Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang ESTB / EEEB

Note / normierte Punkte

Klausur in Programmieren

Winter 2015/2016, 17. Februar 2016

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:
MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	9	22	22	12	20	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

a) Geben Sie an, welcher der C/C++ Ausdrücke eine Variablendefinition (F) und welche eine Variablendeklaration (D) ist. "---" für keines von beiden, falls Sie der Meinung sind. (4 P):

1) #ifdef A ..... ___--- ___
2) double d = 3.41; ..... ___F ___
3) int b; ..... ___D ___
4) printf("Definition: %d\n", b); ..... ___--- ___

b) Welche Werte enthalten die nachfolgenden Variablen: (5 P)

1) int i = 12.6 * 2.0; — i == ___25____
2) double d = 2.6 * 3; — d == ___7.8____
3) int i = 10;
   double d = ((i++ * ((i +1) % 3)) – 1) / 2.0; — i == __11__, d == __9.5__

2. Aufgabe: Grundlagen

Schreiben Sie ein funktionsfähiges 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 die reziproke Summe ($y = 1 / x_1 + 1 / x_2 + \ldots + 1 / x_n$) aller bis zu diesem Zeitpunkt eingegebenen Werte und speichern Sie diese Summe in der Variablen y mit einem dafür geeigneten Datentyp. Danach geben Sie diesen reellen Wert von y auf die Konsole aus. Binden Sie alle dazu notwendigen Bibliotheken ein, aber nicht mehr. (22P)

```c
#include <stdio.h>

int main()
{
  float x = 0.0;
  float y = 0.0;

  do
  {
    printf("Wert: ");
    scanf("%f", &x);

    if(x != 0.0)
    {
      y += 1.0 / x;

      printf("Summe %f\n", y);
    }
  }
  while(x != 0.0);

  return 0;
}
```

3. Aufgabe: Funktionen

a) Erklären Sie aus welchen Teilen eine Funktion besteht (strukturiert vorgehen!) (16 P):

1. Funktionskopf
   a.) Datentyp des Rückgabewertes
      - void bedeutet kein Rückgabewert
   b.) Funktionsname
   c.) Parameterliste in runden Klammern eingeschlossen
      - Parameterliste besteht aus keinem, einem oder beliebig vielen Parametern
      - mehrere Parameter werden mit einem Komma getrennt
      - ein Parameter wird wie eine Variable deklariert und verwendet
2. Funktionsrumpf
   - entspricht einem Anweisungsblock
   - Anweisungsblöcke enthalten keine oder beliebig viele mit einem Semikolon getrennte Anweisungen und beginnen mit ‘{‘ und enden mit ‘}‘.
   - mit ‘return‘ wird ein Funktionsergebnis zurück gegeben

b) Wie bezeichnet man bei folgenden Beispielen die Art, mit der die Parameter der Funktion übergeben werden? Erklären Sie den Mechanismus in kurzen Stichworten (6 P):

i. `int function1(double* pdValue)`

Aufruf mittels call by reference
Es wird die Variable selbst als Referenz übergeben. Änderungen dieses Parameters innerhalb der Funktion bedeuten auch Änderungen der übergebenen Variablen.

ii. `void function2(int iValue)`

Aufruf mittels call by value.
Es wird eine Kopie des übergebenen Wertes im Parameter gespeichert. Änderungen des Parameters haben keinen Einfluss auf Variablen im Funktionsaufruf.

- Lösen Sie die Aufgaben bitte auf dem Blatt -

4. Aufgabe: Array/Feld, Indizierung

a) Erklären Sie den Unterschied zwischen einem Array und einer Struktur (kein Roman – kann Punktabzug geben!). (4 P)

Ein Array ist ein zusammengesetzter Datentyp, bei dem alle Elemente denselben Datentyp besitzen. Der Zugriff auf einzelne Felder erfolgt mit dem Index-Operator [ ].

Eine Struktur ist ein zusammengesetzter Datentyp, bei dem die Elemente unterschiedliche Datentypen besitzen können. Der Zugriff auf einzelne Elemente erfolgt mit dem Punkt-Operator .

b) Schreiben Sie eine kleine Funktion, bei der ein Array mit reellen Messwerten übergeben wird (beachten Sie dabei die Handhabe der Array-Länge!). Deklarieren Sie in der Parameterliste zusätzlich einen weiteren reellen Parameter fSchwelle. Bestimmen Sie nun im Funktionsrumpf, wie viele Elemente des Arrays größer sind als der Wert in fSchwelle und geben diese Anzahl als Funktionswert zurück. Geben Sie anschließend ein Programmfragment an, bei dem die von Ihnen definierte Funktion aufgerufen wird (d.h. weniger als 5 Anweisungen reichen und bitte kein Hauptprogramm oder irgendwelche Ein- oder Ausgaben schreiben!). Bibliotheken dürfen nicht verwendet werden! (8 P)

```
int schwelle(float* afInValues, int iInCount, float fSchwelle)
{
  int iAnzahl = 0;
  int li;

  for(li=0; li < iInCount; ++li)
  {
    if(afInValues[li] > fSchwelle)
    {
      iAnzahl++;
    }
  }
  return iAnzahl;
}

  float afValues[5] = {3.0, 4.5, 2.1, 9.8, 6.5};
  float fMaximum;

  iAnzahl = schwelle(afValues, 5, 3.1);
```

5. Aufgabe: Zeichenketten

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

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

b) Schreiben Sie eine Funktion lower, die eine Zeichenkette acString mit Klein-/ und oder Großbuchstaben in Kleinbuchstaben ändert. Alle anderen Zeichen sollen unverändert bleiben. (kein Hauptprogramm, keine Ein- oder Ausgabe!). (14 P)

lower("This IS An ExamPle!") → "this is an example!"
Hinweis: 'A' … 'Z' und 'a' … 'z' sind lückenlose, zusammenhängende Bereiche beim ASCII-Code; nutzen Sie deshalb: char cOffset = 'a' – 'A';

```
void lower(char* acInOutString)
{
  int iIndex = 0;
  char cOffset = 'a' - 'A';
  while(acInOutString[iIndex] != 0)
  {
    char cCurrent = acInOutString[iIndex];
    if('A' <= cCurrent  &&  cCurrent <= 'Z')
    {
      acInOutString[iIndex] += cOffset;
    }
    iIndex++;
  }
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

6. Aufgabe: Algorithmus

Was macht die nachfolgende Funktion unknown? Welche Ergebnisse sind möglich und was bedeuten sie?

Bitte beschreiben Sie die Funktionsweise möglichst abstrakt – Romane geben Abzug! (15 P)

Hinweis: Testen Sie den Algorithmus anhand zweier Zeichenketten und beobachten Sie die Variablenwerte.

```
int unknown(char* acInChars1, char* acInChars2)
{
  int iIndex = 0;
  while(acInChars1[iIndex] == acInChars2[iIndex] && acInChars1[iIndex] != 0)
  {
    iIndex++;
  }
  if(acInChars1[iIndex] == acInChars2[iIndex])
  {
    return 0;
  }
  else
  if(acInChars1[iIndex] < acInChars2[iIndex])
  {
    return -1;
  }
  return 1;
}
```

```
Vergleich zweier 0-terminierten Zeichenketten.

Als Ergebnis ist möglich:
-1 : Zeichenkette1  < Zeichenkette2
 0 : Zeichenkette1 == Zeichenkette2
 1 : Zeichenkette1  > Zeichenkette2
```