Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang Sensorik/Sensorsystemtechnik

Note / normierte Punkte

Klausur in Programmieren

Wintersemester 2010/11, 17. Februar 2011

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:

MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	12	8	20	20	25	15	100
Punkte							

Alle Fragen beziehen sich auf den Stoff der Vorlesung. Somit sind sie z.B. bezogen auf die Programmiersprache C++. Auch sonst gelten die Konventionen wie in unserer Vorlesung.

1. Aufgabe: Grundlagen

Welche Werte haben die angegebenen Variablen? (12P)

a) int x = 21 / 4; — x = 5 — (1 P)

b) double y = 19.0 / 3; — y = 6.3333 — (1 P)

c) int diff = 'B' – 'A' ; — diff = 1 — (2 P)

d) int x = 1 ;
float f = ++x * 2.1 ; — f = 4.2, x = 2 — (2 P)

e) char c = 200;
double d = (c++ % 100) * 3.141 ; — d = 0.0, c = 201 — (3 P)

f) char* acString = "HELLO FOLKS!";
char cLetter = acString[6] – 'A'; — cLetter = 5 — (3 P)

2. Aufgabe: Grundlagen

Schreiben Sie ein vollständiges und lauffähiges C++-Hauptprogramm, das die Kantenlängen eines gleichschenkligen und rechtwinkligen Dreiecks (der rechte Winkel wird von den gleich langen Schenkeln eingeschlossen) als reellen Wert von der Konsole (über cin) einliest. Berechnen Sie anschließend die Fläche und geben Sie alle Winkelwerte des Dreiecks aus. (8 P)

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

void main()
{
  double dKante;
  cout << "Kantenlaenge: ";
  cin >> dKante;

  cout << "Flaeche: " << dKante * dKante / 2.0;
  cout << "alpha: 90"<< endl
       << "beta: 45" << endl
       << "gamma: 45" << endl;
}
```

3. Aufgabe: Bedingungen und Schleifen

a) Welche Bedingungen gibt es in C/C++. Geben Sie auch ein Beispiel für Bedingungen beim Präcompiler. Erklären Sie die Funktionsweise anhand von Beispielen. (11P)

```
if(a == 12)
{
   cout << “x“;
}
else
{
   cout << “y“;
}
```

Wenn a den Wert 12 hat wird x ausgegeben, andernfalls y
Eine if-Bedingung erlaubt die Ausführung von Programmcode in Abhängigkeit von der Auswertung der Bedingung. Ist die Auswertung der Bedingung true, wird der if-Zweig, andernfalls der else-Zweig ausgeführt. Die Auswertung erfolgt zur Laufzeit.

```
switch(a)
{
   case 12:  cout << “x“;
             break;
   default: cout << “y“
}
```

switch ist eine Mehrfachselektion mit der mehrere Variablen-werte überprüft werden können
Die Auswertung erfolgt zur Laufzeit.

```
#ifdef LINUX
cout << “LINUX-Sytem“
#else
cout << “kein LINUX-Sytem“
#endif
```

Die #ifdef Bedingung ist erfüllt, wenn das Wort LINUX definiert ist. Die Auswertung erfolgt zum Zeitpunkt des Kompilierens.

b) Welche Schleifenarten gibt es in C/C++? Geben Sie auch hier kleine Beispiele. (9P)

Kopfgetestete Schleife

```
int a = 0;
while(a != 12)
{
   ++a;
}
```

Endegetestete Schleife

```
int a = 0;
do
{
   ++a;
}
while(a != 12);
```

Zählschleife

```
for(int a=0;a < 12; ++a)
{
   cout << a;
}
```

4. Aufgabe: Arrays, Strukturen

a) Erklären Sie den Unterschied zwischen einem Array und einer Struktur. (4 P)

Ein Array fasst Variablen desselben Datentyps zusammen.

Eine Struktur fasst Variablen mit beliebigem Datentyp zusammen.

b) Definieren Sie eine Variable, die ein dynamisch angelegtes Array der Größe 20 mit einem Felddatentyp double an. Weisen Sie dem Feldelement mit Index 5 den Wert 7.5 zu und weisen Sie anschließend dem Feldelement mit Index 6 den Inhalt des Feldelements mit Index 5 zu (damit ist nicht der konstante Wert 7.5 gemeint!). Schreiben Sie eine Schleife, die nur die Werte des Arrays in jeweils einer eigenen Zeile auf die Konsole ausgibt (sie können voraussetzen, dass die Bibliothek bereits eingebunden wurde). Geben Sie als letzten Schritt das dynamisch angelegte Array wieder frei. (10 P)

```
double* adArray = new double [20];
adArray[5] = 7.5;
adArray[6] = adArray[5];
for(int li=0; li<20; ++li)
{
  cout << adArray[li] << endl;
}
delete [] adArray;
```

c) Deklarieren Sie eine Struktur mit dem Namen Values. Die Struktur soll eine double und eine int Variable besitzen. Legen Sie eine Variable mit dem Datentyp Values an und weisen Sie der double-Komponente den Wert 5.4 und der int-Komponente 3 zu (6 P):

```
struct Values
{
   double   mdV1;
   int      miV2;
};

Values   sStruktur;
sStruktur.mdV1 = 5.4;
sStruktur.miV2 = 3;
```

5. Aufgabe: Ganzes Programm / Automatische Tests

Ergänzen Sie die Lücken im nachfolgenden Programmtext zu einem lauffähigen C++-Programm. Es hat die Aufgabe, eine Funktion zu testen – compare –, die zwei Zeichenketten auf Gleichheit überprüft. Die Zeichenkette ist durch eine abschließende 0 begrenzt. Geben Sie zum Schluss an wie viele aufgetretene Fehler angezeigt werden. (25P)

```
#include <iostream>

using namespace std;

bool compare(char* acInString1, char* acInString2)
{
  int li = 0;
  while(acInString1[li] == acInString2[li] &&
        acInString1[li] != 0)
  {
    ++li;
  }
  return acInString1[li] == acInString2[li];
}


bool testCaseCompare(char* acInString1, char* acInString2,
                     bool bInExpectedResult)
{
  bool bResult = compare(acInString1, acInString1);
  cout << "Test: Compare " << acInString1 << " mit "
       << acInString2 << ": " << (bResult? "true " : "false ");

  if(bResult == bInExpectedResult)
  {
    cout << "OK!" << endl;

    return true;
  }
  else
  {
    cout << "NICHT OK! Erwartet: "
             << (bInExpectedResult? "true" : "false") << endl;

    return false;
  }
}
```

```
void testCompare()
{
  int iErrorCount = 0;
  // Gleich lange Zeichenkette, 1 Buchstabe Unterschied am Anfang.

  if(!testCaseCompare("Hallo!", “Kallo!“, false))
  {
    iErrorCount++;
  }
  // Gleich lange Zeichenkette, 1 Buchstabe Unterschied in der
  // Mitte.

  if(!testCaseCompare("Hallo!", “Halzo!”, false))
  {
    iErrorCount++;
  }
  // Gleich lange Zeichenkette, 1 Buchstabe Unterschied am Ende.

  if(!testCaseCompare("Hallo!", “Hallo?“, false))
  {
    iErrorCount++;
  }
  // Unterschiedlich lange Zeichenkette, Anfang gleich.
  if(!testCaseCompare("Hallo", "Halloallerseits!", false))
  {
    iErrorCount++;
  }
  // Gleiche Zeichenkette.
  if(!testCaseCompare("Hallo!", "Hallo!", false))
  {
    iErrorCount++;
  }
  cout << endl;
  cout << "Aufgetretene Fehler gesamt: " << iErrorCount << endl;
}

int main()
{
  testCompare();

  cin.get();
}
```

```
Aufgetretene Fehler gesamt: 1
```

6. Aufgabe: Algorithmus

Was machen die nachfolgenden Funktionen unknown1 und unknown2? Welchen Wert erhält die Variable dValue in der main-Routine als Ergebnis?

Bitte beschreiben Sie die Funktionsweise möglichst abstrakt. (15 P)

Hinweis: Testen Sie den Algorithmus, in dem Sie das Programm von Hand Schritt für Schritt ausführen und dabei die Variablenwerte beobachten.

```
double unknown1(double* adValues, int iInCount)
{
  double dValue = 0.0;
  for(int li=0; li<iInCount; ++li)
  {
    dValue += adValues[li];
  }
  return dValue / iInCount;
}

void unknown2(double*& radValues, int iInCount)
{
  radValues = new double[iInCount];
  for(int li=0; li<iInCount; ++li)
  {
    radValues[li] = li + 1;
  }
}

int main()
{
  const int ciCount = 10;
  double* adValues;
  unknown2(adValues, ciCount);
  double dValue = unknown1(adValues, ciCount);
  delete [] adValues;
}
```

unknown1 berechnet den Mittelwert der Feldelemente des Arrays adValues.

unknown2 legt ein Array mit der gegebenen Größe an und initialisiert es aufsteigend mit 1, 2, 3 usw.

Der Wert von dValue wird hier zu 5.5 berechnet.