Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang Sensorik/Sensorsystemtechnik

Note / normierte Punkte

Klausur in Programmieren

Winter 2009/2010, 18. Februar 2010

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:

MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	12	9	16	20	26	17	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) 1011 1110 1010 1111 (binär)	hex:	BEAF	(1P)
	oktal:	137257	(3P)
b) 254 (dezimal)	hex:	FE	(2P)
	binär:	1111 1110	(2P)
c) 6A (hex)	dezimal:	106	(3P)
	binär:	0110 1010	(1P)

2. Aufgabe: Grundlagen

Schreiben Sie ein Hauptprogramm, das die Kantenlänge eines Rechtecks von der Konsole (über cin) als reelle Zahl einliest und sowohl Umfang als auch Fläche als Ergebnis ausgibt (9P).

```
#include <iostream>

using namespace .std;

int main()
{
   double a, b;
   cout << “Kantenlaenge a: “;
   cin >> a;
   cout << “Kantenlaenge b: “;
   cin >> b;
   cout << “Umfang: “<< 2 * (a + b) << endl
        << “Flaeche: “ << a * b << endl;

   return 0;
}
```

3 P für die Syntax
3 P für die richtige Verwendung der C++ Konstrukte (Semantik)
3 P für die Korrektheit des Programms

3. Aufgabe: Schleifen

a) Kreuzen Sie an, welche Schleifen es in C/C++ gibt (6P):
(Falsche Antworten geben 1 Punkt Abzug; minimale Punktzahl dieser Aufgabe sind 0 P)

1. for X
2. repeat …. until ..... O
3. do … while X
4. if ..... O
5. foreach ..... O
6. when ..... O
7. else ..... O
8. while X
9. forget ..... O
10. loop ..... O

b) Schreiben Sie nachfolgend eine Schleife, in der die Quadrate aller Zahlen von 100 bis 199 summiert werden und das Ergebnis nach der Schleife in einer unsigned int-Variablen steht. Verwenden Sie für die Bestimmung der Quadrate ein Makro SQR und definieren Sie dieses Makro vor Beginn des Programmfragments (10P):

```
#define SQR(x) .((x)*(x))

unsigned int uiSumme = 0;
for(unsigned int uiIndex=100; uiIndex<200; uiIndex++)
{
   uiSumme += SQR(uiIndex);
}
```

3 P für die richtige Makrodefinition
2 P für die Initialisierung der Ergebnisvariablen
3 P für den korrekten Schleifenkopf
2 P für den korrekten Schleifenrumpf

4. Aufgabe: Array/Feld, Indizierung

Gegeben ist ein globales Feld mit den Monatstagen im Jahr:

unsigned int monatstage[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

a) Schreiben Sie eine Funktion ‚berechneTagImJahr', der Sie ein Datum mittels dreier Parameter ‚tag', ‚monat', ‚jahr' übergeben. Der Rückgabewert der Funktion soll die Anzahl der bis zu diesem Datum vergangenen Tage (inklusive des angegebenen Datums) bestimmen. Beachten Sie dabei, dass alle durch 4 teilbaren Jahre im Februar 29 Tage besitzen (Ausnahmen von dieser Regel sollen nicht berücksichtigt werden) (18P).

b) Skizzieren Sie eine Idee, wie man mit diesem Wert die Kalenderwoche bestimmen könnte (2P).

Beispiele:

berechneTagImJahr(20, 1, 2010) liefert den Wert 20.

berechneTagImJahr(10, 3, 2008) liefert den Wert 70.

a)

```
unsigned int berechneTagImJahr(unsigned int tag, unsigned int monat, unsigned int jahr)
{
    unsigned int tagImJahr = tag;
    for(unsigned int li=0; li<(monat-1); ++li)
    {
       tagImJahr += monatstage[ li ];
    }
    if( jahr % 4 == 0 && monat > 2)
    {
       tagImJahr++;
    }
    return tagImJahr;
}
```

4 P für den richtigen Funktionskopf
2 P für die Initialisierung der Ergebnisvariablen
4 P für den korrekten Schleifenkopf
4 P für den korrekten Schleifenrumpf
3 P für die richtige Berücksichtigung des Schaltjahres
1 P für den richtigen Rückgabewert

b)
Der Tag im Jahr dividiert durch 7 ergibt (ungefähr) die Kalenderwoche.

5. Aufgabe: Ganzes Programm (26 P)

Ergänzen Sie die Lücken im nachfolgenden Programmtext zu einem lauffähigen C++-Programm. Es hat die Aufgabe, zwei Zeichenketten, die mit dem Wert 0 enden, auf Gleichheit zu prüfen. Dabei soll die Groß- und Kleinschreibung keine Rolle spielen (Da jedem Buchstabensymbol ein Wert zugeordnet ist und die Buchstabensymbole sowohl der großen als auch der kleinen Buchstaben lückenlos und alphabetisch angeordnet sind, kann ein Kleinbuchstabe durch Subtraktion in einen Großbuchstaben geändert werden. Die Differenz ist für alle zugehörigen Buchstaben gleich und entspricht dabei der Differenz zweier zueinander gehörenden Buchstabensymbole, also z.B. ´a´-´A´).

```
#include <iostream>

using namespace std;

// Umwandeln der gegebenen Zeichenkette in Großbuchstaben
void inGrossbuchstaben(char* acZeichenkette)
{
   int iIndex = 0;
   char cAlphabetAbstand = ´a´ - ´A´;
   while(acZeichenkette[iIndex] != 0)
   {
      if(acZeichenkette[iIndex] >= ´a´ && acZeichenkette[iIndex] <= ´z´)
      {
         acZeichenkette[iIndex] -= cAlphabetAbstand;
      }
      iIndex++;
   }
}

// Prüfung der Zeichenketten auf Gleichheit
// Rückgabewert true: Zeichenketten sind gleich Rückgabewert false: Zeichenketten sind ungleich.
bool istGleich(char* acZeichenkette1, char* acZeichenkette2)
{
   int iIndex = 0;
   while(acZeichenkette1[iIndex] != 0 && acZeichenkette2[iIndex] != 0 &&
         acZeichenkette1[iIndex] == acZeichenkette2[iIndex])
   {
      iIndex++;
   }
   if(acZeichenkette1[iIndex] == 0 && acZeichenkette2[iIndex] == 0)
   {
      // Ende beider Zeichenkette erreicht!
      return true;
   }
   return false;
}
```

Kommentar [KM1]: 1P
Kommentar [KM2]: 1P
Kommentar [KM3]: 1P
Kommentar [KM4]: 1P
Kommentar [KM5]: 1P
Kommentar [KM6]: 1P
Kommentar [KM7]: 1P
Kommentar [KM8]: 1P
Kommentar [KM9]: 1P
Kommentar [KM10]: 1P
Kommentar [KM11]: 1P
Kommentar [KM12]: 2P
Kommentar [KM13]: 2P

```
int main()
{
   char   acZeichenkette1[200];
   char   acZeichenkette2[200];

   cout << "Zeichenkette1: ";
   cin >> acZeichenkette1;
   inGrossbuchstaben(acZeichenkette1);

   cout << "Zeichenkette2: ";
   cin >> acZeichenkette2;
   inGrossbuchstaben(acZeichenkette2);

   cout << "Zeichenketten sind ";
   if( ! istGleich(acZeichenkette1, acZeichenkette2))
   {
      cout << "un";
   }
   cout << "gleich";

   return 0;
}
```

Kommentar [KM14]: 1P
Kommentar [KM15]: 1P
Kommentar [KM16]: 1P
Kommentar [KM17]: 1P
Kommentar [KM18]: 1P
Kommentar [KM19]: 4P
Kommentar [KM20]: 1P
Kommentar [KM21]: 1P

6. Aufgabe: Algorithmus

a) Was berechnet die nachfolgende Funktion bzw. was enthalten die Variablen iValue1 und iValue2 des Ergebnisses? Bitte beschreiben Sie die Funktionsweise möglichst abstrakt. (15 P)

b) Wann erhalten Sie ein nicht definiertes Ergebnis? (2P)

Hinweis: Testen Sie den Algorithmus anhand eines kleinen Arrays (z.B. der Größe 5) und beobachten Sie die Variablenwerte.

```
struct tResult
{
  int iValue1;
  int iValue2;
};

tResult unknown(int* aiInArray, unsigned int uiInAnzahl)
{
    tResult  sResult;
    sResult.iValue1 = aiInArray[0];
    sResult.iValue2 = aiInArray[0];
    for(unsigned int uiIndex=0; uiIndex < uiInAnzahl; ++uiIndex)
    {
        if(sResult.iValue1 < aiInArray[uiIndex])
        {
            sResult.iValue1 = aiInArray[uiIndex];
        }
        sResult.iValue2 = sResult.iValue2 > aiInArray[uiIndex] ? aiInArray[uiIndex] : sResult.iValue2;
    }
    return sResult;
}
```

a)
iValue1 enthält das Maximum aller Elemente des Arrays.
iValue2 enthält das Minimum aller Elemente des Arrays.

b)
Wenn uiInAnzahl gleich 0 oder größer als die Elementanzahl des Arrays ist.