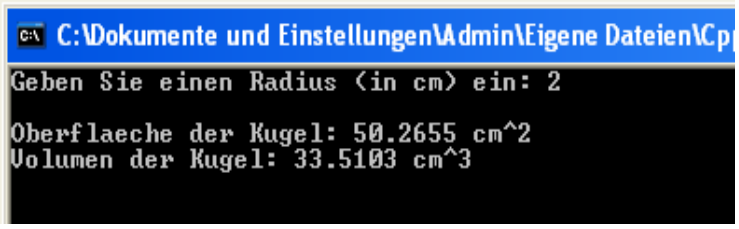
iii.
```
int random( void );
random( 1 );
```
Lösung: Unzulässig. Die Funktion `random()` erwartet keine Argumente.

iv.
```
int random( int a );
random( 1 );
```
Lösung: Richtig. Auch wenn die Funktion einen Rückgabewert hat, muss dieser nicht verwendet / zugewiesen werden.

Aufg. 3.03: Funktionen und Header-Dateien

- Erstellen Sie ein Programm, das die Oberfläche und das Volumen einer Kugel mit dem Radius r berechnet. Dazu soll der Radius über die Tastatur eingelesen werden und später das Ergebnis auf dem Bildschirm ausgegeben werden.

- Die Konstante pi (= 3,1415927) und die zwei Funktionen, die jeweils die Oberfläche und das Volumen berechnen, sollen jeweils in eigenen Datei implementiert werden. Die zugehörigen Headerdateien wiederum sollen in die Hauptdatei eingebunden werden.



$$O(r) = 4 \cdot pi \cdot r^2$$

$$V(r) = \frac{4}{3} \cdot pi \cdot r^3$$

Aufg. 3.03: Funktionen + Header Lsg. (1)

main.cpp

```
#include <iostream>
#include "kugel.h"

using namespace std;

int main() {
  double r = 0;

  cout << "Geben Sie einen Radius (in cm) ein: ";
  cin >> r;

  if( cin == false ) {
    cout << "Ihre Eingabe ist nicht korrekt!" << endl;
  } else if( r < 0 ) {
    cout << "Sie haben einen negativen Wert eingegeben!" << endl;
  } else {
    cout << "Oberflaeche ist: " << oberflaeche( r ) << " cm2" << endl;
    cout << "Volumen ist: " << volumen( r ) << " cm3" << endl;
  }
  return 0;
}
```

Radius abfragen und einlesen

Eingaben überprüfen – ungültig / negativ
Funktionen aufrufen / Ergebnisse ausgeben

Aufg. 3.03: Funktionen + Header Lsg. (2)

kugel.h

```
#ifndef _KUGEL_
#define _KUGEL_

#include <cmath>
const double PI = 3.1415927;

double oberflaeche( double a );
double volumen( double b );

#endif
```

Konstante

Prototypen

kugel.cpp

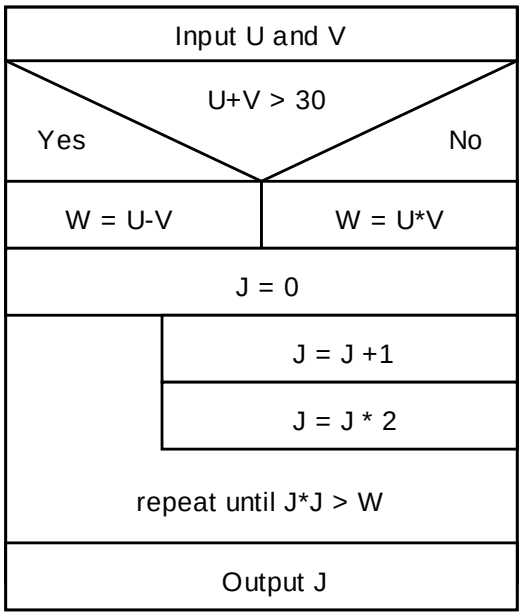
```
#include "kugel.h"

double oberflaeche( double a ) {
  double erg = 0;
  erg = 4 * PI * pow( a, 2.0 );
  return erg;
}

double volumen( double b ) {
  double erg = 0;
  erg = 4.0 / 3 * PI * pow( b, 3.0 );
  return erg;
}
```

Implementierung

Funktion in `<cmath>` für x^y

Aufg. 3.04: Programmstrukturen Lsg.

- Ein Algorithmus ist festgelegt durch das folgende Nassi-Shneiderman-Diagramm



Bestimmen Sie den Ausgang J für die folgenden Eingänge (es soll kein Programm geschrieben werden):

a) U = 20, V = 5
Lösung: 14

b) U = 30, V = 30
Lösung: 2

c) U = 19, V = 2
Lösung: 14

Aufg. 3.05: Gültigkeitsbereich Lsg.

fq, globale Variable: steht im ganzen Programm allen Funktionen zur Verfügung

```
double fq = 0.6180339887;   //Fibonacci-Quotient
int b = 10000;
```

b, globale Variable: steht im ganzen Programm allen Funktionen zur Verfügung

x, Übergabeparameter → lokale Variable: nur innerhalb der Funktion **hash()** verfügbar

```
int hash ( unsinged int x ) {
  double temp = x * fq - (int)( x * fq );
  return b * temp;
}
```

temp, lokale Variable: nur innerhalb der Funktion **hash()** verfügbar

Wegen 0 ≤ **temp** < 1 und **b** = 10000
→ 0 ≤ **b * temp** ≤ 9999

Aufg. 3.06: Referenzen und Zeiger

a) Beschreiben Sie jeweils die Aufgabe der entsprechenden Zeilen des unten stehenden Programms.

b) Was erzeugt das Programm für eine Ausgabe? Bitte überlegen Sie bevor das Programm direkt kompilieren und sich die Ergebnisse anzeigen lassen.

c) Welche zwei Besonderheiten fallen Ihnen in Bezug auf die letzten beiden Zeilen **`++ptr`** und Folgende auf.

Aufg. 3.06: Referenzen & Zeiger Lsg. (1)

a) Beschreiben Sie jeweils die Aufgabe der entsprechenden Zeilen des unten stehenden Programms.

```
int var = 256;
int& ref = var;
int* ptr = &var;

//Referenzen
++ref;
cout << var << "\t\t" << ref << endl;

var += 64;
cout << var << "\t\t" << ref << endl;

cout << &var << "\t" << &ref << endl << endl;
```

- Referenz um eins inkrementieren
- Inhalt der Variablen und Referenz ausgeben
- Variable um 64 erhöhen
- Inhalt der Variablen und Referenz ausgeben
- Adresse der Variablen und Referenz ausgeben

Aufg. 3.06: Referenzen & Zeiger Lsg. (2)

a) Beschreiben Sie jeweils die Aufgabe der entsprechenden Zeilen des unten stehenden Programms.

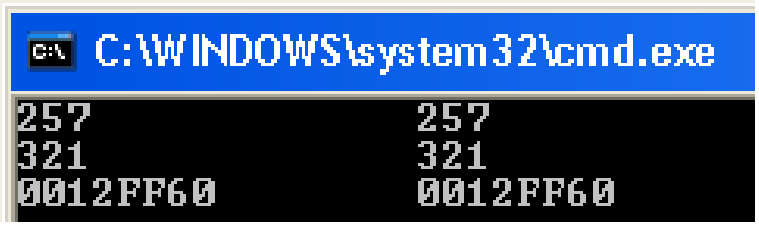
```
//int var = 321;
//int& ref = var;
//int* ptr = &var;

++var;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;

*ptr += 12;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;

cout << &var << "\t" << &ptr << endl << endl;

ptr++;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;
```

- Variable um eins inkrementieren
- Inhalt der Variablen ausgeben
- Inhalt der Adressvariablen ausgeben
- Inhalt des Ziels der Adressvar. ausgeben
- Worauf die Adressvar. zeigt um 12 erhöhen
- Adresse der Variablen und Adressvariablen ausgeben
- Adressvariable um eins inkrementieren

Aufg. 3.06: Referenzen & Zeiger Lsg. (3)

b) Was erzeugt das Programm für eine Ausgabe? Bitte überlegen Sie bevor das Programm direkt kompilieren und sich die Ergebnisse anzeigen lassen.

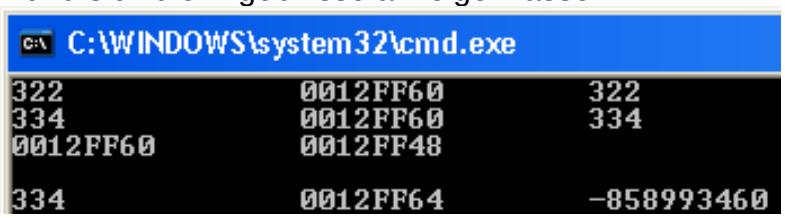
```
int var = 256;
int& ref = var;
int* ptr = &var;

//Referenzen
++ref;
cout << var << "\t\t" << ref << endl;

var += 64;
cout << var << "\t\t" << ref << endl;

cout << &var << "\t" << &ref << endl << endl;
```



Eine Referenz ist nur ein anderer Name für die Variable,
daher sind alle Werte gleich !

Aufg. 3.06: Referenzen & Zeiger Lsg. (4)

b) Was erzeugt das Programm für eine Ausgabe? Bitte überlegen Sie bevor das Programm direkt kompilieren und sich die Ergebnisse anzeigen lassen.




```
//int var = 321;
//int& ref = var;
//int* ptr = &var;

++var;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;

*ptr += 12;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;

cout << &var << "\t" << &ptr << endl << endl;

ptr++;
cout << var << "\t\t" << ptr << "\t" << *ptr << endl;
```

Die Adressvariable ist eigenständig und erreicht die Variable nur über den Verweisoperator !

Aufg. 3.06: Referenzen & Zeiger Lsg. (5)

c) Welche zwei Besonderheiten fallen Ihnen in Bezug auf die letzten beiden Zeilen **`++ptr`** und Folgende auf.

Antwort:

`++ptr` inkrementiert den Inhalt der Adressvariablen um 4, was vier Byte entspricht. Der Grund hierfür besteht darin, dass **`++ptr`** ein Zeiger auf **`int`** ist und Integer bei einem PC 4 Bytes an Speicher einnehmen.

Da die Adressvariable nun auf eine Adresse 4 Byte weiter zeigt und niemand in diesem Fall weiß, was dort gespeichert ist, kann man die Ausgabe nicht vorhersagen.

Das Programm gibt in diesem Fall keine Fehlermeldung aus. Es geht einfach an die entsprechende Speicherstelle, ließt die nächsten 4 Byte (da Integer), interpretiert diese als Integer und gibt das erhaltene Ergebnis aus. → Führt zu Fehlverhalten

Aufg. 3.07: Arrays und Zeigerarithmetik

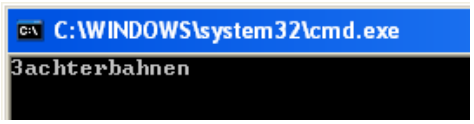
- Was erzeugt das folgende Programm für eine Ausgabe auf dem Bildschirm?

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

int main() {
  char arr[] = "Informationstechnikbuch";
  char* p = arr;
  char* q = arr + 6;

  cout << q - p - 3 << *q;
  p = q++ + 8;

  while( p < q + 9 ) {
    cout << *p++;
  }

  cout << *q << *( p - 3 ) << *( arr + 4 );

  p = arr + 19;
  cout << *p << *--q << *( arr + sizeof( arr ) - 2 ) << "nen";

  return 0;
}
```


Aufg. 3.07: Arrays und Zeigerarithmetik Lsg.

- Was erzeugt das folgende Programm für eine Ausgabe auf dem Bildschirm?

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

int main() {
  char arr[] = "Informationstechnikbuch";
  char* p = arr;
  char* q = arr + 6;

  cout << q - p - 3 << *q;
  p = q++ + 8;

  while( p < q + 9 ) {
    cout << *p++;
  }

  cout << *q << *( p - 3 ) << *( arr + 4 );

  p = arr + 19;
  cout << *p << *--q << *( arr + sizeof( arr ) - 2 ) << "nen";

  return 0;
}
```



3a
ch
t
e
r
b
a
h
nen