Chapter 8: Compound Literals

Section 8.1: Definition/Initialisation of Compound Literals

A compound literal is an unnamed object which is created in the scope where is defined. The concept was first introduced in C99 standard. An example for compound literal is

Examples from C standard, C11-§6.5.2.5/9:

> ```
> int *p = (int [2]){ 2, 4 };
> ```

`p` is initialized to the address of the first element of an unnamed array of two ints.

The compound literal is an lvalue. The storage duration of the unnamed object is either static (if the literal appears at file scope) or automatic (if the literal appears at block scope), and in the latter case the object's lifetime ends when control leaves the enclosing block.

> ```
> void f(void)
> {
>     int *p;
>     /*...*/
>     p = (int [2]){ *p };
>     /*...*/
> }
> ```
>
> `p` is assigned the address of the first element of an array of two ints, the first having the value previously pointed to by `p` and the second, zero.[...]

Here `p` remains valid until the end of the block.

Compound literal with designators

(also from C11)

```
  struct point {
    unsigned x;
    unsigned y;
  };

  extern void drawline(struct point, struct point);

// used somewhere like this
drawline((struct point){.x=1, .y=1}, (struct point){.x=3, .y=4});
```

A fictive function `drawline` receives two arguments of type `struct point`. The first has coordinate values `x == 1` and `y == 1`, whereas the second has `x == 3` and `y == 4`

Compound literal without specifying array length

```
int *p = (int []){ 1, 2, 3};
```

In this case the size of the array is no specified then it will be determined by the length of the initializer.

Compound literal having length of initializer less than array size specified

```
int *p = (int [10]){1, 2, 3};
```

rest of the elements of compound literal will be initialized to `0` implicitly.

Read-only compound literal

Note that a compound literal is an lvalue and therefore it's elements can be modifiable. A *read-only* compound literal can be specified using `const` qualifier as `(const int[]){1,2}`.

Compound literal containing arbitrary expressions

Inside a function, a compound literal, as for any initialization since C99, can have arbitrary expressions.

```
void foo()
{
    int *p;
    int i = 2; j = 5;
    /*...*/
    p = (int [2]){ i+j, i*j };
    /*...*/
}
```