Fachhochschule Gießen-Friedberg

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

25. März 2009

Klausur
Konzepte Systemnaher Programmierung
WS 2008 / 09

– Lösungshilfe –

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 (15 Punkte)

a) Was ist eine Software-Plattform?

Die ‚tiefste' (Hardware-nächste) genutzte Software-Schicht – i.d.R. das Betriebssystem.

b) Sie planen Multimedia-Software für Haushaltsgeräte: Waschmaschinen, Backöfen u.ä., verbunden mit haushaltsüblichen Eingabe- und Ausgabe-Geräten (Kopfhörer, Fernseher, PC-Maus, Telefon u./o. Navigationssystem), sollen in Zukunft mit Bild und/oder Ton die eigene Bedienung erklären, die Behebung von Fehlern ermöglichen etc.. Sie entwickeln Ideen über Bedienungs-Hörbücher, Reparatur-Navigationskarten u.ä..

Aufgrund Ihrer exzellenten KSP-Kenntnisse bekommen Sie den Auftrag, C-Funktionen zu implementieren, über die sich die angeschlossenen Apparate beim Gesamtsystem „anmelden", damit sie von dort zu gegebenen Anlässen korrekt „angesprochen" werden können. Von diesen Funktionen steht bereits fest, daß ihre Parameterlisten jeweils genau eine Variable enthalten sollen.

Was wird die Bitkombination dieser einen Variablen jeweils codiert enthalten? (Zahl, Zeichen, Adresse, Struktur o.ä., mit welchem Inhalt?)

Die Variable wird die (Einsprungs-)Adresse der Callback-Funktion enthalten.

c) Was sind Daten?

Angaben, die etwas kennzeichnen.

d) Was ist Information?

Daten, die in Entscheidungen einfließen.

e) Wird bei der Vermittlung religiösen Glaubens (Katechismus) Information im Sinne der gegebenen Definition vermittelt? (Kurze Begründung!)

Ja. Der Glaube fließt in Entscheidungen ein.

f) Wie Anfang 2009 aus der Universität Dortmund gemeldet wurde, haben 7% der Biologie-Studierenden Schwierigkeiten, ihr Fach zu lernen und Klausuren zu bestehen, weil sie die Theorie einer Evolution über Milliarden von Jahren nicht vereinbaren können mit ihrem Glauben an die biblische Schöpfungsgeschichte (Kreationismus), wonach die Welt in nur sechs Tagen erschaffen wurde. Beantworten Sie bitte hierzu folgende Fragen:

i) Ist die Evolutionstheorie Information im Sinne der gegebenen Definition? (Ja / Nein genügt.)

Ja.

ii) Ist die Schöpfungsgeschichte Information im Sinne der gegebenen Definition? (Ja / Nein genügt.)

Ja.

iii) Falls Sie auf (i) und (ii) gleich geantwortet haben: Sehen Sie da einen Widerspruch?

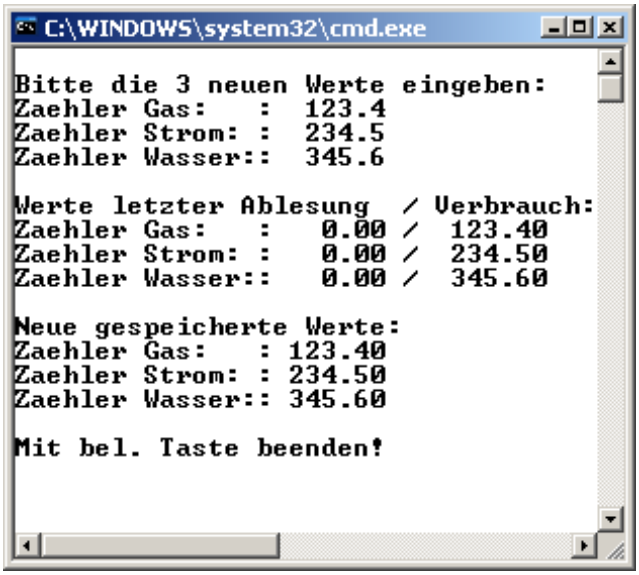
Falls Sie auf (i) und (ii) unterschiedlich geantwortet haben: Worin sehen Sie den Unterschied?

Kein Widerspruch: Beide Theorien fließen in Entscheidungen ein.

2. **Aufgabe** (40 Punkte)

Nach erfolgreicher KSP-Klausur treffen Sie auf der bundesweiten Demo Ihren ehemaligen Lieblingslehrer, der Sie um Hilfe bei einem kleinen C-Projekt bittet.

Ziel des Projektes ist, die im Zuge des Personalabbaus eingeführte Selbst-Ablesung der Zähler für Gas, Strom und Wasser in einer Datei zu speichern und bei der nächsten Ablesung auch den zwischenzeitlichen Verbrauch automatisch anzuzeigen.



Das Programm `GasMan.c` (s. Listing am Ende dieser Aufgabe) zeigt zunächst das gewünschte Verhalten (s. Bild), beim Dauertest aber auch Schwächen, die Sie anhand der nachfolgenden Fragen beheben wollen.

a) Das Programm öffnet die Datei `GM-Kosten.txt` anhand der Auswertung einer Bedingung:

```
if((memo=fopen("./GM-Kosten.txt","r+")) ||
   (memo=fopen("./GM-Kosten.txt","w+"))); else return(-1);
```

Einer der beiden hier verwendeten Modi `"r+"` und `"w+"` garantiert das Öffnen einer Datei, auch wenn der andere versagt. Welcher der beiden ist es, und warum wird nicht immer nur dieser verwendet?

Der Modus "w+" (bzw. "w") öffnet immer eine neue Datei. Existiert eine mit den angegebenen Namen (und Pfad), wird die alte überschrieben (gelöscht). Wenn die exisiterende Datei nicht gelöscht werden soll, verwendet man den Modus "r+" (bzw. "r").

b) Woran erkennen Sie, daß es sich hier um das Öffnen einer formatierten Datei handelt? Was wäre die Alternative?

Die Alternative wäre eine binäre Datei; dazu müßte im Modus ein „b“ stehen ("r+b" bzw. "w+b").

c) Wie wirkt sich die ODER-Verknüpfung der beiden `fopen`-Anweisungen im Programm aus? Kann es passieren, daß nach einem Start die Datei zweimal nacheinander geöffnet wird?

Wenn ja: Wann passiert dies, und welcher Modus gilt dann für die geöffnete Datei?

Wenn nein: Wodurch wird dies verhindert? Werden überhaupt zwei `fopen`-Aufrufe benötigt?

Nein: Aufgrund der „Kurzschluß-Auswertung“ von `if`-Abfragen in C kann es nicht zu einem zweimaligen Öffnen kommen: Wenn eine Datei vorgefunden wird, wird sie zum Lesen und Schreiben geöffnet, und der ODER-Ausdruck kann nicht mehr zu Null werden; sonst wird eine neue Datei eingerichtet. Für diese Fallunterscheidung werden die beiden Aufrufe benötigt.

d) Was ändert sich an der Funktionsweise des Programms, wenn die Reihenfolge der beiden `fopen`-Anweisungen in der `if`-Abfrage vertauscht wird (erst `"w+"`, dann `"r+"`)?

Dann wird bei jeder Programm-Ausführung eine neue Datei eingerichtet: Die alten Daten werden schon beim Öffnen gelöscht. Die `fopen`-Anweisung mit `"r+"` wird nie ausgeführt.

e) Wenn keine der `fopen`-Anweisungen gelingt, soll das Programm beendet werden. Was soll aber nach dem vorliegenden Code passieren, wenn eine Datei geöffnet wird?

Nichts (leere Anweisung). Dann wird die nächste Anweisung ausgeführt (hier: `printf("\n\r ... Werte eingeben ... ALL);`).

f) Ihr Gesprächspartner berichtet, daß `GM-Kosten.txt` in unterschiedlichen Verzeichnissen eingerichtet wird, je nach dem, ob das ausführbare Programm direkt oder aus der Programmierumgebung gestartet wird, obwohl immer dieselbe `fopen`-Anweisung zum Einsatz kommt. Wie erklärt sich diese Beobachtung? Kann man Programme auch so schreiben, daß eine Datei immer im selben Verzeichnis geöffnet wird?

Der angegebene (relative) Pfad zeigt auf das (jeweils aktuelle) Verzeichnis. Das kann unterschiedlich sein, je nach dem, ob das Programm von der Umgebung oder von seinem Speicherort aus gestartet wird.

Alternativ dazu kann man den absoluten Pfad angeben (z.B.: `fopen("C:/GM-Kosten.txt","r+"));` das würde voraussetzen, daß dieser Pfad existiert.

g) Die drei neuen Zähler-Stände werden eingelesen mit der Anweisung:

```
scanf ("%f", stand+nr);
```

Handelt es sich beim Ausdruck `stand+nr` um eine Speicheradresse (hier: eines Zahlenwertes), oder um einen gespeicherten (Zahlen-)Wert? (Nennung genügt.)

`stand+nr` ist die Speicheradresse des eingegebenen Zählerstands.

h) Zum Auslesen des bisherigen Datei-Inhalts wird die Funktion `fscanf()` verwendet. Sie prüft zunächst, ob der Wert `EOF` eingelesen wurde.
   i) Wofür steht die Abkürzung „`EOF`“?
   ii) (1 Bonuspunkt) Welcher Zahlenwert ist dies?
   iii) Warum ist es sinnvoll, beim Antreffen dieses Wertes die bereitgestellte Variable (`dummyf`) explizit auf Null zu setzen?

i) „EOF“ steht für „End Of File“ (Datei-Ende).

ii) „EOF“ wird durch den Wert (-1) dargestellt.

iii) Erfolgt keine explizite Wertzuweisung, werden evtl. nicht vorhersehbare Werte zugewiesen; aber eine leere Datei steht stellvertretend für keinen Inhalt (0).

i) Wo legt die Funktion `fscanf()` den Wert `EOF` ab? (Bitte ankreuzen)

X	In `dummyi`
	In `memo`
	In `dummyf`
	In `&dummyf`

j) Die Funktion `printf()` gibt mit den gelesenen Werten auch den Ausdruck `zaehler[nr]` aus. Steht `zaehler[nr]` (als Bit-Belegung) für eine Zahl, für eine Buchstabenfolge, für eine Speicheradresse? Welche jeweils?

`zaehler[nr]` steht für die Speicheradresse, ab welcher der Name des jeweiligen Zählers gespeichert ist ("Gas: ", "Strom: ", "Wasser:").

k) Nach der Speicherung der Zähler-Werte in `GM-Kosten.txt` wird die Funktion `fflush()` aufgerufen.
   i) Was bewirkt der Aufruf von `fflush()`allgemein?
   ii) (1 Bonuspunkt) Wie oft wird `fflush()` hier aufgerufen?
   iii) Was kann sich am weiteren Lauf des Programms ändern, wenn `fflush()` an dieser Stelle weggelassen wird?

i) `fflush()` veranlaßt Leerung des Datenpuffers vom Programm aus (d.h. Sicherstellung der Daten-Übertragung in die Datei).

ii) `fflush()` wird hier genau einmal aufgerufen (befindet sich außerhalb der `for`-Schleife).

⇒

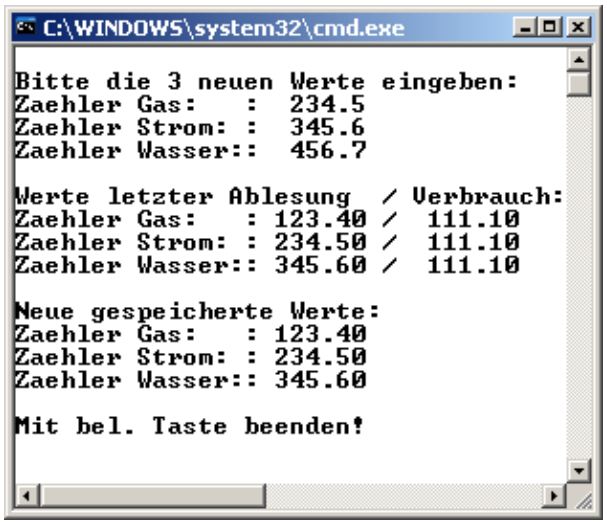
iii) Wird `fflush()` hier ausgelassen, kann es passieren, daß die Werte, die anschließend wieder gelesen werden sollen, noch nicht in der Datei sind.

l) Am Ende des Programms werden die gerade gespeicherten Daten erneut gelesen und ausgegeben. Zuvor wird aber die Funktion `rewind()` aufgerufen.
   i) Was bewirkt der Aufruf von `rewind()` allgemein?
   ii) Was könnte sich am weiteren Lauf des Programms ändern, wenn `rewind()` an dieser Stelle ausgelassen wird?

i) `rewind()` setzt die aktuelle Lese- und Schreib-Position an den Datei-Anfang.

ii) Wird `rewind()` hier ausgelassen, so ist nicht geklärt, ab welcher Stelle in der Datei anschließend gelesen wird.

m) Ab dem zweiten Start von `GasMan` gibt es Probleme (s.Bild): Mehrere Versuche zeigen, daß die Daten-Speicherung offenbar nur in einer leeren Datei funktioniert; auch ein Blick in die Datei bestätigt dies.



Bei der Suche nach den Ursachen stellen Sie zugleich fest, daß dieses Fehlverhalten ausbleibt (d.h.: die Daten werden korrekt gespeichert), wenn Sie den vorausgehenden Code-Abschnitt (mit dem Auslesen des bisherigen Datei-Inhalts) auskommentieren.

Welcher Zusammenhang kann zwischen dem erfolgreichen Auslesen des Datei-Inhalts und dem erfolglosen Schreiben in die Datei bestehen, und wie läßt sich das o.a. Problem beheben?

Bei Wechsel zwischen Datei-Lesen und -Schreiben steht die Plazierung des Positionsindikators nicht mehr zur Verfügung; dies bleibt aus, wenn das Auslesen des bisherigen Datei-Inhalts durch Auskommentieren unterbunden wird.

Abhilfe schafft ein `rewind()`-Aufruf (alternativ: `fseek()`).

```c
                                /* GasMan.c */
#include <conio.h>
#include <stdio.h>

#define ALL 3

int main()
{ FILE *memo;
  char *zaehler[]  = {"Gas:   ", "Strom: ", "Wasser:"};
  float stand[ALL], dummyf=0.;
  int   nr=0, dummyi=0;

  if ((memo=fopen("./GM-Kosten.txt","r+")) ||
      (memo=fopen("./GM-Kosten.txt","w+"))); else return(-1);

 /* Manuelle Eingabe der Werte: */
  printf ("\n\rBitte die %d neuen Werte eingeben:\n\r", ALL);
  for (nr=0; nr<ALL; nr++)
  { printf ("Zaehler %s:  ", zaehler[nr]);
    scanf ("%f", stand+nr);
  }

  /* Bisheriger Datei-Inhalt: */
  printf ("\n\rWerte letzter Ablesung  / Verbrauch:\n\r");
  for (nr=0; nr<ALL; nr++)
  { if ((dummyi=fscanf(memo,"%f", &dummyf))== EOF) dummyf = 0.;
    printf ("Zaehler %s:%7.2f / %7.2f\n",
             zaehler[nr], dummyf, stand[nr]-dummyf);
  }

  /* Daten-Speicherung in Datei: */
  rewind(memo);
  for (nr=0; nr<ALL; nr++)
    fprintf(memo,"%10.2f\n", stand[nr]);  fflush(memo);

  /* Probe-Ausgabe: */
  printf ("\n\rNeue gespeicherte Werte:\n\r");
  rewind(memo);
  for (nr=0; nr<ALL; nr++)
  { fscanf(memo,"%f", &stand[nr]);
    printf ("Zaehler %s:%7.2f\n", zaehler[nr], stand[nr]);
  }

  fclose(memo);
  printf ("\n\rMit bel. Taste beenden!\n\r");  _getch();
   return (0);
}
```

3. Aufgabe (35 Punkte)

Als Freund anspruchsvoller Musik recherchieren Sie im Werk von Frank Sinatra und arbeiten am Programm `Franky.c`, das die sinnigen Texte des alten Meisters rekonstruiert. Mit Hilfe der nächsten Fragen wollen Sie nun das C-Programm am Ende dieser Aufgabe fertigstellen, das sich dann vielversprechend meldet:

Zum u.a. Code wollen Sie nun bitte folgende Fragen beantworten:.

a) Die Funktion `Franky()` enthält einige bitweise Verknüpfungen, die den Programmablauf beeinflussen.

Welcher Zahlenwert steht hinter der Bitkombination `(THEONE | THEOTHER)`?

3

Welche Werte nimmt die Variable `sing` an, während sie die `while`-Schleife steuert (unter Berücksichtigung ihrer Initialisierung und Inkrementierung)?

1, 2, 3

Welche Zahlenwerte für `sing` werden durch den Ausdruck `(sing & 1)` heraussortiert (i) allgemein und (ii) in diesem Anwendungsfall?

i) Mit `(sing & 1)` werden die ungeraden Zahlen heraussortiert .
ii) In diesem Fall sind es: 1 und 3

b) Beim Versuch, den Code zu compilieren, stellen Sie fest, daß es Fehlermeldungen gibt, solange Sie nicht wenigstens den Prototypen von `baby()` an den Code-Anfang stellen. An dieser Situation ändert sich auch nichts, wenn Sie die Reihenfolge der Funktionen innerhalb des Listings ändern. Woran liegt das? (Kurze, allgemeine Erklärung)

Die Funktionen `baby()` und `song()` rufen sich gegenseitig auf, darum wird, unabhängig von der Reihenfolge, bei der Compilierung der einen mindestens die andere dem Compiler unbekannt sein.

Indem die Prototypen dem übrigen Code vorangestellt werden, wird zu Beginn der Compilierung dem Compiler mitgeteilt, welche Funktionen im Programm benötigt werden.

c) Weitere, wenig hilfreiche Compilermeldungen machen Sie aufmerksam darauf, daß sowohl das Argument als auch der Rückgabewert der Funktion `song(lyrics)` default-mäßig als Integer angenommen werden, was bei Betrachtung des Funktionscodes und der Aufrufe in allen Funktionen sicherlich falsch ist. Wie lautet die korrekte Deklaration (erste Zeile) von `song(lyrics)`?

`char *song (char *lyrics)`

d) `Franky()` ruft zuerst die Funktion `song()` auf. Erklären Sie kurz,
   i) wozu dieser Aufruf dient,
   ii) anhand welches Kriteriums in `song()` zwischen dem Aufruf von `Franky()` und den Aufrufen aus den anderen Funktionen unterschieden wird und
   iii) ob es dabei wichtig oder, umgekehrt, unnötig war, die Variable `vocals` als `static` zu deklarieren. (Begründungen!)

i) Der Aufruf von `song()` dient dazu, dort "zentral" die Adresse (`vocals`) des Textes abzulegen, auf die dann auch die anderen Funktionen zugreifen können.

ii) Den Aufruf von `Franky()` erkennt `song()` daran, daß mit `lyrics` keine NULL-Adresse übergeben wird.

iii) Die Deklaration als `static` ist notwendig: Sie stellt sicher, daß der Wert der Variablen `vocals` (die Speicheradresse des Liedertextes) auch für die späteren Aufrufe der anderen Funktionen erhalten bleibt.

e) Der Compiler weist Sie auf den „nichtdeklarierten Bezeichner `gimmemore`" hin. Es handelt sich offenbar um eine globale Variable, die mehrmals verwendet wird, die aber noch nicht angegeben wurde.

Wie lautet ihre korrekte Deklaration, und mit welchem Wert muß sie initialisiert werden, damit der Progamm-Ablauf wie erwartet erfolgt?

`void (*gimmemore)(void) = NULL;`

f) Zum generierten Liedertext tragen alle aufgeführten Funktionen bei. Erhalten dazu die einzelnen Funktionen Kopien des (von Aufruf zu Aufruf wachsenden) Textes, oder greifen sie auf (eine?) einzelne Speicherung(en) zu? Kreuzen Sie bitte die richtige(n) Antwort(en) an:

	Alle Funktionen greifen auf die in `main()` abgelegte Kopie zu.
X	Alle Funktionen greifen auf die in `Franky()`abgelegte Kopie zu.
	Alle Funktionen greifen auf die in `shubiduwa()`abgelegte Kopie zu.
	Alle Funktionen greifen auf die in `baby()` abgelegte Kopie zu.
	Alle Funktionen greifen auf die in `song()` abgelegte Kopie zu.
	Alle Funktionen greifen auf die global abgelegte Kopie zu.
	Alle Funktionen greifen auf die jeweils eigene Kopie zu.

g) Angenommen, die Funktion `Franky()` sei so umgeschrieben, daß die Variable `sing` nacheinander genau die Werte 1 und 2 annimmt. Welcher Liedertext entsteht dann durch das Programm?

"Da-bi / Da-bi / Du / Da / Da / Du / Di-ba / "

h) Der reißende Absatz Ihres Programms läßt Sie darüber nachdenken, den Speicherplatz für den Liedertext zu vergrößern.

Wieviele Zeichen umfaßt der bisher vorgesehene Speicherplatz, und welche Funktion(en) müßte(n) dazu geändert werden?

Bisher kann der Liedertext bis zu 80 Zeichen (inkl. '\0') umfassen.

Für die Speicher-Vergrößerung braucht nur `Franky()` geändert zu werden.

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

#define THEONE   1
#define THEOTHER 2

/*Prototypen:*/
/* ... */
/*Globale Variablen:*/
int  lala=0;
char floskel[9];
void (*gimmemore)(void)=NULL;
```

```
/* ************************************************************** */
   char *song (char *lyrics)
/* ************************************************************** */
{ static char *vocals;
  if (lyrics)  { vocals = lyrics;  return (vocals); }
  if (!gimmemore)
  { sprintf (floskel, "Da-bi / ");
    gimmemore = baby;
  } else
  { sprintf (floskel, "Da / ");
    gimmemore = NULL;
  }
  return (vocals);
}
/* ************************************************************** */
   void baby (void)
/* ************************************************************** */
{ char *loveMe=NULL;
  int   j1=0;
  loveMe = song(loveMe);
  for (j1=0; j1<2; j1++)
  { lala += sprintf (loveMe+lala, "%s", floskel); }
  lala += sprintf (loveMe+lala, "Du / ");
  lala += sprintf (loveMe+lala, "Di-ba / ");
  return;
}
/* ************************************************************** */
   void shubiduwa (void)
/* ************************************************************** */
{ char *oBabe=NULL;
  int   j1=0;
  oBabe = song(oBabe);
  for (j1=0; j1<2; j1++)
  { lala += sprintf (oBabe+lala, "%s", floskel); }
  lala += sprintf (oBabe+lala, "Du / ");
  return;
}
/* ************************************************************** */
   void Franky (void)
/* ************************************************************** */
{ char album [80];
  int  sing=1;
  printf ("\n\rHere comes Fraky's famous \"Da-bi / ... \":\n\n\r");
  song(album);
  while (sing & (THEONE | THEOTHER))
  { if (sing & 1) shubiduwa(); else gimmemore(); sing++;
  }
  printf ("%s", album);
  return;
}
/* ************************************************************** */
   int main(void)
/* ************************************************************** */
{ Franky();  _getch();
  return(0);
}
```

4. Aufgabe (10 Punkte)

Das folgende kurze Programm verwendet teils globale, teils lokale Variablen. Es gibt dann untereinander den Namen der jeweils aufgerufenen C-Funktion und das errechnete Ergebnis aus.

Ergänzen Sie bitte die unten vorbereitete Ausgabe!

```
#include <conio.h>  //wg._getch()
#include <stdio.h>

int x=2;

int squareL(int x)
{ return (x*x);
}

int squareG()
{ return (x*x);
}

int main()
{ int x=2;
  x=squareG();  printf ("squareG: x^2=%d\n", x);
  x=squareL(x); printf ("squareL: x^2=%d\n", x);
  x=squareG();  printf ("squareG: x^2=%d\n", x);
  x=squareL(x); printf ("squareL: x^2=%d\n", x);
  x=squareG();  printf ("squareG: x^2=%d\n", x);
 _getch();
  return 0;
}
```

Ausgabe:

```
squareG: x^2=  4
squareL: x^2= 16
squareG: x^2=  4
squareL: x^2= 16
squareG: x^2=  4
```

Platz für Notizen: