Array Characteristics

The array name is synonymous with the address of its first element. This address cannot be changed; **it is constant**. Look at this code fragment:

```
int list[5];
cout << list << endl;
```

What prints is the **address of the first element**; not the contents of the first `int`, and not the contents of the entire array. Here's one possible output. Click the link to try it.

```
0x505260
```

Since an array name **is not a variable**, you **cannot assign** to an array, nor, can you **meaningfully compare** two arrays using the built-in comparison operators.

```
int a1[] = {1, 2, 3}, a2[3];
a2 = a1;                    // 1. Illegal
a2[0] = a1[0];              // 2. Fine
a2 = {1, 2, 3};             // 3. Illegal
if (a1 == a2) . . .         // 4. Legal, but stupid
```

The arrays `a1` and `a2` are **the same type and dimension**. Given that:

1. It is **illegal to assign** `a1` to `a2`. The name `a1` is the address of the first element in the array. **It is not a variable** that can be assigned to. This is completely different from structures, where `a1` and `a2` **would** both be variables (***lvalues***) and thus **could be** assigned to.
2. It **is legal** to **assign to array elements**; `a1` **is not** a variable, but `a1[0]` **is**.
3. **You can list initialize** an array, but you **cannot list assign** to the array.
4. With structures, **comparing two variables** is a syntax error. **Comparing two array names, while legal, is very stupid**.

Since the array name is the address of the first element in the array, and since the two arrays **cannot live at the same location** in memory, the comparison must be `false`. It doesn't matter what is inside the array.