

TAD idPedido es int
TAD Pedido struct(id : idPedido, cant : int, dir : Direccion)
Tads AlgoConQueso

obs pedidosPendientes : conj(IdPedido)
obs pedidosPreparados : conj(IdPedido)
obs detallePedido : dict(IdPedido, Pedido)
obs pizzasHechas : int

```

proc InitPizzeria () : AlgoConQueso
  requiere {true}
  asegura {res.pedidosPendientes = {} ∧ res.pedidosPreparados = {} ∧ res.detallePedido = {} ∧ res.pizzasHechas = 0}

proc recibirPedido (inout q: AlgoConQueso, in d : Direccion, in c:int)
  requiere {c > 0 ∧ q0 = q}
  asegura {q.pedidosPendientes = q0.pedidosPendiente ∪ {|q0.detallePedido| + 1}}
  asegura {q.detallePedido = SetKey(q0.detallePedido, |q0.detallePedido| + 1, (|q0.detallePedido| + 1, c, d))}
  asegura {q.pedidosPreparados = q0.pedidosPreparados ∧ q.pizzasHechas = q0.pizzasHechas}

proc terminarPedidos (inout q : AlgoConQueso, in id : idPedido)
  requiere {id ∈ q.pedidosPendientes ∧ q0 = q}
  asegura {q.pizzasHechas = q0.pizzasHechas + 1}
  ` asegura {q.pedidosPendientes = q0.pedidosPendientes - {id} ∧ q.detallePedido = q0.detallePedido}
  asegura {(∀x : ℤ) (x ∈ q0.pedidosPendientes ∧L p0.detallePedido[x].zona ≠ p0.detallePedido[id].zona) →L
(∀k : ℤ) (k ∈ q0.pedidosPendientes ∧L q.detallePedido[x].zona ≠ q.detallePedido[id].zona ↔ k ∈ q.pedidosPendientes)
∨L (∃j : ℤ) (j ∈ q0.pedidosPendientes ∧L q.detallePedido[j].zona = q.detallePedido[id].zona ∧
q.pedidosPreparados = q0.pedidosPreparados ∪ {id})}

proc maximoPedidoresPorZona (in q : AlgoConQueso, in zona : int) : conj< Direccion >
  requiere {(∃i : ℤ) (i ∈ q.detallePedido ∧L q.detallePedido[i].zona = zona)}
  asegura {(∀j : res) ((∃k : ℤ) (k ∈ q.detallePedido ∧L q.detallePedido[k].pedido.dir = j))}
  asegura {(∀i : ℤ) (i ∈ q.detallePedido ∧L q.detallePedido[i].zona = zona →L
(∀k : res) (q.detallePedidos[i] ≤L k.cantidad))}

```