

1 C-Strings

Kompilieren Sie folgende Applikation:

```
1 int main() {
2     char msg[10];
3     char *p;
4     char msg2[] = "Hello";
5
6     msg = "Bonjour";
7     p  = "Bonjour";
8
9     msg = p;
10    p = msg;
11 }
```

Sie werden bemerken, dass der Compiler zwei Fehlermeldungen erzeugt. Erklären Sie warum.

2 Arrays als Funktions-Parameter

Implementieren Sie die Funktion `FillArray`, welcher Sie ein 1-D Array und seine Länge als Parameter übergeben können. Die Funktion soll den Inhalt des Arrays verändern und einen Pointer zum ersten Element des veränderten Arrays zurückgeben. Folgend der Aufrufende Code und die Funktions-Signatur.

```
1 void TestFillArray() {
2     static constexpr size_t kArraySize = 10;
3     int y[kArraySize] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
4     int* a = FillArray(y, kArraySize);
5
6     std::cout << "-- TestFillArray --" << std::endl;
7     for (size_t i = 0; i < kArraySize; ++i) {
8         std::cout << a[i] << ", ";
9     }
10    std::cout << std::endl;
11    std::cout << "-----" << std::endl;
12 }
```

```
1 #include <cstddef>
2
3 int* FillArray(int* array, size_t array_size);
```

3 C-Strings und Pointerarithmetik

Schreiben Sie eine Vergleichsfunktion für zwei C-Strings. Die Funktion soll zwei Pointer auf Character-Arrays entgegennehmen und **true** zurückgeben, falls die beiden C-Strings die gleiche Länge haben. Sie dürfen diese Aufgabe nicht mit `std::string` lösen.

Nehmen Sie die nachfolgenden C-Strings, um Ihr Programm zu testen.

```
1 char string0[] ={'J', 'a', 'a', 'v', 's', 's', '\0' };
2 char string1[] ={'J', 'a', 'a', 'v', 's', '\0'};
3 char string2[] = "Jaavs";
4 char string3[] ={'J', 'a', 'a'};
```

Folgend die Signatur der Vergleichsfunktion.

```
1 bool CompareStringLength(const char* ptr0, const char* ptr1);
```