

# Grundlagen der Informatik III

Wintersemester 2010/2011

Wolfgang Heenes, Patrik Schmittat



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## 4. Aufgabenblatt

22.11.2010

**Hinweis:** Der Schnelltest und die Aufgaben sollen in den Übungsgruppen bearbeitet werden. Die Hausaufgaben sind in der Kalenderwoche 48 (29.11. bis 03.12.) bei den Tutoren in **physikalischer Form** (handschriftlich oder gedruckt) abzugeben. Bei allen Abgaben ist der Name des Tutors und die Übungsgruppe deutlich anzugeben. Bei Teamabgaben wird nur eine Lösung eingereicht, die alle Namen der Teammitglieder enthält.

Schicken Sie Ihre Lösungen von Programmieraufgaben *zusätzlich* zur schriftlichen Abgabe per E-Mail an Ihren Tutor. Kommentieren Sie Ihren Quellcode.

### Aufgabe 1: Schnelltest

| Fragen  | Antworten   |
|---|---|
| 1. Welche der folgenden Aussagen über die Speicherverwaltung in C sind korrekt?   | <input type="checkbox"/> Lokale Variablen in Funktionen müssen explizit wieder freigegeben werden.<br><input type="checkbox"/> Mit dem Befehl <code>malloc</code> wird Speicher auf dem Stack reserviert.<br><input type="checkbox"/> Der mit <code>malloc</code> angeforderte Speicher muss wieder explizit freigegeben werden.<br><input type="checkbox"/> Der Garbage-Collector gibt periodisch nicht mehr verwendeten Speicher frei.<br><input type="checkbox"/> Der Angeforderte Speicher wird automatisch mit Nullen initialisiert.                                 |
| 2. Welche Aussagen über Funktionen und Funktionsparameter in C sind richtig?      | <input type="checkbox"/> Funktionsparameter werden als Kopie übergeben ( <i>call-by-value</i> ).<br><input type="checkbox"/> Eine Referenzübergabe von Parametern ( <i>call-by-reference</i> ) ist nicht möglich.<br><input type="checkbox"/> Eine Referenzübergabe von Parametern ist mittels Pointer möglich.<br><input type="checkbox"/> Die Übergabe von Arrays erfolgt über Pointer.   |
| 3. Welche der folgenden Behauptungen über Längen von Arrays in C sind zutreffend? | <input type="checkbox"/> Die Funktion <code>sizeof</code> ermittelt die Länge eines Arrays.<br><input type="checkbox"/> Die Funktion <code>sizeof</code> ermittelt die Größe des Speichers, den ein Array belegt.<br><input type="checkbox"/> Die Funktion <code>length</code> ermittelt die Länge eines Arrays.<br><input type="checkbox"/> Es gibt keine Funktion zur Ermittlung der Länge eines Arrays.<br><input type="checkbox"/> Keine dieser Aussagen ist zutreffend.  |
| 4. Betrachten Sie die folgende C-Deklaration: <code>int a[5], *p;</code>          | <input type="checkbox"/> Innerhalb ihres Scopes können die Variablen <code>a</code> und <code>p</code> beliebig ausgetauscht werden, ohne dass der Compiler einen Fehler meldet.<br><input type="checkbox"/> <code>a</code> und <code>p</code> sind nicht zuweisungskompatibel ( <code>a = p</code> erzeugt einen Compilerfehler).<br><input type="checkbox"/> Die Zuweisung <code>a[0] = *p;</code> wird vom Compiler akzeptiert.<br><input type="checkbox"/> Die Zuweisung <code>*a = p[0];</code> wird vom Compiler akzeptiert.<br><i>Fortsetzung nächste Seite...</i> |

| Fragen   | Antworten   |
|--|---|
|  | <input type="checkbox"/> Die Schreibweise *p ist äquivalent zu p[0].  |
| 5. Welche Aussagen zu boole'schen Ausdrücken in C ist korrekt? | <input type="checkbox"/> In C sind der Datentyp bool sowie die Ausdrücke true und false vorhanden.<br><input type="checkbox"/> Alle Zahlenwerte größer null stellen implizit den logischen Wert true dar.<br><input type="checkbox"/> Alle Zahlenwerte kleiner gleich null stellen implizit den logischen Wert false dar.<br><input type="checkbox"/> Zuweisungen von Zahlenwerten können direkt in logischen Ausdrücken verwendet werden, um die zugewiesene Zahl auf Ungleichheit mit null zu prüfen. |

## Aufgabe 2: Informationsdarstellung in C

Schreiben Sie ein C Programm (ANSI-C), welches eine einzugebene Dezimalzahl als Oktal-, Hexadezimalzahl bzw. als ASCII-Zeichen ausgibt. Bevor die Dezimalzahl eingegeben wird, soll der Benutzer auswählen, welche Darstellung gewünscht wird. Das Auswahlmenü verwendet O für Oktal, H für Hexadezimal und A für ASCII. Die Auswahl soll **nicht** sensitiv auf Groß-/Kleinschreibung reagieren.

## Aufgabe 3: Monte-Carlo-Simulation

Monte-Carlo-Simulationen können zur Berechnung des Inhalts regel- und unregelmäßiger Flächen und Körper verwendet werden. Die Algorithmen werden als Monte-Carlo-Algorithmen bezeichnet und gehören in die Klasse der randomisierten Algorithmen. Das Verfahren der Monte-Carlo-Simulation soll anhand der probabilistischen Berechnung der Kreiszahl  $\pi$  verdeutlicht werden. Hierzu werden zufällige Punkte  $\{(x,y)|x \in [-1,1], y \in [-1,1]\}$  generiert und überprüft, ob diese innerhalb des Einheitskreises liegen. Die sich ergebende Wahrscheinlichkeitsverteilung erlaubt die Berechnung der Kreiszahl  $\pi$  nach folgender Formel.

$$\frac{\text{Treffer in Kreisflaeche}}{\text{generierte Punkte im Rechteck}} = \frac{r^2 \cdot \pi}{(2 \cdot r)^2} = \frac{\pi}{4}$$

- Schreiben Sie ein C Programm, welches die Kreiszahl  $\pi$  berechnet und ausgibt. Die Funktion rand() liefert Zufallszahlen.<sup>1</sup>
- Testen Sie Ihr Programm mit unterschiedlicher Anzahl von Punkten (im Bereich von ca. 100000 bis 100000000). Notieren Sie die Anzahl der Punkte und die Genauigkeit in einer Tabelle.
- Die Berechnung von  $\pi$  kann auch auf ein Flächenintegral zurückgeführt werden.

$$\int_0^1 \frac{4}{1+x^2} dx$$

Schreiben Sie ein C Programm, welches die Integration (z. B. Trapezintegration) ausführt und vergleichen Sie die Ergebnisse mit b).

## Aufgabe 4: Inline-Assembler-Code in C

Assembler-Programme werden häufig zur Optimierung von Programmteilen eingesetzt, die in einer Hochsprache z. B. C vorliegen. Um den Code nicht vollständig neu zu schreiben, erlauben es moderne Compiler sogenannten Inline-Assembler-Code in ein Programm einzubinden.

Ein einfaches Beispiel zeigt das folgende C Programm:

<sup>1</sup> Binden Sie falls notwendig die entsprechenden Libraries ein. Achten Sie bei rand() darauf vorher einen zufälligen Seed zu setzen.

```

#include <stdio.h>

main()
{
    int n = 33;

    asm("imull,%%eax;"
        : "=a" (n)
        : "a" (n)
        );

    printf("Das Quadrat von 33 ist %i\n", n);
}

```

Das Programm berechnet das Quadrat der Zahl 33 und gibt das Ergebnis auf dem Bildschirm aus.

Inline-Assembler-Code wird mit dem Schlüsselwort **asm** eingebettet.

```

asm ( assembler template
      : output operands           (optional)
      : input operands            (optional)
      : list of clobbered registers (optional)
    );

```

Der Assembler-Code (hier als assembler template) bzw. die Befehle werden von “ ” umschlossen. Für die korrekte Funktionsweise des Inline-Assembler-Code müssen die Variablen von C den zu verwendenden Registern zugeordnet werden. Dies geschieht mit durch die Angabe von output- und input operands.

Die Zuweisung an die Register wird über die Angabe folgender Bezeichner erreicht.

|   |  |      |
|---|--|------|
| a |  | %eax |
| b |  | %ebx |
| c |  | %ecx |
| d |  | %edx |
| S |  | %esi |
| D |  | %edi |

Die Angabe (Syntax beachten) im obigen Programm stellt also eine Zuweisung der C Variablen *n* an das Register *%eax* da. Achtung die Register werden im Inline-Assembler-Code mit *%%eax* usw. bezeichnet.

- a) Zur Vorbereitung der Hausaufgabe überzeugen Sie sich von der korrekten Ausführung des C Programms mit Inline-Assembler-Code.

## Hausaufgabe 1: Inline-Assembler-Code in C, Leistungsvergleich (5 Punkte)

Aufbauend auf Aufgabe 3 sollen zwei Implementierungen verglichen werden. Betrachtet wird folgendes C Programm.

```

#include <stdio.h>

main()
{
    int i;
    int result = 1;
    int n = 12;
    for (i = 2; i <= n; i++)
        result *= i; // (result = result * i)

    printf("Fakultaet_%d\n", result);
}

```

- a) Schreiben Sie ein IA32-Assemblerprogramm welches obiges C Programm realisiert. Testen Sie ihr Programm für  $n = 12$ .
- b) Ersetzen Sie in dem vorgegebenen C Programm die Berechnung durch Ihr IA32-Assemblerprogramm. Dazu soll der in Übung 4 vorgestellte Rahmen für Inline-Assembler verwendet werden.<sup>2</sup> Überzeugen Sie sich von der korrekten Funktion des Programms für  $n = 12$ .
- c) Vergleichen Sie die Laufzeiten der beiden Programme. Dazu bietet es sich an, die Berechnung in eine for-Schleife einzubetten und z. B. die Berechnung 100000000 zu wiederholen (das Programm sollte schon 3 - 4 Sekunden laufen). Die Zeitmessung können Sie z. B. unter Unix mit `time` vornehmen. Sollte Ihr IA32-Assemblerprogramm langsamer laufen, sollten Sie sich noch ein paar Gedanken über Ihre Implementierung machen und diese optimieren.

## Hausaufgabe 2: Reverse Engineering (5 Punkte)

Gegeben ist folgender Object Dump:

```

080483a4 <up>:
80483a4: 55                push   %ebp
80483a5: 89 e5            mov    %esp,%ebp
80483a7: 83 ec 18        sub   $0x18,%esp
80483aa: 83 7d 08 00     cmpl  $0x0,0x8(%ebp)
80483ae: 75 09          jne   80483b9 <up+0x15>
80483b0: c7 45 ec 00 00 00 00  movl  $0x0,-0x14(%ebp)
80483b7: eb 36          jmp   80483ef <up+0x4b>
80483b9: 83 7d 08 01     cmpl  $0x1,0x8(%ebp)
80483bd: 75 09          jne   80483c8 <up+0x24>
80483bf: c7 45 ec 01 00 00 00  movl  $0x1,-0x14(%ebp)
80483c6: eb 27          jmp   80483ef <up+0x4b>
80483c8: 8b 45 08        mov   0x8(%ebp),%eax
80483cb: 83 e8 02        sub   $0x2,%eax
80483ce: 89 04 24        mov   %eax,(%esp)
80483d1: e8 ce ff ff ff  call  80483a4 <up>
80483d6: 89 45 fc        mov   %eax,-0x4(%ebp)
80483d9: 8b 45 08        mov   0x8(%ebp),%eax
80483dc: 83 e8 01        sub   $0x1,%eax
80483df: 89 04 24        mov   %eax,(%esp)
80483e2: e8 bd ff ff ff  call  80483a4 <up>
80483e7: 8b 55 fc        mov   -0x4(%ebp),%edx
80483ea: 01 c2          add   %eax,%edx
80483ec: 89 55 ec        mov   %edx,-0x14(%ebp)
80483ef: 8b 45 ec        mov   -0x14(%ebp),%eax
80483f2: c9            leave
80483f3: c3            ret

080483f4 <main>:
...
80483fe: 55                push   %ebp
80483ff: 89 e5            mov    %esp,%ebp
8048401: 51                push   %ecx
8048402: 83 ec 24        sub   $0x24,%esp
8048405: c7 45 f4 14 00 00 00  movl  $0x14,-0xc(%ebp)
804840c: 8b 45 f4        mov   -0xc(%ebp),%eax
804840f: 89 04 24        mov   %eax,(%esp)
8048412: e8 8d ff ff ff  call  80483a4 <up>
...

```

<sup>2</sup> Informationen finden sich auch unter <http://www.ibm.com/developerworks/linux/library/l-ia.html>

- 
- a) Analysieren Sie den Object Dump. Was für eine Funktion wird realisiert.
  - b) Implementieren Sie das Programm in C und testen Sie die Funktionsweise.