A value was moved out of a non-copy fixed-size array.

Erroneous code example:

```
struct NonCopy;

fn main() {
    let array = [NonCopy; 1];
    let _value = array[0]; // error: cannot move out of type `[NonCopy; 1]`,
                           //        a non-copy fixed-size array
}
```

The first element was moved out of the array, but this is not possible because `NonCopy` does not implement the `Copy` trait.

Consider borrowing the element instead of moving it:

```
struct NonCopy;

fn main() {
    let array = [NonCopy; 1];
    let _value = &array[0]; // Borrowing is allowed, unlike moving.
}
```

Alternatively, if your type implements `Clone` and you need to own the value, consider borrowing and then cloning:

```
#[derive(Clone)]
struct NonCopy;

fn main() {
    let array = [NonCopy; 1];
    // Now you can clone the array element.
    let _value = array[0].clone();
}
```

If you really want to move the value out, you can use a destructuring array pattern to move it:

```
struct NonCopy;

fn main() {
    let array = [NonCopy; 1];
    // Destructuring the array
    let [_value] = array;
}
```