```
Estructura principal GLOBAL del programa:
        nLotsArray: array unidimensional de enteros que simula las 80 posiciones de la acera ( 1 acera por ajuste ).
        hRoadSemaphores: array unidimensional de semaforos que simula las 80 posiciones de la carretera ( 1 carretera por ajuste ).
        aTenThousandSemaphoresArray: array unidimensional formado por 10000 semaforos utilizados para sincronizar el orden estricto de la
                                    a la carretera.

struct IPC {
    int nLotsArray[N_AJUSTES][ROAD_LENGTH];                   // Acera
    HANDLE hRoadSemaphores[N_AJUSTES][ROAD_LENGTH];           // Carretera
    HANDLE aTenThousandSemaphoresArray[N_AJUSTES][TEN_THOUSAND];
};

//-------------------------------------------------------------------------------------------------------------------------------------------

    La carretera esta representada como Handles de Semaforos.
        - Los semaforos que están a 0 que esa casilla de carretera esta siendo ocupada por un coche.
        - Los semaforos que tienen valor 1 significa que estan libres.


    La acera de aparcamiento esta representada como un array booleano
        - OCCUPIED = Posición en Acera con coche aparcado.
        - EMPTY = Posición en Acera libre.
        - RESERVED = Posición en Acera reservada

//-------------------------------------------------------------------------------------------------------------------------------------------

int main(int argc, char** argv){

    ...

    CreateIPC();

    ...

}

//-------------------------------- CREACION DE LOS ARRAYS PARA LA ACERA, EL ORDEN DE LOS COCHES Y LA CARRETERA ----------------------

int CreateIPC(void)
{
    ...

    //        CARRETERA

    for (i = 0; i < N_AJUSTES; ++i) {
        for (j = 0; j < ROAD_LENGTH; ++j) {
            hRoadSemaphores[i][j] = CreateSemaphore(NULL, 1, 1, NULL);
        }
    }

    //        ACERA

    for (i = 0; i < N_AJUSTES; ++i) {
        for (j = 0; j < ROAD_LENGTH; ++j) {
            IpcResources.nLotsArray[i][j] = EMPTY;
        }
    }


    //        ORDEN COCHES S0 = 1, S1 = 0, S2 = 0 , ... , S10000 = 0

    for (j = 0; j < N_AJUSTES; j++){

        IpcResources.aTenThousandSemaphoresArray[j][0] = CreateSemaphore(NULL, 1, 1, NULL);

        for (i = 1; i < TEN_THOUSAND; ++i) {
            IpcResources.aTenThousandSemaphoresArray[j][i] = CreateSemaphore(NULL, 0, 1, NULL);
        }
    }

    ...
}



//-------------------------------------------------------------------------------------------------------------------------------------------

int Llegada(HCoche hc) {

    ...

    Si ( Se ha encontrado sitio para aparcar ) {

        CreateThread( ... ,AparcarRoutine, Coche , ... );

    }

    ...
}

int Salida(HCoche hc) {
    ...

    CreateThread( ... ,DesaparcarRoutine, Coche , ... );
```

```
    ...
}

//-----------------------------------------------------------------------------------------------------------------------------------------------

//    ORDEN DE LOS COCHES
        > El hilo espera por el coche i - 1 ( al estar tratando con posiciones de un array empieza en el semaforo de la posicion 0 que cc
        > El coche 1 al valer su semaforo aTenThousandSemaphoresArray[0] = 1 , pasa inmediatamente
        > El resto de coches, espera a que en aparcarCommit se haga signal sobre su semaforo y puedan pasar

        DWORD WINAPI AparcarRoutine(LPVOID lpParam) {
            ...

            Wait(aTenThousandSemaphoresArray[algoritmo][(nNumeroCoche-1) % TEN_THOUSAND]);

            ...

        }

// ESQUEMA DE SINCRONIZACION

        Coche 1        Coche 2        Coche 3      ...    Coche n
        =======     =======     =======            =======

         W(S0)         W(S1)         W(S2)             W(Sn-1)


//    ORDEN DE LOS COCHES
        > El coche i-1 libera al coche i, de manera que se cumple que los coches tienen que acceder a la carretera en estricto orden;

        void AparcarCommit(HCoche hc) {

            ...

            Signal(IpcResources.aTenThousandSemaphoresArray[algoritmo][nNumeroCoche % TEN_THOUSAND];

            ...
        }

// ESQUEMA DE SINCRONIZACION

        Coche 1        Coche 2        Coche 3      ...    Coche n
        =======     =======     =======            =======

         S(S1)         S(S2)         S(S3)             S(Sn)



//-----------------------------------------------------------------------------------------------------------------------------------------------

// SINCRONIZACION DE LA CARRETERA
    > El movimiento de los coches se produce de posicion a posición en una carretera de semaforos binarios.
    > Se pueden distinguir varios datos a tener en cuenta
            - X_fin = Posición a la que se va a mover el coche.
            - X_inicio = Posición a la que se ha movido el coche.
            - X_fin + longitud del Coche - 1 = parte de atras del coche (ultima posición que ocupa por detrás).

    > Si la posicion a la que va está ocupada, espera por ella (el semaforo de esa posición vale 0), en caso contrario (el semaforo vale
    > Si está desaparcando, tiene que esperar a que esté libre todo lo que ocupa el coche en longitud.

void PermisoAvance(HCoche hc) {

    Si( Está en la carretera entre las posiciones 0-79 ){

        Si ( Está moviendose en horizontal ) {

            Wait(IpcResources.hRoadSemaphores[algoritmo][X_fin];

        }
        Si no si( Está desaparcando ) {
            para i = X_fin + longitud del Coche - 1 hasta  X_fin con decremento en 1 unidad entonces

                Wait(IpcResources.hRoadSemaphores[algoritmo][i];

            }

        }

    }

}

// SINCRONIZACION DE LA CARRETERA
    > El movimiento de los coches se produce de posicion a posición en una carretera de semaforos binarios.
    > Se pueden distinguir varios datos a tener en cuenta
            - X_fin = Posición en la que se encuentra el coche.
            - X_inicio = Posición de la que viene  el coche.
            - X_inicio + longitud del Coche - 1 = parte de atras del coche (ultima posición que ocupa por detrás).

    > Si se mueve en horizontal, libera la zona que ha ocupado la parte de atrás del coche (ultima posición que ocupa por detrás).
    > Si está desaparcando, pone la posición que ha desocupado de la acera a LIBRE (EMPTY).
    > Si está aparcando, libera la zona que ha ocupado el coche entero en longitud para que puedan seguir el resto de coches.

```

```
190 void PermisoAvanceCommit(HCoche hc) {
191
192     Si( Está en la carretera entre las posiciones 0-79 ){
193
194         Si ( Está moviendose en horizontal ) {
195
196             Signal(hRoadSemaphores[algoritmo][X_inicio + c_length - 1]);
197
198         }
199         Si no si( Está desaparcando ) {
200             para i = X_inicio + longitud del Coche - 1 hasta  X_inicio con decremento en 1 unidad entonces
201
202                 nLotsArray[algoritmo][i] = EMPTY;
203
204             }
205
206         }
207         Si no si( Está aparcando ) {
208             para i = X_inicio + longitud del Coche - 1 hasta  X_inicio con decremento en 1 unidad entonces
209
210                 Signal(hRoadSemaphores[algoritmo][i]);
211
212             }
213
214         }
215
216     }
217
218 }
```