Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang Sensorik/Sensorsystemtechnik

Note / normierte Punkte

Klausur in Programmieren

Sommer 2011, 20. Juli 2011

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:

MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	12	20	10	25	18	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

Stellen Sie die nachfolgenden Zahlen in den jeweiligen Zahlensystemen dar (12P):

a) 1010 1011 1100 0111 (binär)	hex:	0xABC7	(1P)
	oktal:	0125707	(3P)
b) 231 (dezimal)	hex:	0xE7	(2P)
	binär:	1110 0111	(2P)
c) 7B (hex)	dezimal:	123	(3P)
	binär:	0111 1011	(1P)

2. Aufgabe: Grundlagen

Schreiben Sie ein Hauptprogramm, das eine Schleife enthält und so lange reelle Werte in die Variable x von der Konsole einliest, bis der Wert 0 eingegeben wird. Bestimmen Sie innerhalb der Schleife mit diesem Wert den Funktionswert der Funktion $y = x^2 + 2/x + 3$ und geben Sie diesen reellen Wert auf die Konsole aus. Schreiben Sie das Programm so, dass für den Funktionswert immer definierte Werte berechnet werden (Hinweis: 25 / 0 ist z.B. nicht definiert). (20P)

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

void main()
{
  double x;
  do
  {
    cout << "x = ";
    cin >> x;
    if(x != 0.0)
    {
      double y = x * x + 2.0/x + 3.0;
      cout << "Funktionswert y = " << y << endl;
    }
  }
  while (x != 0.0);
}
```

3. Aufgabe: Variablen

a) Was versteht man unter einer Variablendeklaration im Unterschied zu einer Variablendefinition?: (4 P)

Eine Variablendefinition ist eine Variablendeklaration, bei der die Variable einen Anfangswert erhält. Man spricht hier von einer Initialisierung der Variablen.

b) Bestimmen Sie den Wert der nachfolgenden Variablen: (6 P)

- int x = 13 / 2; x == _____6_____

- double d = 12.67;
  int x = d; x == _____12_____

- double d = 12.37 ;
  int x = (int)(d + 0.5); x == _____12_____

4. Aufgabe: Funktionen

a) Schreiben Sie eine Funktion, die zwei reelle Parameter a und b besitzt und in der die dritte binomische Formel als Ergebnis berechnet wird. (8 P)
(zur Erinnerung: $(a + b)(a – b) = a^2 - b^2$):

```
double binom(double a, double b)
{
  return a * a - b * b;
}
```

b) Wie nennt man die oben verwendete Parameterübergabe, wenn nur der Wert des Parameters verwendet wird? (1 P):
call by value

c) Wie nennt man eine Parameterübergabe, wenn die Variable selbst übergeben wird? (1 P):
call by reference

3 d) Schreiben Sie eine kleine Funktion, bei der die Parameterübergabe nach dem Schema in Aufgabenteil c) durchgeführt wird und dabei die verwendete Variable im Funktionsrumpf geändert wird. Geben Sie anschließend ein Programmfragment an, bei dem die von Ihnen definierte Funktion aufgerufen wird (d.h. eine Anweisung reicht und bitte kein Hauptprogramm oder irgendwelche Ausgaben schreiben!). (5 P)

```
void callByReference(int& riOutParam)
{
  riOutParam = 4;
}

int iVar;
callByReference(iVar);
```

e) Aus welchen Teilen besteht ein Funktionskopf und wie ist er aufgebaut bzw. gibt es Besonderheiten? (10 P)

1. Datentyp des Rückgabewertes
    a. Hat eine Funktion keinen Rückgabewert, wird als Datentyp void angegeben.
2. Name der Funktion
3. Parameterliste eingeschlossen in runden Klammern
    a. eine Parameterliste enthält keinen, einen oder beliebig viele Parameter,
    b. enthält die Parameterliste mehr als einen Parameter, werden die Parameter durch ein Komma getrennt,
    c. die Definition eines Parameters entspricht einer Variablendefinition,
    d. Parameter können wie Variablen verwendet werden.

Aufbau eines Funktionskopf:
Datentyp Funktionsname (Parameterliste)

5. Aufgabe: Array/Feld, Indizierung, Zeichenketten

a) Gegeben sind zwei Zeichenketten char* acString1 und char* acString2. Beide Zeichenketten schließen mit einer 0 als Kennzeichnung für das Ende der Zeichenkette. Schreiben Sie eine Funktion compareLen, die bestimmt, welche Zeichenkette länger ist. Dabei ist compareLen == 1, wenn acString1 länger ist als acString2, 0 bei gleicher Länge und -1 wenn acString1 kürzer ist als acString2 (Hinweis: Machen Sie sich das Leben leicht und schreiben Sie zunächst eine Funktion strlen, die die Länge der Zeichenkette bestimmt). (10 P)

```
int strlen(char* acInString)
{
  int iLen = 0;
  while(acInString[iLen] != 0)
  {
    iLen++;
  }
  return iLen;
}
```

```
int compareLen(char* acString1, char* acString2)
{
  int iLen1 = strlen(acString1);
  int iLen2 = strlen(acString2);
  if(iLen1 > iLen2)
  {
    return 1;
  }
  if(iLen1 < iLen2)
  {
    return -1;
  }
  return 0;
}
```

b) Schreiben Sie ein Programmfragment, das ein dynamisches Feld der Länge 30 vom Typ char anlegt und initialisieren Sie jedes Feldelement in einer passenden Schleife mit 0. Geben Sie anschließend das Feld wieder frei. Verwenden Sie für die Feldlänge 30 eine geeignete Konstante, die Sie immer dann verwenden, wenn Sie auf die Feldlänge zurückgreifen wollen: (8 P)

```
const int ciArrayLength = 30;

char* acString = new char[ciArrayLength];
for(int li=0; li<ciArrayLength; ++li)
{
  acString[li] = 0;
}

delete [] acString;
```

6. Aufgabe: Algorithmus

Was berechnen die nachfolgenden Funktionen unknown1 und unknown2 bzw. wann liefert unknown2 den Wert true und wann den Wert false?

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

Hinweis: Testen Sie den Algorithmus anhand des gegebenen Hauptprogramms und beobachten Sie die Variablenwerte. Welche Zahlenmenge könnte hier eine Rolle spielen, die in C/C++ nicht als Datentyp existiert?

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

struct Tupel
{
  int    miVar1;
  int    miVar2;
};

void printTupel(Tupel sInParam)
{
  cout << "Tupel: " << sInParam.miVar1 << " / " << sInParam.miVar2 << endl
       << "Wert: " << (double)sInParam.miVar1 / (double)sInParam.miVar2 << endl;
}
```

```
void unknown1(Tupel& sInParam, int iInFaktor)
{
  sInParam.miVar1 *= iInFaktor;
  sInParam.miVar2 *= iInFaktor;
}

bool unknown2(Tupel sInParam1, Tupel sInParam2)
{
  int iFaktor1 = sInParam1.miVar2;
  int iFaktor2 = sInParam2.miVar2;
  unknown1(sInParam1, iFaktor2);
  unknown1(sInParam2, iFaktor1);

  return sInParam1.miVar1 < sInParam2.miVar1;
}

void main()
{
  Tupel sVar1;
  sVar1.miVar1 = 10;
  sVar1.miVar2 = 4;

  Tupel sVar2;
  sVar2.miVar1 = 15;
  sVar2.miVar2 = 2;

  bool bResult = unknown2(sVar1, sVar2);
}
```

Die Struktur Tupel beschreibt Zahlen aus der Menge der Brüche. miVar1 ist dabei der Zähler, miVar2 der Nenner.
Die Funktion unknown1 erweitert einen Bruch.
Die Funktion unknown2 gibt true zurück, wenn Bruch 1 kleiner ist als Bruch 2.
Die Funktion unknown2 gibt false zurück, wenn Bruch 1 größer oder gleich Bruch 2 ist.