```
// echo_struct.rs (B1)

pub type Symbol = u8;        // Values 0x00–0xFF
pub type State = u16;        // 2-byte state index

#[derive(Debug, Clone)]
pub struct SymbolPath {
    pub symbols: Vec<Symbol>,
}

pub struct EchoGraph {
    pub transitions: [[State; 16]; 256], // δ: State × SymbolGroup → State
}

impl EchoGraph {
    pub fn resolve(&self, start: State, path: &SymbolPath) -> State {
        let mut state = start;
        for sym in path.symbols.iter() {
            let idx = (sym % 16) as usize;
            state = self.transitions[state as usize][idx];
        }
        state
    }
}

pub struct Mutator {
    pub seed: [u8; 32],
```

```
    pub session_index: u32,
}

impl Mutator {
    pub fn generate_graph(&self) -> EchoGraph {
        // Placeholder: fill with pseudorandom transition graph generation later
        EchoGraph {
            transitions: [[0; 16]; 256],
        }
    }
}
```