Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

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

Übung 03: Informationstechnik (IT)

Daniel Grimm

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

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

f) Bei der Ausführung der Anweisungen:

```
string name;
cin >> name;
```

werden Zeichen von der Standardeingabe eingelesen und zwar

i. alle Zeichen einer Zeile ohne führende Zwischenraumzeichen → Falsch
ii. genau ein Wort ohne führende Zwischenraumzeichen → Richtig
iii. eine ganz Textzeile → Falsch

g) Zum verketten zweier Objekte vom Typ **`string`** kann man den Operator **`+`** oder **`+=`** verwenden.

h) Das erste Zeichen in einem String hat die Position **`0`**.

i) Für den Zugriff auf die einzelnen Zeichen in einem **`string`**-Objekt kann der Operator **`[]`** verwendet werden.

- 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 Referenzen

a) Was gibt das folgende Programm auf der Konsole aus?

```
int main() {
int a = 3;
int& b = a; //Referenz zu a
cout << b << "\n";
a = 18;
cout << b << "\n";
b = 25;
cout << a << "\n";
}
```

Referenz zu a: a wird ausgegeben

a wird geändert: b ändert sich mit

b wird geändert: eigentlich ändert sich a

```
Aufgabe 3.03: Referenzen
----------------------------------------

3
18
25
```

Aufgabe 3.03 Referenzen

b) Welche Fehler werden im folgenden Programm gemacht?

```
int main() {
int& c = 3;
```

Lösung: Eine Referenz kann nur auf eine bereits initialisierte Variable verweisen. Es kann nicht direkt ein Wert zugewiesen werden, da der Speicher bereits reserviert sein muss.

```
bool d = false;
int& e = d;
```

Lösung: Eine Integer-Referenz kann nur auf Integer-Variablen verweisen. Ein Verweis auf eine Bool-Variable ist nicht möglich.

```
}
```

- Aufg. 3.03: Referenzen

Aufgabe 3.04: 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 3.04: 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 3.04: Referenzen und Zeiger Lsg. (2)

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

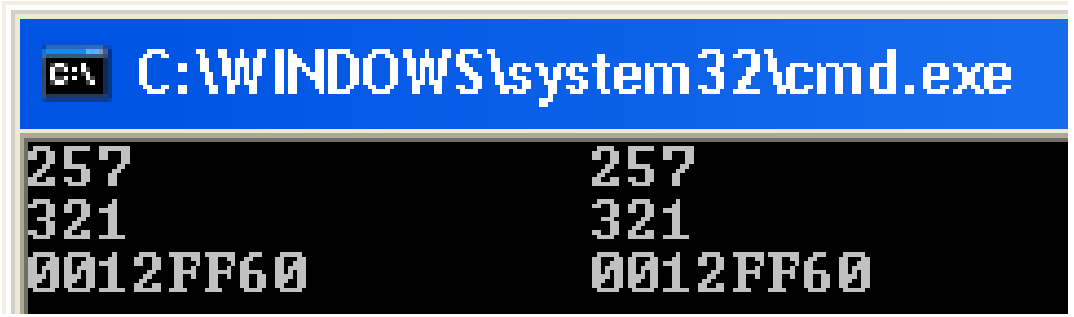
```
//int var = 321;

//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 3.04: 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.

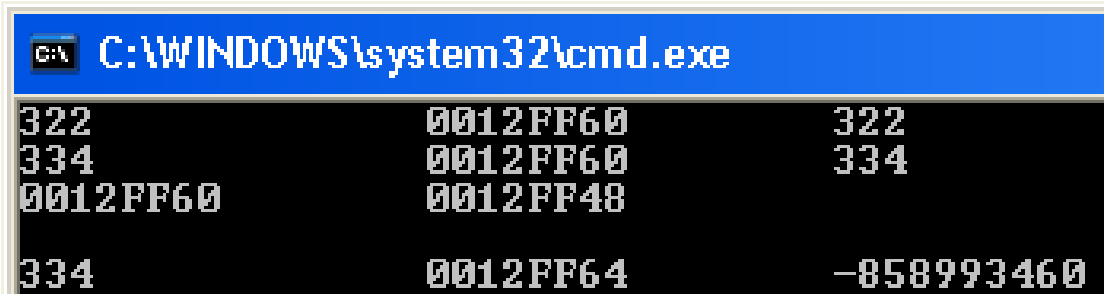
```
int var = 256;
int& ref = var;
int* ptr = &var;

//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 3.04: 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 3.04: 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. 3.04: Referenzen und Zeiger

Aufgabe 3.05: 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;
}
```

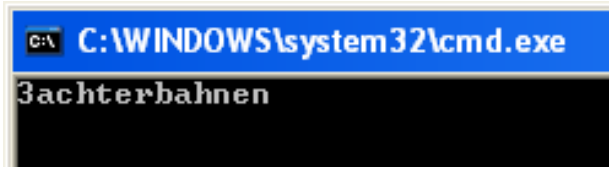
Aufgabe 3.05: Arrays und Zeigerarithmetik

Lsg. (1)

Karlsruher Institut für Technologie

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')

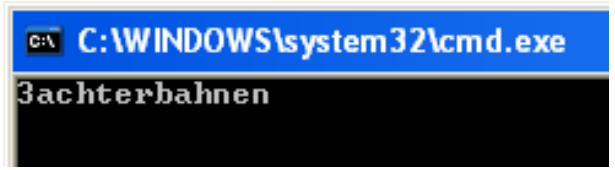
Aufgabe 3.05: Arrays und Zeigerarithmetik

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. 3.05: Arrays und Zeigerarithmetik

Aufgabe 3.06 Strings

Schreiben Sie die Funktion **palindrom**, die feststellt, ob der gegebene String **str_pali** ein Palindrom ist. Ein Palindrom kann man von vorne und von hinten lesen. Jedesmal ist der String gleich. Beispiel: Rentner.

Der String str_pali wird in **main** definiert.

Vorbereitung: Wie sieht die Eingabe- und Ausgabeschnittstelle zur Funktion **palindrom** aus?

Lösung :

Ein/Ausgabe der Funktion `palindrom:`

```
bool palindrom(const char *str_pali, int length_eingabe)
```

oder

```
bool palindrom(string str_pali)
```

Effizienter, da nur der Pointer kopiert werden muss, nicht der gesamte String

Aufgabe 3.06 Strings

```
bool palindrom(const char *str_pali, int length_eingabe)
{
  char a, b;
  for (int i = 0; i < length_eingabe; i++) {
    a = tolower(str_pali[i]);
    b = tolower(str_pali[length_eingabe - i - 1]);
    if (a != b)
    {
      return false;
    }
  }
  return true;
}
```

Aufgabe 3.06 Strings

```
int main(void) {
  string eingabe;
  bool flag;
  cout << "Geben Sie ein Wort ein: " << endl;
  cin >> eingabe;
  int length_eingabe = eingabe.length();

  flag = palindrom(eingabe.c_str(), length_eingabe);

  if (flag) {
    cout << "Glueckwunsch, das ist ein Palindrom!" << endl;
  }
  else
  {
    cout << "Das ist kein Palindrom!" << endl;
  }
  return 0;
}
```

Vielen Dank für Ihre Aufmerksamkeit

Daniel Grimm
Karlsruher Institut für Technology (KIT) – ITIV
daniel.grimm@kit.edu