Chapitre 1 – Opérateurs et boucles for **Exercice Algo 1.4**
Operatoren und for-Schlaufen

Dans cet exercice nous voulons essayer l'utilisation de quelques opérateurs en C, utiliser une boucle for pour répéter des instructions et approfondir l'utilisation de l'environnement de développement..

In dieser Aufgabe wollen wir die Verwendung von einigen Operatoren in C ausprobieren, for-Schlaufen zur Repetition von Anweisungen verwenden und die Verwendung der Entwicklungsumgebung vertiefen.

a) Déclarez globalement un **tableau** de chaînes de caractères **constant**, initialisez le en utilisant des noms de fruits en minuscules et affichez, en utilisant une boucle for, le contenu du tableau entier sur la console
Deklarieren Sie eine **konstante Tabelle** von Zeichenketten global, initialisieren Sie diese mit kleingeschriebenen Namen von Früchten und geben Sie den Inhalt der ganzen Tabelle mittels einer for-Schlaufe auf der Konsole aus

```
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>

int main()
{
  printf("\na) Show constant table with for-loop\n");

  #define NameLen (30)
  char const cMyfruits[][NameLen] = {"orange", "apple ", "banana",
                                     "mango ", "litchi"};
  uint16_t const cNbrOfFruits = sizeof cMyfruits / sizeof cMyfruits[0];

  for (unsigned int i=0; i<cNbrOfFruits; i++) {
    printf("Fruit %u) is %s\n", i+1, cMyfruits[i]);
  }

  return 0;
}
```

b) Adaptez la fonctionnalité de l'exercice précédant (sans adapter la déclaration) tel que le premier caractère du fruits est en plus imprimé en **majuscule** ci-après
Passen Sie die Funktionalität der vorherigen Aufgabe (ohne die Deklaration anzupassen) so an, dass der erste Buchstabe der Frucht danach noch in einem **Grossbuchstaben** ausgegeben wird .

```
  for (unsigned int i=0; i<cNbrOfFruits; i++){
    // Note: Test first character and change to upper case if possible
    char firstChar = cMyfruits[i][0];
    if ((firstChar>='a' ) && (firstChar<='z'))
    {
      firstChar &= ~0x20;
    }
    printf("Fruit %u) is %s - starts with %c\n", i, cMyfruits[i], firstChar);
  }
```

c) Utilisez ici différents opérateurs **bit**.
Verwenden Sie hier verschiedene **bit**-Operatoren.

```
printf("Result of bit operator OR  is: %d\n", cVarA|cVarB);
printf("Result of bit operator AND is: %d\n", cVarA&cVarB);
printf("Result of bit operator XOR is: %d\n", cVarB^cVarA);

printf("Result of bit operator INV is: %d\n", ~cVarA);
```

d) Utilisez différents opérateurs **logiques** et de comparaison. Pour ceci vous pouvez bien utiliser l'opérateur ternaire ?
Verwenden Sie verschiedene **logische** und Vergleichs-Operatoren. Hierzu können sie gut den ternären Operator ? einsetzen:

```
expression ? ExpressionIfTrue : expressionIfFalse
```

```
printf("Result of logical operator OR  is: %s\n", (cVarA||cVarB) ? "true" : "false");
printf("Result of logical operator AND is: %s\n", (cVarA&&cVarB) ? "true" : "false");
printf("Result of logical operator NOT is: %s\n",    !true       ? "true" : "false");

printf("Result of comparator operator == is: %s\n",(cVarA==cVarB)? "true" : "false");
printf("Result of comparator operator != is: %s\n",(cVarA!=cVarB)? "true" : "false");

printf("Result of comparator operator > is: %s\n", (cVarA>cVarB) ? "true" : "false");
```

Example for Terminal output:

```
a) Show constant table with for-loop
Fruit 1) is orange
Fruit 2) is apple
Fruit 3) is banana
Fruit 4) is mango
Fruit 5) is litchi

b) Show constant table with for-loop - start with upper-case
Fruit 0) is orange - starts with O
Fruit 1) is apple  - starts with A
Fruit 2) is banana - starts with B
Fruit 3) is mango  - starts with M
Fruit 4) is litchi - starts with L

With values of cVarA=12 and cVarB=10

c) bit-operators
Result of bit operation OR  (cVarA|cVarB) is: 14
Result of bit operation AND (cVarA&cVarB) is: 8
Result of bit operation XOR (cVarB^cVarA) is: 6
Result of bit operation INV (~cVarA)      is: -13

d) logical and comparison operators
Result of logical operation (cVarA OR  cVarB)   is: true
Result of logical operation (cVarA AND cVarB)   is: true
Result of logical operation (NOT(cVarA==cVarB)) is: true
Result of comparison (cVarA==cVarB) is: false
Result of comparison (cVarA!=cVarB) is: true
Result of comparison (cVarA>cVarB)  is: true
Result of comparison (cVarA<cVarB)  is: false


Process returned 0 (0x0)   execution time : 0.099 s
```