Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang Sensorik/Sensorsystemtechnik

Note / normierte Punkte

Klausur in Programmieren

Sommer 2010, 19. Juli 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	15	19	30	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

Kreuzen Sie an, was in der Sprache C/C++ ohne zusätzliche Definitionen zutrifft. Falsche Aussagen geben einen Punkt Abzug. (0 … 12P):

		korrekt	**/ nicht korrekt**
a)	int x := 4;	O	X
b)	char* z = “Hello“;	X	O
c)	constant e = 2.1;	O	X
d)	int i = 8 > 7 ? 6 % 3 : 1;	X	O
e)	long int li = (int) 9.123;	X	O
f)	real c = 2.1;	O	X
g)	short int si = 417365876 >> 4;	X	O
h)	char c = 044;	X	O
i)	double d = 3,4;	O	X
j)	int h = 0xABCdEF;	X	O
k)	const int x = (8 * 8) * ((2 + 5) – 6 * 3 + (2+2);	O	X
l)	int hex = 0#acbef;	O	X

2. Aufgabe: Grundlagen

Schreiben Sie ein Hauptprogramm, das die Kantenlänge a eines Quadrates und den Radius r eines Kreises beides als reellen Wert von der Konsole (über cin) einliest. Berechnen Sie anschließend beide Flächen (a^2, πr^2) zumindest näherungsweise (definieren Sie π als globale Konstante) und geben Sie anschließend nur die eine Meldung aus, welche der beiden Flächen größer ist (Gleichheit brauchen Sie nicht zu berücksichtigen). (9P)

```
#include <iostream>

using namespace std;

const double pi = 3.1415926536;

int main()
{
   double a;
   double r;
   cout << “Kantenlaenge a: “;
   cin >> a;
   cout << “Radius r: “;
   cin >> r;

   if(a * a > pi * r * r)
   {
      cout << “Quadratflaeche ist groesser als Kreisflaeche.” << endl;
   }
   else
   {
       cout << “Kreisflaeche ist grosser oder gleich zur Quadratflaeche.” << endl;
   }

   return 0;
}
```

3. Aufgabe: Funktionen

a) Erklären Sie den Unterschied zwischen „call by value“ und call by reference“ beim Aufruf einer Funktion, wenn ein Parameter vom Datentyp int verwendet werden soll. Machen Sie ein Beispiel, in dem Sie die Funktionsköpfe mit beiden Aufrufmethoden vervollständigen und einen Aufruf der jeweiligen Funktion beschreiben. Der Variablenname des Parameters im Funktionskopf soll „param“ sein: (6 P)

1. call by value: void demoCallByValue(int param);

Bei einem Aufruf demoCallByValue(2) oder demoCallByValue(x) wird der Wert 2 oder der Wert von x in die Variable param kopiert. Eine Änderung der Variablen param ändert x beispielsweise nicht.

2. call by reference: void demoCallByReference(int & param);

Beim Aufruf von demoCallByReference kann nur eine Variable z.B. x übergeben werden: demoCallByReference(x). Dabei ist innerhalb der Funktion demoCallByReference die Variable x in als Variablen param verfügbar. Änderungen von param ändern auch x.

b) Schreiben Sie eine Funktion mult, der zwei integer-Parameter start und ende übergeben werden. Berechnen Sie die Multiplikation aller Zahlen zwischen start und ende (inklusive start und ende). Geben Sie das Ergebnis als Rückgabewert von mult zurück und berücksichtigen Sie das im Funktionskopf. (9 P):

```
int mult(int start, int ende)
{
  int iErgebnis = start;
  for(int li=start+1; li<=ende; ++li)
  {
    iErgebnis *= li;
  }
  return iErgebnis;
}
```

4. Aufgabe: Array/Feld, Indizierung

a) Erklären Sie den Unterschied zwischen einem statischen und einem dynamischen Feld und was dabei generell zu beachten ist. Geben Sie jeweils ein einfaches Beispiel mit der Deklaration eines Feldes mit 10 int-Feldelementen: (10 P)

Statisches Feld: int feld[10];

Bei einem statischen Feld wird die Größe einmalig mit einer Konstanten festgelegt. Das Bereitstellen und Freigeben von Speicher wird automatisch im Hintergrund erledigt. Ein Feld ist wie jede andere Variable auch entweder global oder nur innerhalb eines Anweisungsblocks verfügbar.

Dynamisches Feld: int* feld = new int[10]; …. delete [ ] feld;

Der Speicher eines dynamischen Feldes muss mittels new angefordert und mittels delete [ ] wieder freigegeben werden. Die Anzahl der Feldelemente kann aber zur Laufzeit festgelegt werden und muss nicht konstant sein.

b) Gegeben ist folgender Funktionskopf: int maximum(int* aiInDaten, unsigned int uiInAnzahl) Schreiben Sie den dazu passenden Funktionsrumpf, der als Rückgabewert den größten Wert des Feldes zurück gibt. Der Parameter uiInAnzahl ist dabei immer größer 0 und enthält die Anzahl der Feldelemente. Sie können davon ausgehen, dass das Feld aiInDaten auch genau so viele Feldelemente enthält, wie uiInAnzahl angibt. (9 P)

```
int maximum(int* aiInDaten, unsigned int uiInAnzahl)
{
   int iMax =aiInDaten[ 0 ];
   for(unsigned int li=1; li < uiInAnzahl; ++li)
   {
      if(iMax < aiInDaten[ li ])
      {
         iMax = aiInDaten[ li ];
      }
   }
   return iMax;
}
```

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 – numberOf -, die die Anzahl eines Zeichens in einer Zeichenkette zählt. Die Zeichenkette ist durch eine abschließende 0 begrenzt. Falls alle Testfälle ohne Fehler durchlaufen werden, soll die Anzahl aller Kleinbuchstaben in einem dazu passenden Array gespeichert und anschließend ausgegeben werden. (30P)

```
#include <iostream>

using namespace std;

// Anzahl des Buchstaben cInChar in der Zeichenkette acInString zählen
// und als Funktionswert zurückgeben

int numberOf(char* acInString, char cInChar)
{
  int iCount = 0;

  for(int li=0; acInString[li] != 0; ++li)
  {
    if(acInString[li] == cInChar)
    {
      ++iCount;
    }
  }
  return iCount;
}

// Funktion zur Anwendung eines Testfalls um die Korrektheit von numberOf
// zu prüfen. Gibt numberOf den erwarteten Wert zurück, wird von
// testCaseNumberOf true zurück gegeben, false im Fehlerfalle

bool testCaseNumberOf(char* acInString, char cInChar,
                      int iInExpectedResult)
{
  int iResult = numberOf(acInString,  cInChar);

  cout << "Test: Anzahl " << cInChar << " in " << acInString << ": "

       << iResult << " ";

  bool bCheck = (iResult == iInExpectedResult);
  if(bCheck)
  {
    cout << "OK!" << endl;
  }
  else
  {
    cout << "NICHT OK! Erwartet: " << iInExpectedResult << endl;
  }
  return bCheck;
}
```

```
// Festlegung aller Testfälle für die Funktion numberOf
// Rückgabewert: Anzahl aller .fehlgeschlagenen Tests
int testNumberOf()
{
  int iErrorCount = 0;   // Anzahl aller aufgetretenen Fehlerfälle

  // Anzahl des Buchstaben 'l' mit testCaseNumberOf testen,
  // nur bei Rückgabe von false iErrorCount hochzählen

  if(! testCaseNumberOf("Hallole liebe Leut!", 'l', 4))
  {
    iErrorCount++;
  }

  // Anzahl des Buchstaben 'e' mit testCaseNumberOf testen,
  // nur bei Rückgabe von false iErrorCount hochzählen

  if(! testCaseNumberOf("Hallole liebe Leut!", 'e', 4))
  {
    iErrorCount++;
  }
  cout << endl;
  cout << "Aufgetretene Fehler gesamt: " << iErrorCount << endl;
  return iErrorCount;
}

// Zählen aller Kleinbuchstaben in der angegebenen Zeichenkette
void countCharacters()
{
  const int ciSize = 'z' - 'a' + 1;
  int aiCount[ciSize];
  for(char cFind='a'; cFind <= 'z'; ++cFind)
  {
    // Anzahl des Buchstaben cFind mit numberOf zählen

    aiCount[cFind - 'a'] = numberOf("Anzahl der Kleinbuchstaben.", cFind);
  }
  for(int li=0; li<ciSize; ++li)
  {
    cout << "'" << (char)('a'+li) << "': " << aiCount[li] << endl;
  }
}

int main()
{
  if(testNumberOf() == 0)
  {
    // alle Tests ok! rufe countCharacters auf

    countCharacters();
  }

  // Programm erst nach Betätigen der Enter-Taste beenden

  cin.get();

  return 0;
}
```

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 einer Zeichenkette, z.B. „Beatles“, „Abba“, „Scooter“ und beobachten Sie die Variablenwerte.

```
char unknown1(char cInChar)
{
  const int ciDiff = 'a' - 'A';
  if('a' <= cInChar && cInChar <= 'z')
  {
    return cInChar - ciDiff;
  }
  return cInChar;
}

bool unknown2(char* acInString)
{
  int iValue = 0;
  while(acInString[iValue] != 0)
  {
    ++iValue;
  }
  for(int li=0; li<iValue/2; ++li)
  {
    if(unknown1(acInString[li]) != unknown1(acInString[iValue - li - 1]))
    {
      return false;
    }
  }
  return true;
}
```

unknown1 ändert Kleinbuchstaben in Großbuchstaben. Andere Zeichen bleiben unverändert.

unknow2 liefert true, wenn es sich um ein Palindrom handelt, also eine Zeichenkette, die vorwärts und rückwärts dasselbe Wort ergibt. False, wenn es sich um kein Palindrom handelt.