Technische Universität Berlin **Neuronale Informationsverarbeitung**

C/C++ Übungsblatt 4 (Block 1)

Prof. Dr. Klaus Obermayer und Mitarbeiter

Arrays und Pointerarithmetik

Verfügbar ab:	**30.11.2020**
Abgabe bis:	**07.12.2020**

Aufgabe 1: Arithmetischer und geometrischer Mittelwert — 2 Punkte

Schreiben Sie ein C-Programm, welches ein `int`-Array von $N = 100$ Zufallszahlen zwischen 1 und 100[1] erzeugt. Berechnen Sie sich anschließend den geometrischen Mittelwert und den arithmetischen Mittelwert. Geben Sie diese auf der Konsole aus.
Der arithmetische Mittelwert μ wird berechnet mit

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Der geometrische Mittelwert $\overline{x}_{geom}$ wird berechnet mit

$$\overline{x}_{geom} = \sqrt[n]{\prod_{i=1}^{N} x_i}$$

Eine Vorgabe ist unter den Hinweisen zu finden und steht auf ISIS zum Download bereit.

Hinweis 1: Zum Erzeugen einer Zufallszahl wird die Funktion `int rand()` aus der Standardbibliothek `<stdlib.h>` verwendet, welche eine positive Zufallszahl vom Typ `int` im Intervall $[0, RAND\_MAX]$ erzeugt. Hierbei ist in der Regel $RAND\_MAX = 32767$. Diese muss aber von Ihnen noch so angepasst werden, dass sie zwischen 1 und 100 liegt. Erinnern Sie sich noch an den Modulo-Operator?

Hinweis 2: Da Computer keine richtigen, sondern nur sogenannte Pseudozufallszahlen erzeugen können, ist ein Aufruf der Funktion `void srand(int seed)`, ebenfalls aus der Standardbibliothek, nötig, um eine Startposition festzulegen. Dies ist in der Vorgabe bereits unter der Verwendung der aktuellen Zeit mit `time(0)` erfolgt.

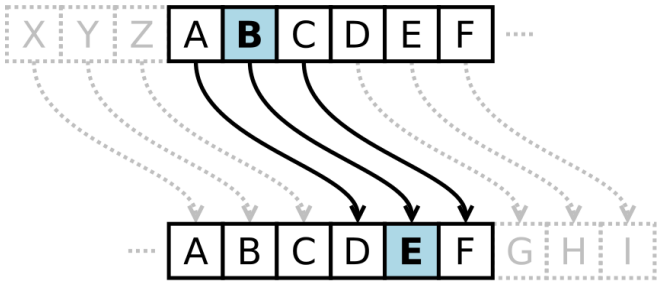
Hinweis 3: Die Standardbibliothek `<math.h>` bietet die Funktion `double pow(double, double)` zur Berechnung einer Potenz. Sie dürfen diese ausnahmsweise zur Bearbeitung dieser Aufgabe benutzen. Der erste Parameter ist für die Basis, der zweite Parameter für den Exponent.
Hinweis zum dritten Hinweis: Denken Sie daran, dass die Wurzel der Potenz des Radikanden mit dem Reziproke des Wurzelexponenten entspricht.

[1] Die Grenzen inklusive, also 1 und 100 sollen mögliche Werte sein.

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

/*
 * Dieses Programm erzeugt und initialisiert einen int-Array zufaellig.
 * Anschliessend werden geometrischer Mittelwert und arithmetischer
 * Mittelwert berechnet und auf der Konsole ausgegeben.
 */
int main(void)
{
    // Initialisiere Zufallszahlengenerator
    srand(time(0));

    // Deklariere und Intialisiere Variablen
    double arithm_mitelwert = 0.0;
    double geom_mittelwert = 1.0;

    // Array fuer Zufallszahlen anlegen
    //  ... Code hier einfuegen ...

    // Zufallszahlen erzeugen
    //  ... Code hier einfuegen ...

    // Arithmetischen Mittelwert berechnen
    //  ... Code hier einfuegen ...

    // Geometrischen Mittelwert berechnen
    //  ... Code hier einfuegen ...

    // Ausgabe
    printf("Die arithmetische Mittelwert ist: %g\n", arithm_mittelwert);
    printf("Der geometrische Mittelwert ist: %g\n", geom_mittelwert);
}
```

Aufgabe 2: Caesar-Chiffre **3 Punkte**

Nach Angaben des römischen Schriftstellers Sueton „verschlüsselte“ Julius Caesar seine militärische Korrespondenz mittels einer nach ihm benannten Methode: Jedes Zeichen aus dem Klartext wird um eine konstante Anzahl von Zeichen im Alphabet verschoben. Das Chiffrieren mit einer Verschiebung von 3 Zeichen bedeutet für Zeichen aus dem ASCII-Code also:



Schreiben Sie zuerst eine Funktion `char schiebZeichen(char zeichen, int shift)`, welche das Zeichen `zeichen` um `shift` Zeichen im Alphabet verschiebt. Der Einfachheit halber nehmen wir an, dass wir nur große und kleine lateinische Buchstaben verschieben.
Hinweis An der oberen Grenze soll umgebrochen werden, also *y* geshiftet um 3 soll *b* sein, *Y* geshiftet um 3 soll *B* sein.
Zeichen, welche nicht zwischen `'a'` und `'z'` bzw. `'A'` und `'Z'` liegen (zum Beispiel `' '` oder `'!'`), sollen nicht verändert werden.

Schreiben Sie weiterhin eine Funktion `void cipher(char str[], int shift)`, welche alle Zeichen des Strings `str` mit dem Caesar-Ciffre chiffriert, in dem sie auf jedes Zeichen in `str` die `char shiftChar(char, int)`-Funktion anwendet.
Die `void cipher(char str[], int shift)`-Funktion wird in der *main*-Funktion der Vorgabe zum chiffrieren aufgerufen. Zum dechiffrieren dient die gleiche Funktion mit negativem Verschiebungsparameter.
Hinweis: Gehen Sie davon aus, dass `shift` immer im Bereich von -25 bis 25 liegt.

Als Vorgabe dient der folgende Quellcode (welcher auch auf ISIS bereit steht):

Listing 1: caesar.c

```c
#include <stdio.h>
#include <stdlib.h>

/*
 * Bekommt ein beliebiges Zeichen zeichen und einen Wert shift uebergeben.
 * Shiftet alle Zeichen aus dem Bereich a-z und A-Z um den Wert Shift.
 * An den Wertebereichsgrenzen findet ein Umlauf statt (Beispielsweise:
 * nach Z folgt A, vor A liegt Z, nach z folgt a und vor a liegt z).
 * Zahlen ausserhalb des Bereichs werden unveraendert zurueckgegeben.
 * Gibt das kalkulierte Zeichen zurueck.
 */
char shiftChar(char zeichen, int shift)
{
    // ... hier Code einfuegen ...
}

/*
 * Bekommt einen beliebigen C-String uebergeben.
 * Fuehrt auf jedem Zeichen des Strings die shiftChar-Funktion aus.
 * Der uebergebene originale String wird dabei veraendert.
 */
void cipher(char str[], int shift, int maxlength)
```

```
{
    // ... hier Code einfuegen ...
}

/*
 * Testprogramm, das Strings mit dem Caesar-Chiffre chiffrieren kann.
 * Es benutzt dazu die cipher-Funktion.
 */
int main(void)
{
    char str[25] = "Das ist der Originaltext";  // Originaltext
    int shift = 5;
    printf("Original: ");
    printf("%s\n", str);

    // Verschluesseln
    cipher(str, shift);
    printf("Verschluesselt: ");
    printf("%s\n", str);

    // Entschluesseln
    cipher(str, -shift);
    printf("Entschluesselt: ");
    printf("%s\n", str);
}
```

Bei richtiger Implementation sollte der folgende Text auf der Konsole ausgegeben werden:

```
Original: Das ist der Originaltext
Verschluesselt: Ifx nxy ijw Twnlnsfqyjcy
Entschluesselt: Das ist der Originaltext
```

Aufgabe 3: Finde Substring

3 Punkte

Vervollständigen Sie das untere Programm so, dass es in einer Zeichenkette eine zweite Zeichenkette sucht. Die Rückgabe ist der erste Index von dem an die beiden Zeichenketten übereinstimmen, z.B: `finde("InfTechHA4", "ch")` liefert 5 zurück. Falls der zweite String nicht gefunden wurde, soll -1 zurückgeliefert werden.
Hinweis: Ihr müsst zwei Schleifen ineinander schachteln: Die äußere untersucht jede mögliche Position von `text[]`, die innere vergleicht die Zeichenketten untereinander. Hinweis: Verwende zur Lösung keine zusätzlichen Funktionen zur Arbeit mit Strings, die dir möglicherweise aus der Standardbibliothek bekannt sind.

```
#include <stdio.h>

int finde(char text[], char zuFinden[])
{
    //Hier Code einfügen
}

int main()
{
    char text[] = "DieserTextistsehrsehrlang";
    char zuFinden[] = "ist";
    int index = finde(text, zuFinden);
    printf("%s beginnt bei Index %d\n", zuFinden, index);
```

```
14 }
```

Aufgabe 4: Pointerarithmetik — 2 Punkte

Gegeben sei das folgende Programm:

```
1  #include <stdio.h>
2
3  int main(void){
4      char daten[1024];
5      char *a = daten;
6      int *b = (int*) daten;
7      double *c = (double*) daten;
8      printf("%p %p %p\n", a++, b++, c++);
9      printf("%p %p %p\n", a, b, c);
10     a += 2;
11     b += 10;
12     c += 3;
13     printf("%p %p %p\n", ++a, ++b, ++c);
14     printf("%d\n", (int)(a - daten));
15 }
```

Angenommen, `daten` liegt an Speicheradresse 8000. Was gibt das Programm aus?

Hinweis 1: Das Programm soll nicht kompiliert werden. Überlegt stattdessen was das Programm ausgeben würde wenn `daten` an der Speicheraddresse 8000 läge.

Hinweis 2: Schauen Sie sich noch einmal an, was Rückgabe und was Nebeneffekt des Inkrementoperators `++` bei der Prefix- und bei der Postfixvariante sind (Tutoriumsblatt 3 Abschnitt 3.3).

Hinweis 3: Gehen Sie bei der Bearbeitung davon aus, dass ein `char` 1 Byte, ein `int` 4 Byte und ein `double` 8 Byte groß ist.