3.10.2 Enumerated type ranges

A range of enumeration elements can be specified automatically, via the following syntax:

Table 3-3: Enumeration element ranges

`name`	Associates the next consecutive number with name.
`name = C`	Associates the constant C with name
`name[N]`	Generates N names in the sequence: name0, name1, ..., nameN-1. Each generated name is associated the next consecutive number. Optionally, a constant can be assigned to the generated names to associate that constant to the first generated name; subsequent generated names are associated consecutive values. N must be an integral literal constant.
`name[N] = C`	
`name[N:M]`	Generates a sequence of names starting with nameN and incrementing or decrementing until reaching name nameM. Optionally, a constant can be assigned to the generated names to associate that constant to the first generated name; subsequent generated names are associated consecutive values. N and M must be integral literal constants.
`name[N:M] = C`	

For example:

```
typedef enum { add=10, sub[5], jmp[6:8] } E1;
```

This example defines the enumerated type E1, which assigns the number 10 to the enumerated label `add`. It also creates the enumerated labels `sub0`, `sub1`, `sub2`, `sub3`, and `sub4`, and assigns them the values 11..15, respectively. Finally, the example creates the enumerated labels `jmp6`, `jmp7`, and `jmp8`, and assigns them the values 16-18, respectively.

```
enum { reg[1] = 1, reg[2:4] = 10 } vr;
```

The example above declares enumerated variable `vr`, which creates the enumerated labels `reg0` and `reg1`, which are assigned the values 1 and 2, respectively. Next, it creates the enumerated labels `reg2`, `reg3`, and `reg4`, and assigns them the values 10, 11, and 12.