Fachhochschule Gießen-Friedberg **14. Juli 2009**

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

Klausur
Konzepte Systemnaher Programmierung
SS 2009

Personalien:

Name, Vorname: .....

Matrikelnummer: .....

Hinweise:

- **Die Bearbeitungszeit beträgt 90 Minuten.**
- **Alle schriftlichen Hilfsmittel sind zugelassen; andere Hilfsmittel, insb. elektronische Rechen- und Kommunikationsapparate dürfen nicht verwendet werden.**
- **Die Aufgaben sollen nur auf diesen Aufgabenblättern bearbeitet werden. Bei Bedarf kann zusätzliches Papier zur Verfügung gestellt werden.**
- **Zur sicheren Zuordnung aller Lösungen wird um eine persönliche Kennung (Name u./o. Matrikelnr.) auf allen Blättern gebeten.**
- **Auf Wunsch darf auch Bleistift verwendet werden.**

Zur leichteren Lesbarkeit werden Substantive nur in einem Geschlecht („Nutzerin“) verwendet.

1. Aufgabe (35 Punkte)

a) Was ist eine Software-Umgebung?

b) In Zusammenhang mit Software-Plattformen und -Umgebungen ist meist auch die Rede von „hoch“ bzw. „Höhe“. Was ist dabei hoch? Die Komplexität, die Abstraktion, der Entwicklungsaufwand oder der Preis? (Nennung genügt)

c) Was versteht man unter einem „Software-Modul“?

d) Was sind Daten?

e) Sind Informationen Daten? (Ja/Nein genügt)

f) Kurz vor den Wahlen besuchen Sie eine öffentliche Diskussion, die in der Pinakothek einer Großstadt stattfindet. Der Weg zum Veranstaltungsraum führt durch einen Gang, an dessen Wänden wertvolle Gemälde hängen. An jedem Bild steckt ein Schild mit dem Namen des Künstlers, der es gemalt hat, und daneben ein weiteres Schild mit der Aufschrift: „Unverkäuflich“. Am Ende des Gangs befindet sich der Saal mit einem langen Tisch; darauf stehen Schilder, die wie jene an den Bildern aussehen, mit den Namen der Politiker, die an diesen Plätzen erwartet werden. Neben diesen Namensschildern sind keine weiteren Schilder mit der Aufschrift „Unverkäuflich“. Beantworten Sie dazu bitte folgende Fragen:

Was ist die größte Gemeinsamkeit, welche die Namensschilder an den Gemälden und auf dem Tisch haben: Angaben, Daten oder Informationen? (Nennung genügt)

g) Warum dürfen bei Politikern die Schilder fehlen, die bei Gemälden wichtig erscheinen?

h) Ist Provokation eine Art von Interaktion? Begründen Sie Ihre Antwort!

i) Welcher Begriff ist allgemeiner: Blockierung, oder Blockade? (Nennung genügt)

j) Hängt die Entstehung von Blockierungen typischerweise mit synchronen oder mit asynchronen Ereignissen zusammen? Was aus der Definition dieser Begriffe führt Sie zu Ihrer Aussage?

k) Sie fahren mit dem Zug und bemerken nach einiger Zeit, daß eine Person im selben Waggon ständig auf das Leuchtschild „WC“ schaut, weil sie offenbar die Toilette benutzen will. Aber jedes Mal, wenn die Toilette frei wird und diese Person aufsteht, um dorthin zu gehen, stehen andere Menschen auf, die näher an diesem Ort sitzen, so daß der notleidende Fahrgast über die ganze Fahrt keine Möglichkeit erhält, das WC aufzusuchen.

Wie heißt die Blockierungsart, deren Zeuge Sie werden (deutsche, englische Bezeichnung), und woran erkennen Sie sie? ⇒

l) Worin unterscheidet die o.a. Blockierungsart von den anderen? Nennen Sie mindestens ein Kriterium, das jeweils nicht erfüllt ist!

2. Aufgabe (55 Punkte)

Sie übernehmen von einem Freund ein unvollständiges Programm, das sich in die Reihe bereits gestarteter Kopien von sich selbst einordnet, nach einer vorgegebener Laufzeit selbständig terminiert und jeweils die Wartezeit zwischen dem aktuellen und dem vorausgegangenen Start festhält. (Listing des bisher entwickelten Quellcodes am Ende dieser Aufgabe.)

Die Fertigstellung von `MemoCheck.c` soll anhand der Beantwortung folgender Fragen erfolgen:

a) Die Funktion `memChk()` hat eine koordinierende Rolle im gesamten Programm. Von dort meldet auch der Compiler gleich zwei Fehler: Der Bezeichner `p2f` erscheint in der Zuweisung `p2f=msgESC`, obwohl er nicht als Variable deklariert wurde und im Aufruf `p2f()`, obwohl es keine Funktion mit diesem Namen gibt.

Wie lautet eine Deklaration für `p2f`, die beide Fehler behebt?

b) Ihre Kollegen machen Sie darauf aufmerksam, daß Ihr Code evtl. auch mit alten Compilern verwendet werden soll, und daß der o.a. `p2f()`-Aufruf am Ende von `memChk()` angepaßt werden sollte. Welche Form ist dafür empfehlenswert?

Das Programm kann jetzt compiliert und gestartet werden, so daß Sie seine innere Struktur zeilenweise untersuchen können:

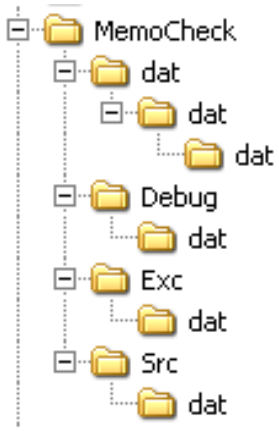
c) Als erstes wird mit einem Aufruf von `initStruc()` ein Struktur-Zeiger (`shared`) initialisiert und eine Datei eröffnet.
   i) In welcher Form sollen Daten in diese / aus dieser Datei übertragen werden?
   ii) Woran erkennen Sie dies?
   iii) Was wäre die Alternative?

d) Wird die Datei eröffnet, falls sie nicht existiert?
   Wenn ja: Welche der beiden `fopen()`-Anweisungen sorgt dafür?
   Wenn nein: Welche Absicht steckt dahinter?

e) Der Ausdruck:
   `(_shared.memo=fopen("../dat/Memo.txt","r+b"))==NULL`
   steht für (bitte korrekte Antwort/en ankreuzen):

	mißlungenen Versuch, eine vorhandene Datei zu öffnen
	erfolgreichen Versuch, eine vorhandene Datei zum Lesen zu öffnen
	erfolgreiche Neueinrichtung einer Datei
	mißlungenen Versuch, eine neue Datei im angegebenen Pfad einzurichten
	mißlungenen Versuch, eine vorhandene Datei nur zum Lesen zu öffnen

f) Angenommen, Sie starten das Programm im Verzeichnis Exc einer Verzeichnisstruktur nach der nebenstehenden Abbildung. Markieren Sie bitte das Verzeichnis, auf das der o.a. Pfad zeigt, worin die Datei Memo.txt liegt.



g) Nach dem Öffnen der Datei wird in `initStruc()` die Funktion `time()` aufgerufen.
   i) Welche Zeitdifferenz zu welchem Zeitpunkt wird mit ihr gemessen?
   ii) In welcher physikalischen Einheit und in welcher Codierung wird das Ergebnis ausgedrückt?

h) Ein erfahrener Kollege sagt Ihnen, die Speicherung der abgefragten Zeit habe die Wahl des Datei-Modus beeinflußt. Was ist darunter zu verstehen?

i) Mit der anschließenden Zuweisung erhält die Variable `dummyi`

	die Anzahl korrekt gelesener Bytes aus der Datei
	die Anzahl korrekt aus der Datei gelesener Datenobjekte
	die Anzahl korrekt über die Tastatur eingegebener Datenobjekte
	die Zahl 1, wenn das Lesen geglückt ist, andernfalls die Zahl 0
	den als ganze Zahl codierten Inhalt des ersten gelesenen Elementes (hier: Adresse des Datei-Zeigers `memo`)

j) Wie lange nach einem Programm-Start beginnt `initStruc()`, neu zu zählen und meldet erneut den 1. Start? (Zeitangabe genügt)

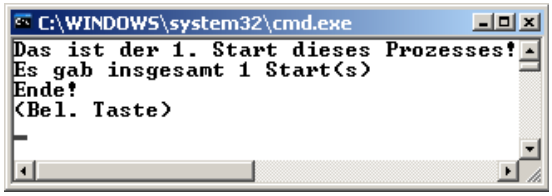
k) Die in `initStruc()` verwendete Strukturvariable `_shared` ist vom Typ `static`. Was bedeutet das für den Speicher und die Werte dieser Variable?

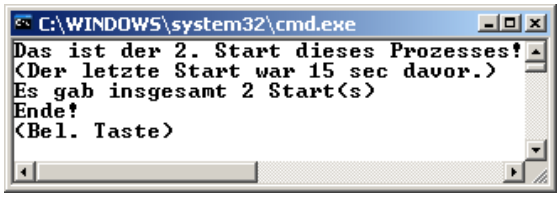
l) Die Ausführung von `memChk()` kommt nicht weit: Bereits nach dem Aufruf von `initStruc()` meldet das System eine Zugriffsverletzung, die zum Absturz führt. Ihre Recherche ergibt, daß der Zeiger `shared` auf nichts Sinnvolles zeigt; da fällt Ihnen auf, daß Sie die Compiler-Warnung mißachtet hatten, wonach `initStruc()` keine Typspezifizierung hat.

Wie lauten die korrekte Deklaration der Funktion `initStruc()` und ihre abschließende `return`-Anweisung?

m) Nach der letzten Änderung kann nun ein erster Programmlauf bis zum Ende beobachtet werden (Abb. li.): Im Debugger kann überprüft werden, daß die Strukturvariable `shared` die erwarteten Werte bekommt, mit denen `memChk()` die Datei aktualisieren soll. Doch weitere Programm-Starts (während des Laufs des ersten) bieten exakt dasselbe Bild und melden ebenfalls den „1. Start“.

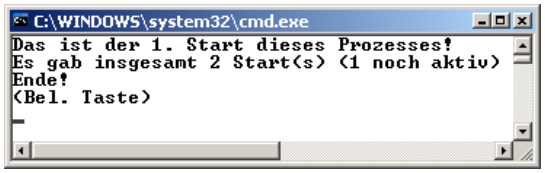
Nach weiteren Versuchen stellen Sie fest, daß die Daten durchaus in die Datei übertragen werden – allerdings nach einer gewissen Zeit. Nur so können Sie auch einen „2. Start“ erreichen (Abb. re.).

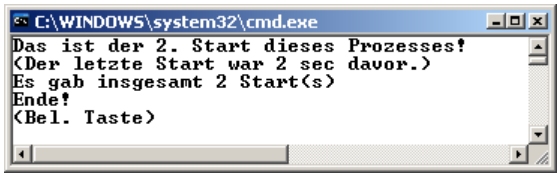




i) Woran liegt diese Erscheinung und wie läßt sie sich beheben?
ii) Innerhalb welcher Zeitspanne meldet das Programm einen „2. Start“, und welche Ereignisse markieren Beginn und Ende dieser Zeitspanne?

n) Nun vermittelt `MemoCheck` die erwarteten Eindrücke (s. Abbn), und Sie arbeiten an der Verbesserung seiner Handhabung: Die Funktion `updateMsg()`, die in `memChk()` die jeweils erreichte Anzahl von Programm-Starts ermittelt und ausgibt, ist so aufgebaut, daß ihre Meldung sich selbst überschreibt. Woran ist das zu erkennen?





o) Sie beschließen, den `updateMsg()`-Aufruf aus `memChk()` in den Timer (Funktion `waitnChk()`) zu verschieben. Geben Sie ein Beispiel für Stelle(n) in `waitnChk()`, die dafür in Betracht kommen!

Und nun widmen Sie sich dem allgemeinen Verständnis des funktionierenden Programms und beantworten folgende Fragen:

p) In den letzten Zeilen von `memChk()` war bereits die (inzwischen von Ihnen angepaßte) bedingte Anweisung: `if (p2f) p2f();`

Ist die davor gesetzte `if`-Abfrage sinnvoll und notwendig, sagt sie etwas über die Aktualisierung der Variablen `p2f` aus, oder kann sie ebenso gut ausgelassen werden, ohne daß sich das Programm-Verhalten ändert?

q) Ihnen fällt auf, daß bei Beendigung mit der Escape-Taste der Aufruf `fclose()` in `memChk()` nicht erreicht wird. Ist in diesem Fall die Speicherung der Daten in der Datei nicht gefährdet? (Kurze Erklärung)

r) Sie wollen sich die benutzte Datei Memo.txt im Editor anschauen. Welches der drei folgenden Bilder erwarten Sie? (Kurze Begründung)

Abb. 1

Abb.2

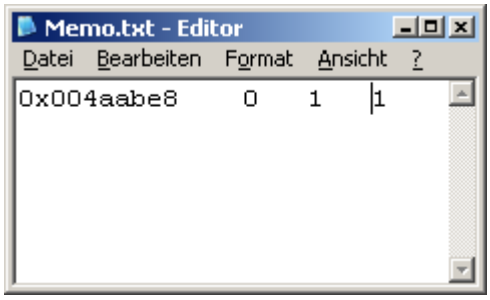


Abb.3

```
/*                                MemoCheck.c                                 */
#include <conio.h>  // wg. _getch(), _kbhit()
#include <stdio.h>  // wg. fopen() etc.
#include <stdlib.h> // wg. exit()
#include <time.h>   // wg. time(), clock() etc.

#define CR       13
#define ESC      27
#define WARTE    10

typedef struct
{ FILE   *memo;           //Datei-Zeiger
  time_t  lastTime;       //Zeitpunkt letzter Start
  int     nStart, nActive;//Gesamtzahl aller u. der noch aktiven Starts
} filStat;

/* ********************************************************************** */
   void msgESC (void)
/* ********************************************************************** */
/*Massnahmen bei Druck von Esc:*/
{ printf ("\n\rAbbruch durch BenutzerIn! \n\r(Bel. Taste)\n");
 _getch(); exit(0);
}
/* ********************************************************************** */
   void updateMsg (filStat *shared)
/* ********************************************************************** */
/*Daten-Aktualisierung aus Datei:*/
{ rewind(shared->memo);
  fread(shared, sizeof(filStat), 1, shared->memo);
  printf ("\rEs gab insgesamt %d Start(s) ", shared->nStart);
  return;
}
/* ********************************************************************** */
   int waitnChk (filStat *shared)
/* ********************************************************************** */
/*Daten aktualisieren und Warten :*/
{ int    ch=' ';
  time_t tNow, tEnd;
  time(&tNow);  tEnd = tNow + WARTE;
  while (tNow < tEnd)
  { if (_kbhit())
    { ch = _getch();
      if (ch == ESC) return(ch);
    }

    time(&tNow);
  }
  return (0);
}
/* ********************************************************************** */
            initStruc (void)
/* ********************************************************************** */
{ static filStat _shared;
  time_t myTime;
  int    dummyi;

  /*Datei oeffnen:*/
  if (!_shared.memo)
  { if ((_shared.memo=fopen("../dat/Memo.txt","r+b"))==NULL &&
        (_shared.memo=fopen("../dat/Memo.txt","w+b"))==NULL) exit(0);
  time(&myTime);
```

```
  /*Datei auslesen, auswerten:*/
  dummyi = fread(&_shared, sizeof(filStat), 1, _shared.memo);
  if (dummyi < 1 || _shared.nStart < 1 ||
              (int) ((double)myTime-(double)_shared.lastTime) > 2*WARTE)
  { _shared.nStart = 1;  _shared.nActive = 1;
  } else
  { _shared.nStart++;  _shared.nActive++;
  }

  /*Meldung:*/
  printf ("Das ist der %d. Start dieses Prozesses!\n\r", _shared.nStart);
  if (_shared.nStart>1) printf("(Der letzte Start war %ld sec davor.)\n\r",
                       (int) ((double)myTime-(double)_shared.lastTime));
  _shared.lastTime = myTime;
  }

}

/* ********************************************************************** */
   void memChk (void)
/* ********************************************************************** */
{ filStat *shared;


  /*Initialisierungen:*/
  shared = initStruc();

  /*Datei-Aktualisierung:*/
  rewind(shared->memo);
  fwrite(shared, sizeof(filStat), 1, shared->memo);


  /*Warteschleife mit Daten-Aktualisierung aus Datei:*/
  if (waitnChk(shared) == ESC) p2f = msgESC;

  /*Abmeldung:*/
  updateMsg (shared);

  shared->nActive--;
  if (shared->nActive > 0) printf ("(%d noch aktiv)\r", shared->nActive);
  rewind(shared->memo);
  fwrite(shared, sizeof(filStat), 1, shared->memo);

  /*Evtl. Meldung zu Abbruch (mit p2f):*/

  if (p2f) p2f();
  fclose (shared->memo);
  return;
}

/* ********************************************************************** */
   int main (void)
/* ********************************************************************** */
{ memChk ();
  printf ("\n\rEnde! \n\r(Bel. Taste)\n");  _getch();
  return (0);
}
```

3. Aufgabe (10 Punkte)

Das folgende kurze Programm arbeitet teils mit Funktionen, die einen Wert erwarten, und teils mit solchen, die eine Adresse erwarten.

Ergänzen Sie bitte die unten vorbereitete Ausgabe!

```
/*                              ValRef.c                                */
#include <conio.h>  //wg. _getch()
#include <stdio.h>  //wg. printf()

/*Aufruf mit Wert- u. Adressenuebergabe:*/
int incByVal (int x)
{ x = x*x;
  printf (" byVal: x=%4d\n\r", x);
  return x*x;
}

int incByRef (int *x)
{ *x=(*x)*(*x);
  printf (" byRef: x=%4d\n\r", *x);
  return *x;
}

int main()
{ int x=2;
  x = incByRef(&x);  printf (" x(byRef)=%4d\n\r", x);
  x = incByVal( x);  printf (" x(byVal)=%4d\n\r", x);
 _getch();
  return (0);
}
```

Ausgabe:

```
byRef: x=
x(byRef)=
byVal: x=
x(byVal)=
```

Platz für Notizen: