Technische Universität Berlin **Neuronale Informationsverarbeitung**

C/C++ Übungsblatt 4 (Block 1)

Prof. Dr. Klaus Obermayer und Mitarbeiter

Arrays und Pointerarithmetik

Verfügbar ab:	**14.11.2022**
Abgabe bis:	**21.11.2022**

Aufgabe 1: Ausgabe von Arrays

2 Punkte

Schreiben Sie ein C-Programm, welches ein `int`-Array von $N = 20$ Zufallszahlen zwischen 1 und 100[1] erzeugt.

Schreiben Sie anschließend ein Programm, welches Ihnen das Array auf die Konsole wie folgt ausgibt:
{erste Zahl, zweite Zahl,..., 20te Zahl}.
Ein Beispiel Ausgabe des Arrays könnte wie folgt aussehen:

```
{23, 45, 78, 64, 33, 5, 32, 65, 71, 25, 53, 43, 73, 79, 95, 96, 18, 49, 20, 47}
```

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: Achten Sie darauf, die geschweiften Klammern nicht zu vergessen und dass hinter der letzten Zahl kein Komma steht! Also die Ausgabe von: {...,4,12,} wäre falsch, richtig wäre {...,4,12}.

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


int main(void) {
    srand(time(0));

    // Array fuer Zufallszahlen anlegen

    // Zufallszahlen erzeugen
```

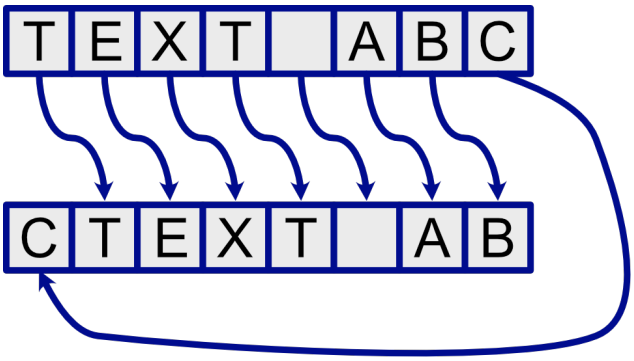
[1] Die Grenzen inklusive, also 1 und 100 sollen mögliche Werte sein.

```
12
13      // Ausgabe des Arrays
14  }
```

Aufgabe 2: Shift String — 3 Punkte

Schreiben Sie zunächst eine Funktion `void shiftStringOnce(char* text)`, welches jedes Zeichen in `text` eine Position nach rechts verschiebt und das letzte Zeichen an den Anfang verschiebt.

Beispiel: Der Text `"TEXT ABC"` um eine Position nach rechts verschoben ist `"CTEXT AB"`, wie folgende Grafik verdeutlichen soll:



Hinweis: In der Quelldatei ist eine Funktion `strlen(char* text)` definiert, welche die Länge eines Strings bestimmt. Sie können diese Funktion zur Lösung der Aufgabe verwenden, müssen es aber nicht.

Danach implementieren Sie die Funktion `void shiftString(char* text, unsigned int shift)`, welche den String um `shift` Position nach rechts verschiebt.
Als Vorgabe dient der folgende Quellcode (welcher auch auf ISIS bereit steht):

Listing 1: shift.c

```
1  #include <stdio.h>
2
3
4  /**
5   * Bekommt einen beliebigen C-String uebergeben.
6   * Und gibt dessen Länge (ohne '\0') zurück
7   */
8  int textlen(char text[]) {
9      int len = 0;
10     while (text[len] != '\0')
11         ++len;
12     return len;
13 }
14
15 /**
16  * Bekommt einen beliebigen C-String in text uebergeben.
17  * Nun wird jeder Buchstabe von text um eine Position nach rechts und der letzte
      Buchstabe nach ganz vorne vorschoben.
```

```c
 */
void shiftStringOnce(char text[]) {
}


/**
 * Bekommt einen beliebigen C-String in text uebergeben und eine Zahl shift.
 * Nun wird der text um shift position nach rechts verschoben.
 */
void shiftString(char text[], unsigned int shift) {
}

/**
 * Testprogramm, das Strings um eine beliebige Position shiften kann.
 * Es benutzt dazu die shiftString-Funktion.
 */
int main(void) {
    char str[25] = "Das ist der Originaltext";    // Originaltext

    printf("Original: %s\n", str);
    shiftString(str, 5);
    printf("Verschoben um %d stellen: %s\n", 5, str);
    shiftString(str, 19);
    printf("Nochmal um um %d stellen verschoben: %s\n", 19, str);
}
```

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

```
Original: Das ist der Originaltext
Verschoben um 5 stellen: ltextDas ist der Origina
Nochmal um um 19 stellen verschoben: Das ist der Originaltext
```

Aufgabe 3: Finden von Duplikaten **3 Punkte**

Vervollständigen Sie das untere Programm so, dass es in einem Array von Integern nach Duplikaten sucht (Zahlen, die doppelt vorkommen). Die Rückgabe ist der Wert eines Duplikates, z.B. gilt für folgenden Code:

```c
    int numbers[5] = {2, 1, 3, 5, 1};
    int arrayLength = 5;
    findeDuplikat(numbers, arrayLength));
```

dass 1 zurückgeben werden soll, da 1 doppelt in dem Array vorkommt.

Falls kein Duplikat gefunden werden konnte, soll -1 zurückgeliefert werden. Sie können dafür davon ausgehen, dass -1 in dem Array nicht enthalten sein wird.

Hinweis: Sie müssen zwei Schleifen ineinander schachteln.
Hinweis: Sollte es mehrere Duplikate geben, kann ein beliebiges ausgegeben werden.

```c
#include <stdio.h>


/**
 * Bekommt ein Array von Zahlen übergeben und dessen Länge.
 * Und sucht nach einem beliebigen Duplikat in dem Array und gibt dessen Wert
   zurück.
```

```c
 * Sollte keins gefunden werden, soll -1 zurückgegeben werden.
 *
 * Sie können davon ausgehen, dass -1 nicht als Wert im Array enthalten ist.
 */
int findeDuplikat(int array[], int arrayLength) {
}

/**
 * Programm, das Duplikate in einem Array findet.
 */
int main(void) {
    int numbers[5] = {2, 1, 3, 5, 1};
    printf("Ein Duplikat ist: %d\n", findeDuplikat(numbers, 5)); // 1 ist ein Duplikat

    int numbers2[5] = {1, 2, 3, 4, 5};
    printf("%d\n", findeDuplikat(numbers2, 5)); // es gibt kein Duplikat (-1, soll ausgegeben werden)
}
```

Aufgabe 4: Pointerarithmetik — 2 Punkte

Gegeben sei das folgende Programm:

```c
#include <stdio.h>

int main(void) {
    char *daten[1024];
    char *a = daten;
    int *b = (int *) daten;
    double *c = (double *) daten;
    printf("%p %p %p\n", a++, b++, c++);
    printf("%p %p %p\n", a, b, c);
    a += 3;
    b += 9;
    c += 5;
    printf("%p %p %p\n", ++a, ++b, ++c);
    printf("%d\n", (int) (a - daten));
}
```

Angenommen, `daten` liegt an Speicheradresse 3000. 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 3000 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.