Here is some actual code from the original Neliac compiler for the Univac M-460 "Countess" computer (written in Neliac, of course):

1a ⟨* 1a⟩≡ 1b▷

```
DEBUG SCAN:
i = 0: standard compiling location → i; ;
j = 0: obj prog std last address → j; ;
i = i(1)j{ [i] = straight jump function ∪ [i] = return jump function:
    fault 9. ; [i](15 → 29) = 61000₈ ∩ [i](0 → 14) − bias → k ≠ 0:
    { [k] = 0 ∪ [k] = straight jump function: fault 10. ;  }; ;
l|oop exit:  }. check key sets, turn off flex, clear indices,
key[2] ≠ 0: dump name lists and stop. exit.
F|AULT 9:
start flex, carriage return upper case, 69 → lower loop limit,
72 → upper loop limit, dump a title,
n = 177₈(1)0{ undefined name location[n] = i:
    write undefined name, continue. ;  },
C|ONTINUE: write address, loop exit.
F|AULT 10:
start flex, carriage return upper case,
77 → lower loop limit, 82 → upper loop limit, dump a title,
n = 777₈(1)0{ name address[n] − bias = k: write name, go on. ;  },
k → upper dump buffer[1], dump five number,
G|O ON: write address, loop exit.
W|RITE ADDRESS:
{ 73 → lower loop limit, 76 → upper loop limit, dump a title,
i → upper dump buffer[1], dump five numbers,  }. e|xit: . .
```

And here is the very same code without the `custom-code` typesetting:

1b ⟨* 1a⟩+≡ ◁1a

```
DEBUG SCAN:
i = 0: standard compiling location -> i; ;
j = 0: obj prog std last address -> j; ;
i = i(1)j{ [i] = straight jump function | [i] = return jump function:
    fault 9. ; [i](15 -> 29) = 61000_8 & [i](0 -> 14) - bias -> k /= 0:
    { [k] = 0 | [k] = straight jump function: fault 10. ;  }; ;
l'oop exit:  }. check key sets, turn off flex, clear indices,
key[2] /= 0: dump name lists and stop. exit.
F'AULT 9:
start flex, carriage return upper case, 69 -> lower loop limit,
72 -> upper loop limit, dump a title,
n = 177_8(1)0{ undefined name location[n] = i:
    write undefined name, continue. ;  },
C'ONTINUE: write address, loop exit.
F'AULT 10:
start flex, carriage return upper case,
77 -> lower loop limit, 82 -> upper loop limit, dump a title,
n = 777_8(1)0{ name address[n] - bias = k: write name, go on. ;  },
k -> upper dump buffer[1], dump five number,
G'O ON: write address, loop exit.
W'RITE ADDRESS:
{ 73 -> lower loop limit, 76 -> upper loop limit, dump a title,
i -> upper dump buffer[1], dump five numbers,  }. e'xit: . .
```