Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang Sensorik/Sensorsystemtechnik

Note / normierte Punkte

Klausur in Programmieren

Wintersemester 2011/12, 15. Februar 2012

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:

MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	10	8	21	23	23	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? (10 P)

a) int x = 31 / 4; x = 7 (1 P)

b) double y = 25.0 / 2; y = 12.5 (1 P)

c) int diff = ‘C’ – ‘A’ ; diff = 2 (2 P)

d) int x = 2;
float f = ++x / 3.0 ; f = 1.0, x = 3 (2 P)

e) char c = 0xA0;
double d = (c++ % 160) * 3.141 ; d = 0.0, c = 0xA1 (4 P)

2. Aufgabe: Grundlagen

Schreiben Sie ein vollständiges und lauffähiges C++-Hauptprogramm, das zwei ganzzahlige Werte von der Konsole einliest, Minimum und Maximum ausgibt, sowie den Mittelwert aus beiden Werten bestimmt. Der Mittelwert soll in eine float-Variable gespeichert und dann ausgegeben werden. Berücksichtigen Sie bei der Ausgabe auf die Konsole eine passende Beschriftung (8 P)

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

void main()
{
  int iValue1;
  int iValue2;

  cout << “Wert1: “;
  cin >> iValue1;

  cout << “Wert2: “;
  cin >> iValue2;

  if(iValue1 > iValue2)
  {
    int iTmp = iValue1;
    iValue1 = iValue2;
    iValue2 = iTmp;
  }

  float fMittelwert = (iValue1 + iValue2) / 2.0;

  cout << “Minimum: “ << iValue1 << endl;
  cout << “Maximum: “ << iValue2 << endl;
  cout << “Mittelwert: “ << fMittelwert << endl;
}
```

3. Aufgabe: Funktionen und Schleifen

a) Erklären Sie die **wesentlichen** Unterschiede zwischen den folgenden Funktionsaufrufen in **wenigen** und **kurzen** Sätzen (bei Romanen kann es Punktabzug geben!): (12 P)

1. void functionCall1(int iInValue)
2. void functionCall2(int& riInOutValue)
3. void functionCall3(int* piInOutValue)
4. int functionCall4()

1. call by value-Aufruf, nur der Wert wird als Kopie übergeben
2. call by reference-Aufruf, die Variable selbst wird übergeben und kann geändert werden
3. call-by-reference-Aufruf wie 2. nur über den Zeiger-Mechanismus
4. hier wird keine Variable übergeben, nur ein Funktionswert wird zurückgegeben

b) Welche Schleifenarten gibt es in C/C++? Geben Sie kleine Beispiele als Programm**fragmente** (keine Funktionen, kein Hauptprogramm, keine Ein- oder Ausgabe!). (9 P)

```
int iPos=10;
while(iPos > 0)
{
  --iPos;
}
```

```
int iPos=0;
do
{
  iPos++;
}
while(iPos < 10);
```

```
for(int li=0; li<20; li++)
{
}
```

4. Aufgabe: Arrays, Strukturen

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

Bei einem Array besitzen alle Elemente des Arrays denselben Datentyp, bei einer Struktur können sie unterschiedliche Datentypen besitzen. Der Zugriff bei Feldelementen eines Arrays erfolgt über einen Index, bei einer Struktur erfolgt der Zugriff auf die Strukturelemente über den Punktoperator.

b) Schreiben Sie ein Programm**fragment** (kein Hauptprogramm, keine Funktion), bei dem über die Konsole eine Zahl eingelesen wird, die anschließend die Größe eines integer-Arrays festlegt. Schreiben Sie danach mit einer Schleife in jedes Feldelement eine 0. Sorgen Sie dafür, dass kein Speicherleck entsteht. (10 P)

```
int iSize;
cout << "Feldgröße: ";
cin >> iSize;
int* aiFeld = new int[iSize];
for(int li=0; li<iSize; ++li)
{
  aiFeld[li] = 0;
}
delete [] aiFeld;
```

c) Schreiben Sie eine Struktur mit dem Namen Bruch. Die Struktur soll zwei double-Variablen mit dem Namen mdZaehler, und mdNenner besitzen. Deklarieren Sie eine Variable namens oKaufpreis von diesem Datentyp (keine Funktion, kein Hauptprogramm, keine Ein- oder Ausgabe!) (3 P)

```
struct Bruch
{
  double mdZaehler;
  doube mdNenner;
};

Bruch oKaufpreis;
```

d) Schreiben Sie eine **Funktion** initBruch, die im Funktionskopf die Parameter-Variable rdOutBruch vom Datentyp Bruch mit call-by-reference-Aufruf besitzt. Initialisieren Sie diese Variable im Funktionsrumpf so, dass die Variable mdZaehler den Wert 0.0 und die Variable mdNenner den Wert 1.0 bekommt (kein Hauptprogramm, keine Ein- oder Ausgabe!). (6 P)

```
void initBruch(bruch& rdOutBruch)
{
  rdOutBruch.mdZaehler = 0.0;
  rdOutBruch.mdNenner = 1.0;
}
```

5. Aufgabe: Zeichenketten

a) Schreiben Sie eine Funktion strlength, die die Länge einer mit 0 terminierten Zeichenkette bestimmt. Der Funktionswert soll die Länge zurückgeben (kein Hauptprogramm, keine Ein- oder Ausgabe!). (8 P)

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

b) Schreiben Sie eine Funktion charAt, die bestimmt, an welcher Stelle ein Zeichen in einer Zeichenkette zum ersten Mal vorkommt. Der Funktionswert von charAt bestimmt die Position des ersten Auftretens. Kommt das Zeichen gar nicht vor, soll -1 zurückgegeben werden (kein Hauptprogramm, keine Ein- oder Ausgabe!). (15 P)

```
int charAt(char cInChar, char* acInString)
{
  int iPos = 0;
  while(acInString[iPos] != 0 && acInString[iPos] != cInChar)
  {
    iPos++;
  }
  if(acInString[iPos] != 0)
  {
    return iPos;
  }
  return -1;
}
```

6. Aufgabe: Algorithmus

Was macht die nachfolgende Funktion unknown? Geben sie für die Beispiele

1. acString1 = “all“, acString2 = “Hallo Welt!“
2. acString1 = “ALL“, acString2 = “Hallo Welt!“

an, welcher Wert in der main-Routine als Ergebnis ausgegeben wird.

Beschreiben Sie die Funktionsweise möglichst abstrakt. Kommentierung der Programmzeilen gibt keine Punkte. (15 P)

Hinweis: Testen Sie den Algorithmus, in dem Sie das Programm von Hand Schritt für Schritt ausführen und dabei die Variablenwerte anhand einer Tabelle (Variablen als Spalte, Zeile als neuer, aktueller Wert der Variable) beobachten.

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

int unknown(char* acInValue1, char* acInValue2)
{
  int iPos = 0;
  while(acInValue2[iPos] != 0)
  {
    int li=0;
    while(acInValue1[li] != 0 && acInValue1[li] == acInValue2[iPos + li])
    {
      ++li;
    }
    if(acInValue1[li] == 0)
    {
      return iPos;
    }
    ++iPos;
  }
  return -1;
}

void main()
{
  char acString1[64];
  char acString2[1024];

  cout << "Zeichenkette1: ";
  cin >> acString1;

  cout << "Zeichenkette2: ";
  cin >> acString2;

  cout << "unknown: " << unknown(acString1, acString2) << endl;
}
```

zu 1. Es wird der Wert 1 zurückgegeben.
zu 2. Es wird der wert -1 zurückgegeben.
unknown liefert die erste Position von acString1 in acString2 oder -1 wenn acString1 nicht in acString2 enthalten ist.