5. Tournoi de Bridge

Salle avec des tables (en nombre quelconque) de capacité de 4 chacune.
Il faut maintenir toutes les tables complètes (ou vide)
Les joueurs peuvent entrer/sortir
=> On peut accepter :

- un échange
- 4 entrées
- 4 sorties

Interface

`entrer`, `sortir`
`préparer_entrée`, `préparer_sortie`

Code d'un joueur

```
boucle
        preparer_entree!_
        entrer!_
        //joue
        preparer_sortie!_
        sortir!_
finboucle
```

Approche conditions

Variables d'état

`nbdemE`, `nbdemS`
(nb dans la salle est inutile, sauf si nb de table est borné)

Code

```
*[
        préparer_entrée?_; ne++;
        []
          préparer_sortie?_; ns--;
        []
          ne >= 1 || ns >= 1 -> entrer?_; sortir?_; ne--; ns--;
        []
          ne >= 4 -> entrer?_;entrer?_;entrer?_;entrer?_;
          ne = ne - 4;
        []
          ns >= 4 -> sortir?_;sortir?_;sortir?_;sortir?_;
          ns = ns - 4
]
```

```
for {
        select {
                case <- preparer_entrer:
```

```
                ne++
            case <- preparer_sortie:
                ns++
            case <- when(ne >= 1 && ns >= 1, entrer):
                <- sortir
                ne--; ns--;
            case <- when(ne >= 4, entrer):
                <- entrer; <- entrer; <- entrer;
                ne = ne - 4;
            case <- when(ns >= 4, sortir):
                <- sortir; <- sortir; <- sortir;
                ns = ns - 4;
        }
}
```

Approche automate

état = (ne,ns)