Aufgabe 5.1 Arrays definieren und an Funktionen übergeben (Pflicht)

Erstelle ein neues Projekt in Visual Studio und tippe den folgenden Quelltext ab:

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

void schreibe_zahlen(int param[], int n)
{
    int i;
    printf("sizeof param == %2u\n", sizeof param);
    for (i = 0; i < n; ++i)
    {
        printf("%d\n", param[i]);
    }
}

int main()
{
    int lokal[] = {4, 8, 15, 16, 23, 42};
    printf("sizeof lokal == %2u\n", sizeof lokal);
    schreibe_zahlen(lokal, 6);
    system("pause");
}
```

Wie viel Speicherplatz verbrauchen die beiden Variablen `param` und `lokal` im Speicher? Führe das Programm aus und achte auf die entsprechenden Ausgaben von `printf`. Warum muss man die Größe eines Arrays offenbar explizit an Funktionen übergeben?

Welchen Wert hat `i`, wenn `param[i]` zum ersten/letzten Mal ausgewertet wird?

Was passiert, wenn Du den Vergleich `i<n` durch `i<=n` ersetzt?

Aufgabe 5.2 Kompaktere Ausgabe (Pflicht)

Die oben aufgeführte Funktion `schreibe_zahlen` schreibt jeweils eine Zahl pro Zeile. Ändere die Implementation so ab, dass auf dem Bildschirm folgendes erscheint:

```
{4, 8, 15, 16, 23, 42}
```

Die Kommata sollen nur *zwischen* den Elementen auftauchen; vor dem ersten und nach dem letzten Element steht keins! Leere Arrays sind als Klammerpaar `{}` auszugeben.

Aufgabe 5.3 Fibonacci-Folge (Pflicht)

Schreibe eine Funktion, welche die ersten n Fibonacci-Zahlen generiert. Verwendung:

```
int main()
{
    int fibs[47];
    generiere_fibonacci_folge(fibs, 47);
    schreibe_zahlen(fibs, 47);
    system("pause");
}
```

Aufgabe 5.4 In welchem Monat liegt ein bestimmter Tag? (Pflicht)

Die folgende Funktion bestimmt für einen Tag im Jahr, in welchem Monat dieser Tag liegt:

```
int in_welchem_monat(int tag_im_jahr)
{
     if (tag_im_jahr <=  31) return  1;   // Januar
     if (tag_im_jahr <=  59) return  2;   // Februar
     if (tag_im_jahr <=  90) return  3;   // März
     if (tag_im_jahr <= 120) return  4;   // April
     if (tag_im_jahr <= 151) return  5;   // Mai
     if (tag_im_jahr <= 181) return  6;   // Juni
     if (tag_im_jahr <= 212) return  7;   // Juli
     if (tag_im_jahr <= 243) return  8;   // August
     if (tag_im_jahr <= 273) return  9;   // September
     if (tag_im_jahr <= 304) return 10;   // Oktober
     if (tag_im_jahr <= 334) return 11;   // November
     if (tag_im_jahr <= 365) return 12;   // Dezember
     return 0;
}
```

Ändere die Funktion mit Hilfe eines Arrays und einer Schleife so ab, dass nur noch eine einzige Fallunterscheidung benötigt wird.

Bonusaufgabe: Füge der Funktionen einen weiteren Parameter hinzu, um Schaltjahre zu unterstützen. Beispielsweise könnte man 1 oder 0 übergeben, je nachdem, ob es sich um ein Schaltjahr handelt oder nicht. Alternativ übergibt man die Anzahl Tage im Februar.

Aufgabe 5.5 Sieb des Eratosthenes (Kür)

Schreibe eine Funktion `void schreibe_primzahlen()`, die alle Primzahlen kleiner als 1000 berechnet und auf die Konsole schreibt. Dazu eignet sich ein effizienter Algorithmus, der als „Sieb des Eratosthenes“ (siehe Wikipedia) bekannt geworden ist:

- Erzeuge ein Array `prim` mit 1000 Zellen, die (mit Ausnahme der ersten beiden Zellen) alle auf 1 initialisiert sind.
- Durchlaufe `i` ab `2` für alle Zahlen kleiner als 1000.
- Wenn `prim[i]` gleich 1 ist, ist `i` eine Primzahl. Alle Vielfachen von i (2*i, 3*i, 4*i, …) sind keine Primzahlen und müssen daher im Array mit 0 markiert werden.

Wenn der Algorithmus durchgelaufen ist, sind alle Primzahlen in `prim` mit 1 markiert. Schreibe diese Primzahlen auf die Konsole.

Aufgabe 5.6 Tic Tac Toe (Kür)

Implementiere das Spiel Tic Tac Toe für zwei menschliche Spieler. Für das Spielbrett eignet sich ein Array der Größe 9 vom Typ `char` mit `' '` oder `'X'` oder `'O'` als Inhalt.