Take Home Exam
Programmierung für Naturwissenschaften 2

im Sommersemester 2021

19.07.2021, 10:00 Uhr, Dauer: 90 Minuten

virtuell

25 Fragen, maximal 90 Punkte

Prüfer: Stefan Kurtz und Andrew Torda

Name: _____

Matrikelnummer: _____

Unterschrift: _____

Anmerkungen

Sie sollten alle Aufgabenstellungen zu Beginn einmal
komplett durchlesen und dann mit den einfachen beginnen.

Es ist jeweils Programmcode gefragt, der im Wesentlichen syntaktisch und semantisch korrekt ist. Kleinere Fehler werden toleriert, solange der Sinn des Codes ersichtlich ist.

Zu allen Aufgaben ist links von der laufenden Nummer der Frage
in runden Klammern die maximal mögliche Punktzahl angegeben.

Aufgabe:	1	2	3	4	5	6	7	8	9	10	11	12	13
Punkte:	3	3	12	2	8	1	7	8	12	6	9	1	1
erreicht:													

Aufgabe:	14	15	16	17	18	19	20	21	22	23	24	25	
Punkte:	1	1	1	1	1	1	1	2	1	4	1	2	
erreicht:													

Gesamtzahl erreichter Punkte:

Aufgaben zur C-Programmierung

(3) 1. Beschreiben Sie kurz, was die folgenden Zeilen in einem Makefile bedeuten:

```
CC?=clang
CFLAGS=-g -Wall -Werror -O3

%.o: %.c
        ${CC} $< -c ${CFLAGS} -o $@
```

Nehmen wir an, dass es im aktuellen Verzeichnis nur zwei Dateien `program.c` und `Makefile` gibt, wobei letztere Datei den obigen Inhalt hat. Geben Sie vollständig das Kommando an, das ausgeführt wird, wenn wenn man im Terminal den Befehl `make program.o` ausführt. Begründen Sie Ihre Antwort.

(3) 2. Wenn man in C einen Datentyp implementiert, dann ist es sinnvoll, die Schnittstelle von der Implementierung zu trennen. Der Benutzer oder die Benutzerin des Datentyps sollte nur die Schnittstelle kennen, die in der Header-Datei spezifiziert ist, aber nichts über die Implementierung selbst wissen müssen. Man spricht in diesen Zusammenhang von *Data hiding*.

Nehmen wir an, es soll ein Binärbaum implementiert werden, und zwar in den Dateien `bintree.c` (für die Implementierung) und `bintree.h` (für die Schnittstelle). Dazu soll ein Typname `Bintree` eingeführt werden, der ein Synonym für eine Struktur ist. Diese Struktur soll jeweils einen Zeiger auf ein linkes und rechtes Kind vom Typ `Bintree` und einen Wert `data` vom Typ `int` enthalten. Geben Sie entsprechende C-Deklarationen des Typsynonyms `Bintree` und der genannten Struktur an. Geben Sie, unter Berücksichtigung des *Data Hiding*-Prinzips, an, in welchen Dateien diese Deklarationen erfolgen und welche `include`-Anweisung in der C-Datei vorkommen müssen.

(12) 3. Wenn man Textdateien unter MS-Windows erstellt, dann werden Zeilen jeweils durch `\r\n` abgeschlossen. Das sind zwei aufeinanderfolgende Zeichen. Unter Linux ist es üblich, nur das Zeichen `\n` als Zeilenende zu verwenden. Implementieren Sie eine C-Funktion

```
size_t replace_line_terminator(unsigned char *s,size_t n)
```

die im Speicherbereich, der durch den Zeiger `s` referenziert wird, die Ersetzung von `\r\n` durch `\n` vornimmt, ohne eine Kopie zu erzeugen. Die Länge des Strings, in dem die Ersetzung vorgenommen werden soll, ist `n`. Der genannte Speicherbereich kann das Zeichen `\0` enthalten. Dieses Zeichen hat in diesem Kontext aber keine besondere Bedeutung. Als `return`-Wert soll die Funktion die Länge des Strings nach der Ersetzung liefern. Beachten Sie, dass sich nach Ersetzung von `\r\n` durch `\n` die danach folgenden Zeichen nach links verschieben. In der folgenden Tabelle finden Sie Beispiele:

`s`	`n`	`s`$[0,\ldots,m-1]$ nach Ersetzung	`return`-Wert m
`acgt\r\nabc\r\n`	11	`acgt\nabc\n`	9
`a\rbc\r\n`	6	`a\rbc\n`	5
`acgt\r\nabc\n`	10	`acgt\nabc\n`	9
`at\n\rabc\n`	8	`at\n\rabc\n`	8
`\r\n\r\n`	4	`\n\n`	2

(2) 4. Nehmen wir an, Sie wollen die Funktion `qsort` aus der C-Standard-Bibliothek mit verschiedenen Vergleichsfunktionen aufrufen und müssen dazu ein Array von Zeigern auf solche Vergleichsfunktionen speichern. Geben Sie eine `typedef`-Deklaration für den Basistyp `QsortCompareFunction` des Arrays an. Hier ist zur Erinnerung die Vorwärts-Deklaration der Funktion `qsort`:

```
void qsort(void *base, size_t num_elems, size_t size_elem,
           int (*compare)(const void *, const void *));
```

(8) 5. Implementieren Sie, basierend auf der Funktion `qsort` aus der C-Standardbibliothek, eine C-Funktion

```
void sort_complex(Complex *tab, size_t len)
```

zum Sortieren eines Arrays von komplexen Zahlen, die jeweils durch den folgenden Typ repräsentiert werden:

```
typedef struct
{
  double real, imag;
} Complex;
```

Für eine komplexe Zahl $z = a + b\mathtt{i}$ wird der reelle Anteil a in der Strukturkomponente `real` gespeichert und der imaginäre Anteil b in der Strukturkomponente `imag`.

Das Array, das durch den Zeiger `tab` referenziert wird und `len` Einträge hat, soll aufsteigend nach den Absolutwerten der komplexen Zahlen sortiert werden. Der Absolutwert $|z|$ einer komplexen Zahl $z = a + b\mathtt{i}$ ist definiert durch $|z| = \sqrt{a^2 + b^2}$. Zur Berechnung der Quadratwurzel können Sie die Funktion `sqrt` aus der mathematischen Bibliothek von C verwenden.

Zur Erinnerung geben wir hier nochmal den Typ der Funktion `qsort` an:

```
void qsort(void *base, size_t num_elems, size_t size_elem,
           int (*compare)(const void *, const void *));
```

(1) 6. In der Vorlesung wurden eindeutige Konventionen bzgl. der Begriffe Zeilen und Spalten einer Matrix getroffen. Diese sollen auch hier gelten. Es ist nur eine der folgenden vier Aussagen korrekt. Markieren Sie die korrekte Aussage.

1. In C deklariert man eine Matrix `M` über dem Basistyp `double` mit 3 Spalten und 2 Zeilen durch `double M[3][2];`
2. In C deklariert man eine Matrix `M` über dem Basistyp `double` mit 3 Spalten und 2 Zeilen durch `double M[2][3];`
3. In C deklariert man eine Matrix `M` über dem Basistyp `double` mit 3 Spalten und 2 Zeilen durch `double M[2,3];`
4. In C deklariert man eine Matrix `M` über dem Basistyp `double` mit 3 Spalten und 2 Zeilen durch `double M[3,2];`

Aufgaben zu numerischen und kombinatorischen Problemen

(7) 7. Schreiben Sie eine C-Funktion

```
char *random_sequence_iid(const char *alphabet, size_t n)
```

die eine zufällige Sequenz der Länge `n` über dem Alphabet `alphabet` liefert. Für alle Positionen ist die Wahrscheinlichkeit des Vorkommens eines Zeichens $\frac{1}{k}$ für alle Zeichen eines Alphabets der Größe k. `alphabet` ist ein `\0`-terminierter String, der keine Duplikate enthält. Für jede Position der Sequenz soll die Bestimmung eines zufälligen Zeichens in einer konstanten Anzahl von Berechnungsschritten erfolgen, also insbesondere nicht von Alphabetgröße abhängen. Der Rückgabewert der Funktion soll ein Zeiger auf einen Speicherbereich mit der Zufallssequenz sein. Die Sequenz ist nicht `\0`-terminiert. Benutzen Sie die Funktion `drand48()` zum Generieren von Zufallszahlen. Sie können davon ausgehen, dass `drand48()` konstante Zeit benötigt. Um die Initialisierung des Seeds für den Zufallsgenerator brauchen Sie sich nicht zu kümmern.

(8) 8. In der Vorlesung wurde gezeigt, wie man mit Hilfe einer Monte-Carlo Simulation eine Teilfläche in einem Einheitsquadrat bestimmen kann. Vorraussetzung ist, dass man die Funktion kennt, die genau dann `true` zurückliefert, wenn ein Punkt zur Teilfläche gehört. Die Idee besteht darin, eine zufällige Stichprobe von Punkten im Einheitsquadrat zu generieren und den Anteil der in der Teilfläche liegenden Punkte als Schätzung der Größe der Teilfläche zu verwenden.

Eine analoge Monte-Carlo Simulation kann man auch zur Schätzung des Volumens eines Teils eines Einheitsquaders (mit Kantenlängen 1) verwenden. Nehmen wir an, dass eine Funktion `is_inside` bekannt ist. Diese hat drei Parameter `x`, `y` und `z`, jeweils vom Typ `double`, die die X-, Y, und Z-Koordinaten eines Punktes im Einheitsquader angeben (d.h. $0 \leq x, y, z \leq 1$). Die Funktion liefert genau dann `true` zurück, wenn der entsprechende Punkt im betrachteten Teil des Einheitsquaders liegt. Schreiben Sie eine C-Funktion

```
double estimate_volume(bool (*is_inside)(double,double,double),
                       size_t num_points)
```

die eine Schätzung des Volumens des Teils des Einheitsquaders, entsprechend der Funktion `is_inside` liefert. Der Parameter `num_points` ist die Anzahl der Punkte in der Stichprobe. Zum Erzeugen der Koordinaten eines Punktes aus der Stichprobe können Sie die Funktion `drand48()` verwenden.

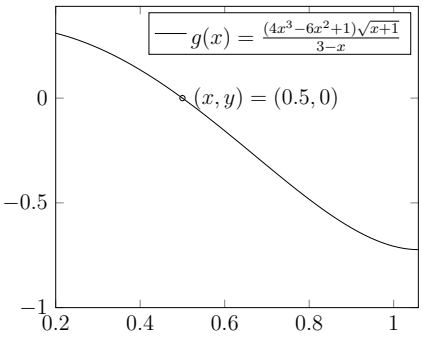
(12) 9. Sei $f : \mathbb{R} \to \mathbb{R}$ eine Funktion und sei $\ell, r \in \mathbb{R}$ mit $\ell \leq r$, so dass $f(x) \geq f(x')$ für alle $\ell \leq x \leq x' \leq r$. D.h. f ist monoton fallend im Intervall $[\ell, r]$. Es geht nun darum, für ein gegebenes $y \in \mathbb{R}$ mit $f(r) \leq y \leq f(\ell)$ ein $x \in \mathbb{R}$ zu bestimmen, so dass $f(x) = y$ ist. Durch die Monotonieeigenschaft kann man das gesuchte x effizient mit Hilfe einer binären Suche berechnen.

Schreiben Sie eine C-Funktion

```
double invert_decreasing_function(double (*f)(double), double l,
                                  double r, double y)
```

die für eine Funktion `f`, die monoton fallend ist im Intervall `[l,r]` und für ein `y` einen Wert `x` mit der Eigenschaft `f(x)=y` zurückliefert. Die Funktion darf keine Rekursion verwenden. Verwenden Sie eine binäre Suche, die in jedem Schritt ein neues Teilintervall, in dem der gesuchte Wert liegt, berechnet. Die Iteration soll stoppen, wenn die Breite des Teilintervalls kleiner als eine vorgegebene Konstante `TOL` ist. In diesem Fall soll die Intervallmitte zurückgegeben werden.

Beispiel: $g(x) = \frac{(4x^3-6x^2+1)\sqrt{x+1}}{3-x}$ ist monoton fallend im Intervall $[0.2, 1.0584]$. Für $y = 0$ ist $g(0.5) = y$. Die zu implementierende Funktion muss also beim Aufruf mit den Parametern `g, 0.2, 1.0584, 0` durch eine `return`-Anweisung den Wert 0.5 zurückliefern.



Aufgaben zur C++-Programmierung

(6) 10. In der Vorlesung wurde die folgende rekursive C-Implementierung des Quicksort-Algorithmus vorgestellt. Hier ist die Implementierung für Arrays mit dem Basistyp `size_t`:

```
static size_t distribute_elements(size_t *buf, size_t *a, size_t len) {
  size_t sidx = 0, lidx = len - 1, pivot = a[len-1];

  for (size_t idx = 0; idx < len - 1; idx++)
    if (a[idx] < pivot)
      buf[sidx++] = a[idx];
    else
      buf[lidx--] = a[idx];
  for (size_t idx = 0; idx < len; idx++) a[idx] = buf[idx];
  a[lidx] = pivot;
  return lidx;
}

void quicksort(size_t *a, size_t *buf, size_t len) {
  if (len > 1) {
    size_t pivotidx = distribute_elements(buf, a, len);
    quicksort(a, buf, pivotidx);
    quicksort(a+pivotidx+1, buf+pivotidx+1, len - (pivotidx + 1));
  }
}

void quicksort_all(size_t *a, size_t len) {
  size_t *buf = malloc(len * sizeof *buf);
  quicksort(a, buf, len);
  free(buf);
}
```

Ergänzen und modifizieren Sie den C-Code so, dass eine C++-Implementierung mit einem Template Typ `T` entsteht, damit Arrays von Werten eines Typs `T` sortiert werden können. Sie können davon ausgehen, dass Werte vom Typ `T` durch < verglichen werden können. Die dynamische Speicherallokation soll durch `new` und `delete` erfolgen. Anstatt den Code zu modifizieren, können Sie aber auch gerne beschreiben, welche Änderungen notwendig sind.

Aufgaben zu Shared Memory Multithreading

(9) 11. Die rekursive Funktion

```
size_t f(size_t n) { return n<=1 ? 1 : (f(n-1) + f(n-2)); }
```

soll auf alle Werte in einer Tabelle `table` von `num_tasks` ganzen Zahlen vom Typ `size_t` angewendet werden, und zwar so, dass an der i-ten Stelle in der Tabelle nach Ausführung von `f` der Wert `f(table[i])` steht. Wenn man nur einen Thread verwendet, kann man z.B. die folgende Schleife implementieren:

```
for (size_t i = 0; i < num_tasks; i++) table[i] = f(table[i]);
```

Hier geht es aber darum, die Werte in `table` mit mehreren Threads gleichzeitig zu berechnen. In der Vorlesung wurde gezeigt, wie man eine Funktion

```
void pfn_run_threaded(size_t k, size_t num_tasks,
                      PfNThreadFunc thread_proc, void *thread_data);
```

implementiert, die eine Funktion `thread_proc` mit `k` Threads auf Werte zwischen 0 und `num_tasks-1` anwendet. Jeder dieser Werte kann als Nummer einer Aufgabe interpretiert werden. Der Typ `PfNThreadFunc` ist dabei wie folgt definiert:

```
typedef void (*PfNThreadFunc)(size_t thread_id, size_t task_num,
                              void *thread_data);
```

Es wird nun jede Aufgabennummer als Index in `table` interpretiert, an dem zunächst das Argument `n` für die Funktion `f` steht und dann der Funktionswert `f(n)` gespeichert wird.

Implementieren Sie nun eine Funktion `eval_f`, so dass

```
pfn_run_threaded(k, num_tasks, eval_f, (void *) table);
```

die Funktion `f` mit `k` Threads auf `num_tasks` Werte in `table` anwendet. Es soll das gleiche Ergebnis entstehen, wie bei der Berechnung mit einem Thread.

(1) 12. Wie nennt man das Problem, das auftritt, wenn mehrere Threads gleichzeitig auf Variablen an der gleichen Adresse zugreifen und mindestens ein Thread dabei den Wert der Variablen ändert.

Aufgaben zur R-Programmierung

(1) 13. Welchen Datentyp hat die Variable `a` nach der folgenden Wertzuweisung in R?

```
a = c(1.0, 2.0, 3.0)
```

(1) 14. Welchen Datentyp hat die Variable `a` nach der folgenden Wertzuweisung in R?

```
a = c(F, T, F)
```

(1) 15. Sei `a` eine Variable in R, die an einen Ausdruck gebunden ist. Geben Sie einen solchen Ausdruck an, so dass bei der Auswertung von `a` in der R-Befehlszeile die folgende Ausgabe erscheint:

```
[[1]]
[1] "protein"

[[2]]
[1] 1

[[3]]
[1] FALSE
```

(1) 16. Sei `a` ein Vektor mit numerischen Werten. Das folgende R-Script berechnet die Summe der Elemente in `a`.

```
s <- 0
for (x in a) {
     s <- s + x
}
```

Schreiben Sie eine einzelne Anweisung in R, die das gleiche Ergebnis berechnet.

(1) 17. Gegeben sei die folgende R-Anweisung:

```
x <- c(1, 2, 3, 4, 5)
```

Geben Sie einen syntaktisch korrekten Ausdruck in R an, der einen Vektor mit allen Elementen aus `x`, die größer als 3 sind, liefert.

(1) 18. Gegeben sei die folgenden R-Anweisung:

```
df <- read.table("data.txt", header = TRUE)
```

Welchen Datentyp hat `df`?

(1) 19. Was bedeutet das Symbol λ in einer exponentiellen Verteilung der Form $\lambda e^{-\lambda t}$?

(1) 20. Nennen Sie eine von zwei Eigenschaften von Ereignissen, deren Verteilung einer Poisson-Verteilung entspricht.

(2) 21. Evaluieren Sie den folgenden Ausdruck algebraisch, nicht numerisch:

$$\lim_{x \to 0} (\ln(x+1))$$

(1) 22. Wir betrachten das folgende Fragment eines R-Skriptes:

```
f <- function (x) {
     z <- 0
     for (y in x) {
          z <- z + y
     }
     z/length(x)
}

k <- f(c(1, 2, 3, 4, 5))
```

Welchen Wert hat `k`?

(4) 23. Aus einer nichtlinearen Regression auf $y = e^{-\beta x} \sin(2\pi\omega x + \varphi)$ bekam ich einen Wert von $\varphi = 7$. Schreiben Sie eine R-Funktion, die ein beliebiges x nimmt und in einen Bereich von $(0 \ldots 2\pi)$ verschiebt. R hat eine Variable, `pi` mit einem vordefinierten Wert. Verwenden Sie diesen Rahmen

```
squash_to_range <- function (x) {
    ..
    return (x)
}
```

(1) 24. Wir betrachten den folgende Ausdruck, in dem β eine positive Zahl ist:

$$y = \mathrm{e}^{-\beta x} \sin(2\pi\omega x + \phi)$$

Was ist die Bedeutung bzw. der Einfluss von β (für $\beta > 0$) in diesem Ausdruck?

(2) 25. Wir betrachten die folgende Anweisung in R:

```
a <- replicate(100,f(m,n))
```

Welche syntaktischen Eigenschaften müssen die Bezeichner `f`, `m` und `n` haben? (1 Punkt)

Welchen Datentyp hat `a`? (1 Punkt)