Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Übung 03: Informationstechnik (IT)

Marc Weber

Institutsleitung
Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

Teil 1: Besprechung Aufgabenblatt 3

Aufgabe 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

Aufgabe 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 (`using namespace std;`)

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

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

- Aufg. 3.01: Verständnisfragen

Aufgabe 3.02: Funktionen – Deklaration, Prototypen, Aufruf 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 );
```

Aufgabe 3.02: Funktionen – Deklaration, Prototypen, Aufruf 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

Aufgabe 3.02: Funktionen – Deklaration, Prototypen, Aufruf Lsg. (3)

Karlsruher Institut für Technologie

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

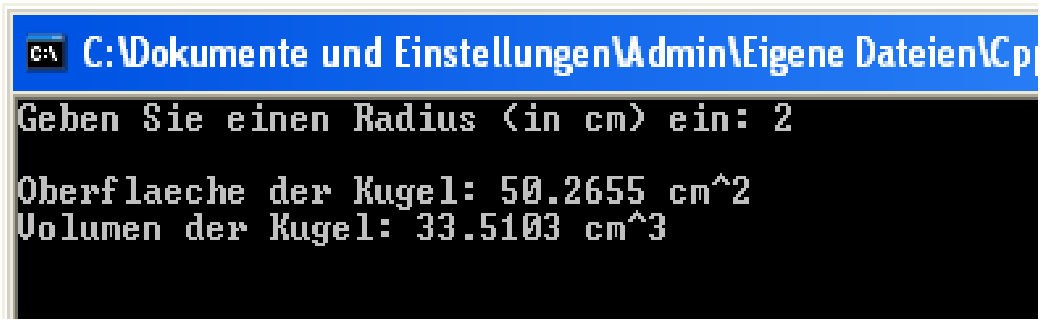
iv.
```
int random( int a );
random( 1 );
```
Lösung: Richtig. Auch wenn die Funktion einen Rückgabewert hat, muss dieser nicht verwendet / zugewiesen werden.

- Aufg. 3.02: Funktionen – Deklaration, Prototypen, Aufruf

Aufgabe 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 in einer Header-Datei `.h` und der zugehörigen Quelldatei `.cpp` implementiert werden. Die Header-Datei wiederum soll in die Hauptdatei des `main`-Programms eingebunden werden.



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

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

Aufgabe 3.03: Funktionen und Header-Dateien

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) {
    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

Aufgabe 3.03: Funktionen und Header-Dateien

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;
}
```

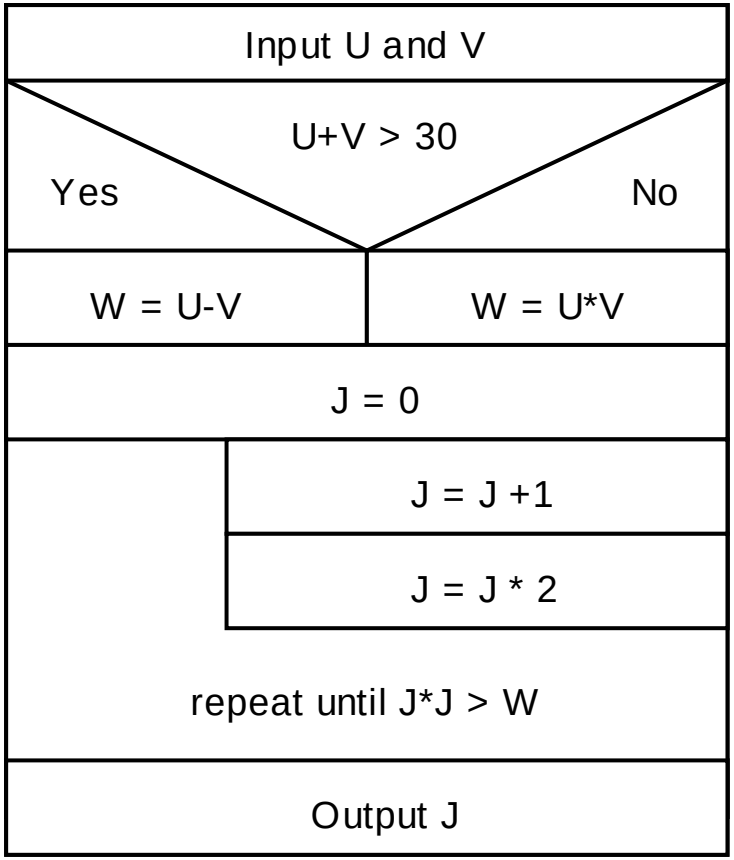
Implementierung

Funktion in **<cmath>** für x^y

- Aufg. 3.03: Funktionen und Header-Dateien

Aufgabe 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. 3.04: Programmstrukturen

Aufgabe 3.05: 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 Sie das Programm direkt kompilieren und sich die Ergebnisse anzeigen lassen.

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

Aufgabe 3.05: Referenzen und 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

Aufgabe 3.05: Referenzen und Zeiger Lsg. (2)

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

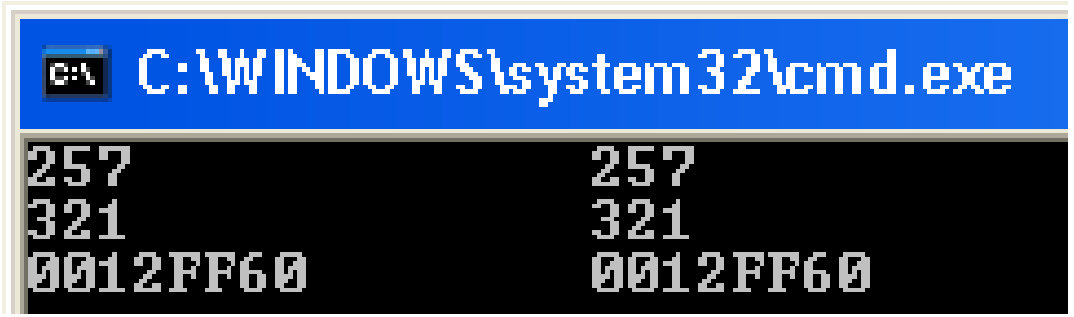
```
//int var = 321;

//Zeiger
++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 Var. und Adressvar. ausgeben
- Adressvariable um eins inkrementieren

Aufgabe 3.05: Referenzen und 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.

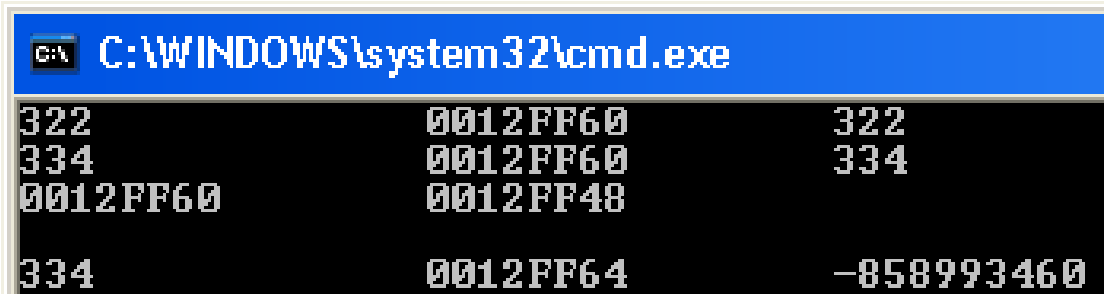
```
int var = 256;
int& ref = var;
int* ptr = &var;

//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!

Aufgabe 3.05: Referenzen und 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;

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

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

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

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

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

Aufgabe 3.05: Referenzen und 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. 3.05: Referenzen und Zeiger

Aufgabe 3.06: Arrays und Zeigerarithmetik

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

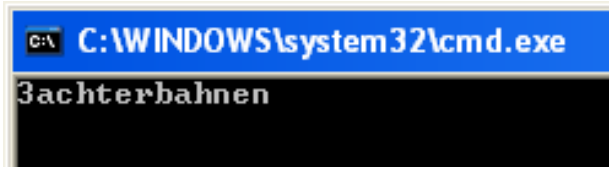
```
#include <iostream>
using namespace std;

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" << endl;

  return 0;
}
```

Aufgabe 3.06: Arrays und Zeigerarithmetik

Lsg. (1)

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

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

int main() {
  char arr[] = "Informationstechnikbuch";
  char* p = arr;              //p = &arr[0]
  char* q = arr + 6;          //q = &arr[6]

  cout << q - p - 3 << *q;
  p = q++ + 8;                //p = &arr[6+8]
                              //hier ist q = &arr[7]

  while( p < q + 9 ) {        //&arr[14] < &arr[7+9]
    cout << *p++;             //cout << arr[14]; p = &arr[15];
  }
  cout << *q << *( p - 3 ) << *( arr + 4 );
  //== cout << arr[7] << arr[16-3] << arr[4];
  p = arr + 19;
  cout << *p << *--q << *( arr + sizeof( arr ) - 2 ) << "nen";
  //== cout << arr[19] << arr[6] << arr[22] << "nen";
  return 0;
}
```



Schleife wird wiederholt, solange p < &arr[16]

sizeof() gibt Anzahl der Bytes zurück bzw. bei char-Array, Anzahl der Zeichen (inkl. '\0')

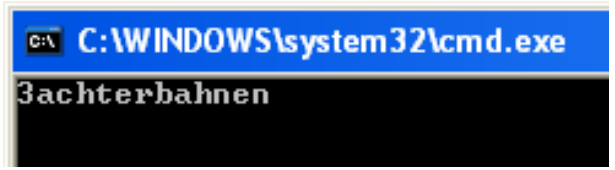
Aufgabe 3.06: Arrays und Zeigerarithmetik

Lsg. (2)

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;
}
```

C-String! Letztes Zeichen `'\0'`!



- 3a
- ch
- t
- e
- r
- b
- a
- arr[24-2] = h
- nen

- Aufg. 3.06: Arrays und Zeigerarithmetik

Vielen Dank für Ihre Aufmerksamkeit

Marc Weber
Karlsruher Institut für Technology (KIT) – ITIV
marc.weber3@kit.edu

