Prof. Dr. Carsten Meyer
Fachbereich Informatik & Elektrotechnik
Fachhochschule Kiel

Programmieren in C – Wintersemester 2019 / 20

Aufgabenblatt 5

Abgabe: Dienstag, 07.01.2020, 08:30 Uhr (schriftlich und elektronisch)

Aufgabe 1 (Kontrollstrukturen: for / while, switch):

a) Schreiben Sie ein Programm, das unter Verwendung verschachtelter for-Schleifen zählt, wie viele Kombinationen beim einfachen Zahlen-Lotto 6 aus 49 (so genannter Binomialkoeffizient 49 über 6) möglich sind.

WICHTIG: Beachten Sie, dass jede Zahl nur 1 Mal angekreuzt werden kann. Eine Kombination wie beispielsweise 4, 4, 12, 12, 12, 46 ist daher nicht möglich und muss ausgeschlossen werden, um zum richtigen Ergebnis zu gelangen.

HINWEIS: Da die Reihenfolge der gezogenen Zahlen in einer Kombination beliebig ist, können Sie sich eine geeignete Anordnung der Zahlen überlegen, um die Schleifen möglichst einfach zu gestalten (denken Sie daran, wie die Lottozahlen in den Medien dargestellt werden).

Die Lösung soll das Programm und die Programmausgabe umfassen, die die Anzahl der Kombinationen auf dem Bildschirm ausgibt.

b) Erstellen Sie ein Programm, das eine Textzeile einliest und den Anteil der verschiedenen Vokale in der Textzeile mitzählt (unabhängig von Groß- und Kleinschreibung!). Geben Sie den prozentualen Anteil vom jeweiligen Vokal im Text aus (bezogen auf die Gesamtanzahl der Zeichen in der Textzeile).

ACHTUNG: Diese Aufgabe soll *ohne* Felder (Arrays) gelöst werden. Lesen Sie also einzelne Zeichen (z.B. mit der Bibliotheksfunktion **`int getchar(void)`** aus **`stdio.h`** über eine **`while`**-Schleife) ein und verwenden Sie eine **`switch`**-Anweisung, um die Vokale zu unterscheiden.

Die Lösung soll das Programm und die Programmausgabe umfassen.

(Nach Schwenk, Uni Bochum)

Aufgabe 2 (Funktionen, Kontrollstrukturen):

Schreiben Sie ein Programm mit einer Funktion **`int hoch (int x, int y)`**, die beliebige Potenzen x^y (mit nicht-negativem, ganzzahligem x und y) berechnet. Hierzu soll im Hauptprogramm mittels separater **`do-while`**-Schleifen solange Basis x und Exponent y abgefragt werden, bis jeweils nicht-negative Werte eingegeben werden. Anschließend wird x^y über die Funktion **`hoch`** berechnet. Das Ergebnis der Berechnung soll dann im Hauptprogramm ausgegeben werden (wobei der Fall 0^0 gesondert als nicht definiert gekennzeichnet werden möge, s.u.).

Beachten Sie für die Implementierung der Funktion **`hoch`** folgende Hinweise:

- Der Rückgabewert der Funktion **`hoch`** ist das Ergebnis der Berechnung von x^y.
- Es ist zweckmäßig, zur Berechnung von x^y eine **`for`**-Schleife zu verwenden.
- Für $x \neq 0$ gilt: $x^0 = 1$
- Für $y \neq 0$ gilt: $0^y = 0$
- 0^0 ist (in dieser Aufgabe) nicht definiert. Daher soll Ihre Funktion in diesem Fall –1 als Zeichen für den Fehlerfall zurückgeben.

Die Lösung soll das Programm und die Ausgabe für 3 verschiedene Eingaben umfassen.

Aufgabe 3 (Funktionen, Kontrollstrukturen, Array):

In dieser Aufgabe sollen Sie einen Kassenautomaten simulieren, der einen zufälligen Betrag zwischen 1 und 50 € kassiert. Dieser Automat akzeptiert alle Scheine und Münzen zwischen 1 Cent und 50 € und zeigt nach jeder Einzahlung den noch zu zahlenden Betrag an. Möglicherweise zu viel gezahlte Beträge werden in möglichst großen Scheinen und Münzen zurückgezahlt, wobei ein Schein oder eine Münze auch mehrfach ausgezahlt werden kann. Sie dürfen zur Vereinfachung davon ausgehen, dass der Automat immer genügend Wechselgeld zur Verfügung hat.

Hier ein Beispiel für einen möglichen Programmablauf:

```
Ihr Weihnachtsgeschenk kostet 17.74 Euro
Sie haben noch 17.74 Euro zu zahlen. Ihre Zahlung? 20.00
Danke!
Sie erhalten zurueck: 2.26 Euro = 1 * 2.00 Euro + 1 * 20 Cent + 1 *
5 Cent + 1 * 1 Cent

Wir danken fuer Ihr Vertrauen.
Beehren Sie uns bald wieder!
```

Hinweis 1:

Zweckmäßig strukturieren Sie Ihr Programm durch die Benutzung von drei Funktionen mit den folgenden Prototypen:

- `float kaufpreis( );` /* *Liefert Kaufpreis zwischen 1 Cent und 5000 Cent (50.00 Euro)* */
- `float zahlungen ( float p );` /* *Fordert Zahlungen an, bis deren Summe >= p ist; Rückgabewert: evt. zuviel gezahlter Betrag (Restbetrag)* */
- `void rueckgabe ( float r );` /* *Rückzahlung des zuviel bezahlten Betrages r* */

Hinweis 2:

Der Kaufpreis soll zufällig bestimmt werden. Verwenden Sie hierzu die Bibliotheksfunktion **`int rand()`** aus **`stdlib.h`**, die Zufallszahlen zwischen 0 und RAND_MAX liefert. RAND_MAX ist eine Konstante, die bei den meisten C-Systemen den Wert 32767 hat.

Beispiele:

```
rand() % 100        /* liefert eine Zufallszahl zwischen 0 und 99 */
1 + rand() % 100    /* liefert eine Zufallszahl zwischen 1 und 100 */
```

Die Zufallszahl für den Kaufpreis soll zwischen 1 und 5000 Cent liegen (**`int`**) und kann durch entsprechende Division in Euro (**`float`**) umgerechnet werden.

Um bei verschiedenen Programmausführungen unterschiedliche Zufallszahlen zu erzeugen, muß der Zufallszahlengenerator allerdings noch initialisiert werden. Dies kann durch den Aufruf

```
srand( time(0) ); /* Initialisierung des Zufallszahlengenerators */
```

am Anfang des Hauptprogramms erfolgen, bei dem über die Funktion **`time(0)`** die Zeit seit dem 1.1.1970 zur Initialisierung verwendet wird.

Hinweis 3:

Arbeiten Sie in der Funktion `rueckgabe` mit einem **`int`**-Feld **`muenzen`**, in dem die zur Zahlung zulässigen Geldbeträge in Cent (5000, 2000, 1000, …, 1) sortiert aufgeführt sind. Durchlaufen Sie bei der Rückzahlung dieses Feld von den größten bis zu den kleinsten Beträgen in Form einer Schleife und prüfen Sie jeweils, welcher möglichst große Geldbetrag und wie viele Einheiten (Scheine bzw. Münzen) davon als nächstes zurückgegeben werden muss. Vermindern Sie dann den zurückzugebenden Restbetrag um den auszuzahlenden Betrag. Da das Feld der zulässigen Geldbeträge aus Rundungsgründen vom Typ **`int`** sein sollte, der rückzuzahlende Betrag aber vom Typ **`float`**, müssen Sie jeweils Euro- (**`float`**) in Cent-(**`int`**)-Beträge umrechnen und umgekehrt.

Wie immer, soll die Lösung das Programm und die Programmausgabe umfassen.

Aufgabe 4 (Felder, Funktionen, Zeiger):

a) Lösen Sie die Aufgabe 1b dieses Blattes (siehe oben) jetzt mit Feldern (Programm und Programmausgabe). Deklarieren Sie hierzu zwei Felder:

   i. Ein Feld aus **int**-Werten, das als Elemente die Anzahl der einzelnen Vokale hat,

   ii. ein Feld aus **char**-Werten für die einzugebende Zeichenkette, z.B.

```
#define MAX_BUCHSTABEN 80 /* Maximalanzahl von Buchstaben plus '\n' plus '\0' */
char zeile[MAX_BUCHSTABEN];
```

   und lesen Sie die Textzeile z.B. über die Bibliotheksfunktion

```
char* fgets(char* buffer, int n, FILE *stream)
```

   in die Zeichenkette ein. **fgets** liest maximal **n-1** Zeichen aus **stream** in den Puffer **buffer** ein. Dabei ist für **char* buffer** die Zeichenkette einzusetzen (also **zeile**), und für **FILE *stream** können Sie die Standardeingabe **stdin** setzen. Anschließend können Sie die einzelnen Zeichen der Zeichenkette z.B. über eine **while**-Schleife bis zum ersten auftretenden '\n' durchlaufen.

   Hinweis 1: Falls weniger als **MAX_BUCHSTABEN** Zeichen eingegeben werden, sind die restlichen Felder der Zeichenkette uninitialisiert. Falls Ihre Schleife also die gesamte Zeichenkette von 0 bis **MAX_BUCHSTABEN – 1** durchläuft, können die Ergebnisse verfälscht werden. Daher darf die Zeichenkette entweder nur bis zum ersten auftretenden '\n' ausgewertet werden, oder die gesamte Zeichenkette muß geeignet initialisiert werden!

   Hinweis 2: Ein einzelnes Wort kann auch über **scanf** mit dem Formatspezifizierer **%s** in eine Zeichenkette eingelesen werden. Beachten Sie dabei, daß die Zeichenkette **zeile** bereits ein Zeiger (auf das erste Element der Zeichenkette) ist, also KEIN Adreßoperator in **scanf** mehr verwendet werden muß:

```
scanf("%s", zeile);
```

   Beachten Sie weiterhin, daß **scanf** nur bis zum ersten auftretenden „white space“ liest (z.B. einem Leerzeichen), also nur das erste Wort erfaßt. Wenn Sie *mehrere* Wörter (d.h. eine ganze Zeile) einlesen wollen, sollten Sie daher mit **fgets** arbeiten.

b) Schreiben Sie ein Programm, das Mittelwert und Standardabweichung von **numElements** (z.B. 10) **float**-Werten berechnet. Legen Sie hierzu ein Feld von **numElements** **float**-Werten an, die Sie in der Funktion
   void initialisiereFeld(float* feld, const int numElements);
   auf Zufallswerte initialisieren (siehe hierzu auch den Hinweis 2 von Aufgabe 3).
   Schreiben Sie dann eine Funktion **berechneStatistik,** der Sie das Feld übergeben; die Funktion berechnet dann Mittelwert und Standardabweichung der Feldelemente. Da *zwei* Werte berechnet werden sollen, können Sie das Ergebnis nicht über den

Rückgabewert an die **`main`**-Funktion zurückgeben; vielmehr müssen Sie Übergabeparameter nutzen (überlegen Sie hierzu, ob Sie Zeiger benötigen). Geben Sie anschließend in der Hauptfunktion die Feldelemente sowie Mittelwert und Standardabweichung aus.

Hinweis: Für die Standardabweichung können Sie die Formel

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2}$$

nutzen, wobei x_i die Feldelemente, N die Anzahl der Feldelemente und $\bar{x}$ den Mittelwert bezeichnet (d.h. Sie können den Koeffizienten $1/N$ statt $1/(N-1)$ verwenden). Für eine effiziente Implementierung sollten Sie diese Formel geeignet umstellen, so daß zur Berechnung von Mittelwert und Standardabweichung nur *eine* Schleife über die Feldelemente erforderlich ist.

Wie immer, soll die Lösung das Programm und die Programmausgabe umfassen.