Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang ESTB / EEEB

Note / normierte Punkte

Klausur in Programmieren

Sommer 2018, 19. Juli 2018

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:
MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	12	15	15	19	24	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 ohne zusätzliche Definitionen zutrifft. (12P):

		zutreffend	**nicht zutreffend**
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 vollständiges und lauffähiges C-Hauptprogramm, das die Höhe h und die Grundseite g eines Dreiecks als reelle Zahlen einliest und dessen Fläche (g * h)/2.0 berechnet. Die Fläche soll kaufmännisch gerundet werden und als ganze Zahl in einer Variablen vorliegen. Negative Werte brauchen nicht berücksichtigt bzw. abgefangen werden. Anschließend soll die gerundete Fläche auf die Konsole ausgegeben werden. Berücksichtigen Sie bei der Ausgabe auf die Konsole eine passende Beschriftung. Es reicht eine main-Routine, weitere Funktionen sind (hier) nicht gefordert. Keine globalen Variablen verwenden! (15 P)

```c
#include <stdio.h>

int main()
{
  float fHoehe;
  float fGrundseite;

  printf("Hoehe: ");
  scanf_s("%f", &fHoehe);
  printf("Grundseite: ");
  scanf_s("%f", &fGrundseite);

  int iFlaeche = (int)((fHoehe * fGrundseite)/2.0 + 0.5);

  printf("Gerundete Flaeche: %d\n", iFlaeche);

  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 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 die Adresse auf eine Variable z.B. x übergeben werden: demoCallByReference(&x). Dabei ist innerhalb der Funktion demoCallByReference die Variable x in als Variable *param verfügbar. Änderungen von *param, z.B. *param = 5, ändern auch x, hier im Beispiel zu 5.

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, verwenden Sie eine passende Schleife!). Geben Sie das Ergebnis als Rückgabewert von mult zurück und berücksichtigen Sie das im Funktionskopf. (Bitte kein Hauptprogramm oder irgendwelche Ein- oder Ausgaben schreiben!) (9 P):

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

- Lösen Sie die Aufgaben bitte auf dem Blatt -

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 = (int*)malloc(10 * sizeof(int)); …. free(feld);

Der Speicher eines dynamischen Feldes muss mittels malloc angefordert und mittels free 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. (Bitte kein Hauptprogramm oder irgendwelche Ein- oder Ausgaben schreiben!) (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;
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

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!). (10 P)

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

b) Schreiben Sie eine Funktion count, die den Parameter cInSample mit dem Datentyp char sowie den Parameter acInString mit dem Datentyp char* besitzt. Die Funktion count zählt die Anzahl von Zeichen in der Zeichenkette acInString die gleich dem Zeichen definiert durch cInSample ist. (kein Hauptprogramm, keine Ein- oder Ausgabe!). (14 P)

count('a', "This is a little and also pretty example!") → 4

```
int count(char cInSample, char* acInString)
{
  int iResult = 0;
  int iIndex = 0;
  while (acInString[iIndex] != 0)
  {
    if (acInString[iIndex] == cInSample)
    {
      ++iResult;
    }
    ++iIndex;
  }
  return iResult;
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

6. Aufgabe: Algorithmus

Was machen die nachfolgenden Funktionen unbekannt1-3? Was wird ausgegeben?

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

Hinweis: Testen Sie den Algorithmus anhand eines Funktionsaufrufs und beobachten Sie die Variablenwerte.

```
#include <stdio.h>

int unbekannt1(int iInValue1, int iInValue2)
{
  int aMonatstage [] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
  if (iInValue1 != 2)
  {
    return aMonatstage[iInValue1 - 1];
  }
  return (iInValue2 % 4 == 0) && (iInValue2 % 100 != 0 || iInValue2 % 400 == 0) ? 29 : 28;
}

int unbekannt2(int iInValue1, int iInValue2, int iInValue3)
{
  int iTage = iInValue1;
  for (int iMonat = 1; iMonat < iInValue2; ++iMonat)
  {
    iTage += unbekannt1(iMonat, iInValue3);
  }
  --iInValue3;
  iTage += iInValue3 * 365 + iInValue3 / 4 - iInValue3 / 100 + iInValue3 / 400;

  return iTage;
}

char* unbekannt3(int iInValue1, int iInValue2, int iInValue3)
{
  char* acBezeichner [] = {"Montag", "Dienstag", "Mittwoch", "Donnerstag",
                           "Freitag", "Samstag", "Sonntag"};
  int iWert = (unbekannt2(iInValue1, iInValue2, iInValue3) + 6) % 7;
  return acBezeichner[iWert];
}

int main()
{
  printf("unbekannt3: %s\n", unbekannt3(19, 7, 2018));
  return 0;
}
```

unbekannt1: Bestimmt die Anzahl der Tage eines Monats im angegebenen Jahr.
unbekannt2: Anzahl Tage vom angegebenen Datum zum 01.01.0000.
unbekannt3: Gibt den Wochentag des angegebenen Datums zurück.

Ausgabe: „unbekannt3: Donnerstag“

- Lösen Sie die Aufgaben bitte auf dem Blatt -