

UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

FACULTAD DE INGENIERIA

DEPARTAMENTO DE INGENIERIA EN SISTEMAS

SISTEMAS OPERATIVOS I
IS412 – 0900

CASO DE ESTUDIO: IMPLEMENTACIÓN DE ALGORITMOS DE SEMÁFOROS BINARIOS A UNA SIMULACIÓN DE CIRCUITO DE KARTS

GRUPO #3

NOMBRE	N. DE CUENTA
Alhanis Carolina Espinal Flores	20181031451
Kevin Alejandro Ponce	20181031257

LICENCIADO: Jaime Antonio Galeas Diaz.

FECHA DE ENTREGA: Abril 21, 2022.

CONTENIDO

En la informe continuación se presenta el análisis y desarrollo del proyecto de una simulación de circuito de karts mediante la implementación de algoritmos con semáforos binarios. Que se desarrolló mediante el análisis de los diferentes modelos de estado y ciclos de interrupciones e instrucciones, contiene también los detalles de implementación del algoritmo de semáforos binarios, y el respectivo código con partes documentadas para su comprensión.

OBJETIVO GENERAL

Realizar un código que aplique los algoritmos de semáforos binarios al caso de estudio del proyecto de una simulación circuito de karts.

OBJETIVOS ESPECIFICOS

Entender el funcionamiento de los semáforos binarios, y aprender a codificar los algoritmos que los implementan.

- Alhanis Carolina Espinal Flores.

Aprender a usar los semáforos binarios para simulaciones y ver el comportamiento de nuestras variables en un entorno controlado.

- Kevin Alejandro Ponce:

1. DEFINICIÓN DE PROYECTO

Diseño de un Programa Concurrente que simule el siguiente sistema:

En las afueras de Ciudad Real, se ha construido un circuito de karts. Los karts (carros a escala) de los que se dispone, son de dos clases: de pequeña cilindrada para menores de edad, y de una cilindrada mayor para adultos. El circuito está dotado con Ni karts de cada clase y, además de ello, una sala de espera de M plazas en la que las personas esperan para entrar al circuito. Las personas (adultos o niños) que quieran conducir un kart en el circuito, lo podrán hacer siempre y cuando uno de los karts de su correspondiente clase esté disponible.

Se debe cumplir que:

- Adultos y niños no deben conducir en la pista al mismo tiempo.

- Las personas competirán con el kart un tiempo aleatorio (según el precio que paguen en la taquilla).

- Cuando finalice ese tiempo, entrarán en Áreas de Estacionamiento para dejar el kart libre.

- Nadie en el circuito debe esperar eternamente para competir.

- Si llega una persona y están ocupados todos los karts, aguardará su turno en una sala de espera, hasta que uno de los karts quede libre. Si en la sala de espera, no queda sitio, se marchará a su casa sin poder competir.

La tarea Circuito se encarga de atender los distintos tipos de peticiones que puede recibir por parte de las tareas Persona cuando una persona llega al circuito, cuando una persona está en la sala de espera, y cuando una persona termina de correr y devuelve el coche, así como de monitorizar el estado de la simulación en todo momento.

Por otro lado, las tareas Persona, tras ser creadas e inicializadas, lo primero que intentarán será acceder al circuito (nótese que normalmente sólo pasarán directamente las primeras personas que lleguen). Si no es posible, pasarán a la sala de espera que consiste en sucesivas peticiones de entrada al circuito, hasta que la tarea Circuito le dé permiso para salir de la sala y coger un kart. Cuando una Persona obtenga el permiso para correr que ocurrirá cuando en la pista estén corredores de su tipo (adulto o niño) y no hayan pasado el máximo número de personas seguidas (puesto que en ese caso el turno cambiaría) o, por otro lado, cuando el circuito este vacío y el turno no esté asignado pasará al circuito, dará unas vueltas y terminará, devolviendo el kart y marchándose.

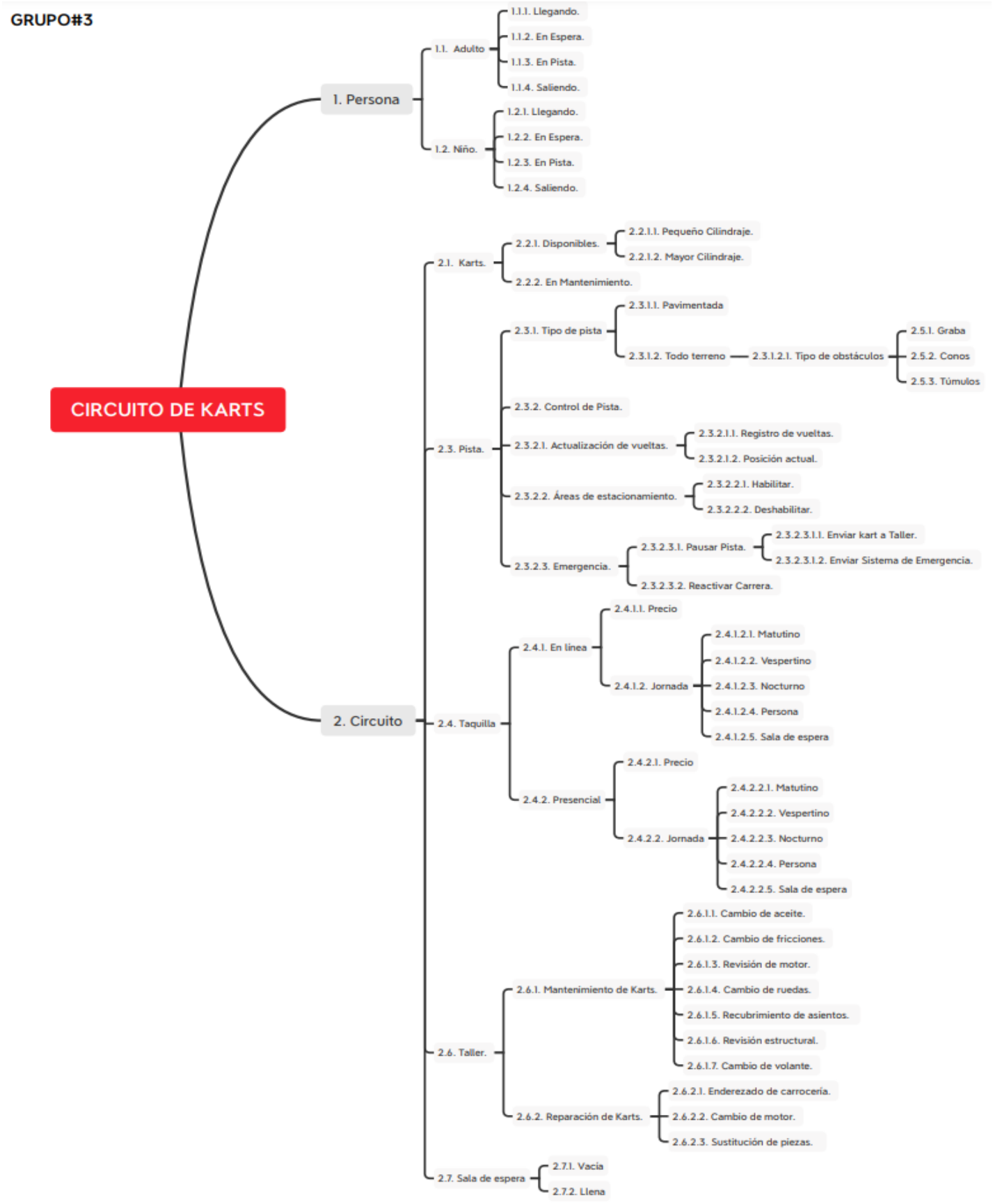
2. ANÁLISIS

El sistema de circuito de Karts consta de 2 partes/tareas/objetos por un lado se tienen las personas, y por otro el circuito.

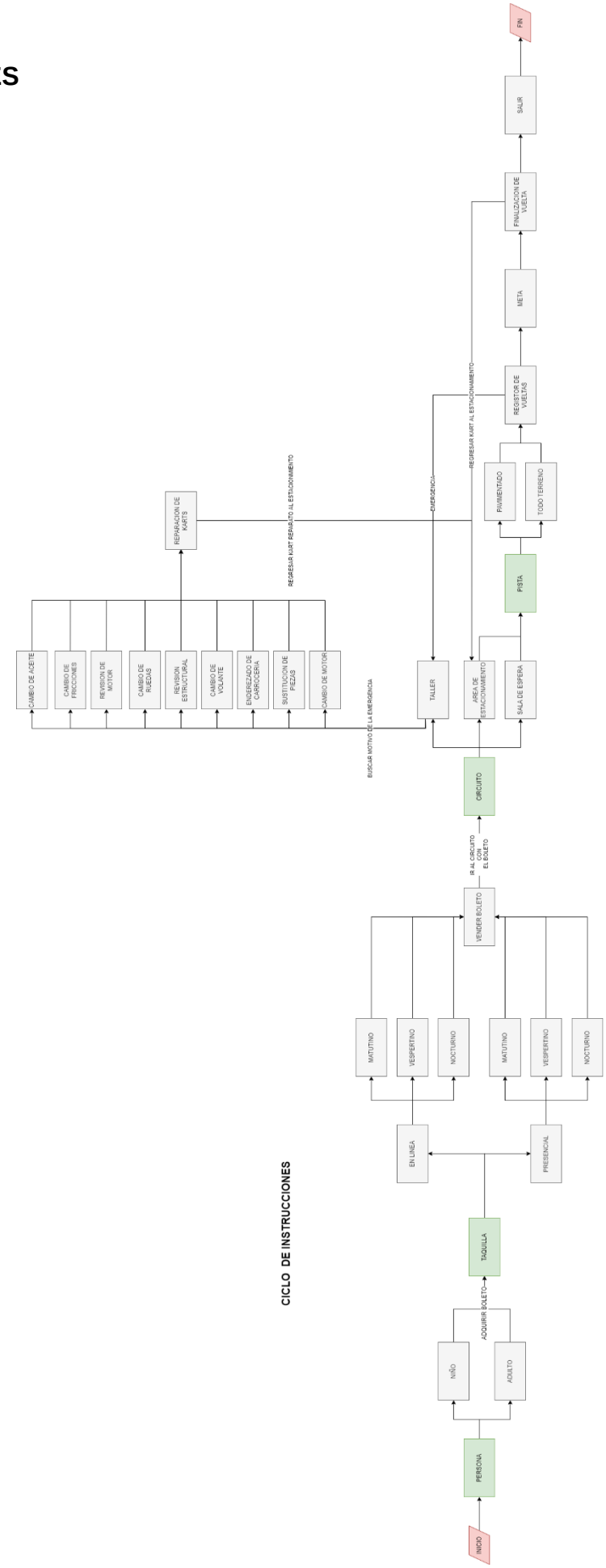
a) Persona: Representa a los corredores. Se crean tantas tareas de este tipo como personas se desee que tenga la simulación. Dentro de ellas, un campo representará el tipo de persona: adulto o niño.

b) Circuito: Representa al circuito. También se encarga de la sala de espera. Las tareas Persona interactuarán con ella durante toda la simulación.

DESGLOSE


GRUPO#3
CIRCUITO DE KARTS
1. Persona
1.1. Adulto
1.1.1. Llegando.
1.1.2. En Espera.
1.1.3. En Pista.
1.1.4. Saliendo.
1.2. Niño.
1.2.1. Llegando.
1.2.2. En Espera.
1.2.3. En Pista.
1.2.4. Saliendo.
2. Circuito
2.1. Karts.
2.2.1. Disponibles.
2.2.1.1. Pequeño Cilindraje.
2.2.1.2. Mayor Cilindraje.
2.2.2. En Mantenimiento.
2.3. Pista.
2.3.1. Tipo de pista
2.3.1.1. Pavimentada
2.3.1.2. Todo terreno
2.3.1.2.1. Tipo de obstáculos
2.5.1. Graba
2.5.2. Conos
2.5.3. Túmulos
2.3.2. Control de Pista.
2.3.2.1. Actualización de vueltas.
2.3.2.1.1. Registro de vueltas.
2.3.2.1.2. Posición actual.
2.3.2.2. Áreas de estacionamiento.
2.3.2.2.1. Habilitar.
2.3.2.2.2. Deshabilitar.
2.3.2.3. Emergencia.
2.3.2.3.1. Pausar Pista.
2.3.2.3.1.1. Enviar kart a Taller.
2.3.2.3.1.2. Enviar Sistema de Emergencia.
2.3.2.3.2. Reactivar Carrera.
2.4. Taquilla
2.4.1. En línea
2.4.1.1. Precio
2.4.1.2. Jornada
2.4.1.2.1. Matutino
2.4.1.2.2. Vespertino
2.4.1.2.3. Nocturno
2.4.1.2.4. Persona
2.4.1.2.5. Sala de espera
2.4.2. Presencial
2.4.2.1. Precio
2.4.2.2. Jornada
2.4.2.2.1. Matutino
2.4.2.2.2. Vespertino
2.4.2.2.3. Nocturno
2.4.2.2.4. Persona
2.4.2.2.5. Sala de espera
2.6. Taller.
2.6.1. Mantenimiento de Karts.
2.6.1.1. Cambio de aceite.
2.6.1.2. Cambio de fricciones.
2.6.1.3. Revisión de motor.
2.6.1.4. Cambio de ruedas.
2.6.1.5. Recubrimiento de asientos.
2.6.1.6. Revisión estructural.
2.6.1.7. Cambio de volante.
2.6.2. Reparación de Karts.
2.6.2.1. Enderezado de carrocería.
2.6.2.2. Cambio de motor.
2.6.2.3. Sustitución de piezas.
2.7. Sala de espera
2.7.1. Vacía
2.7.2. Llena

CICLO DE INSTRUCCIONES

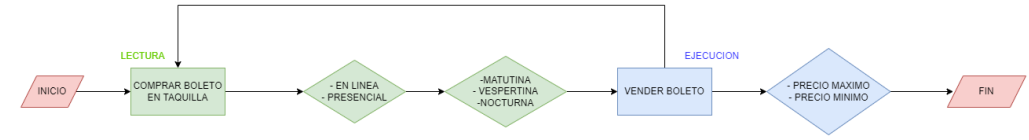


CICLO DE INSTRUCCIONES

CICLO DE INTERRUPCIONES

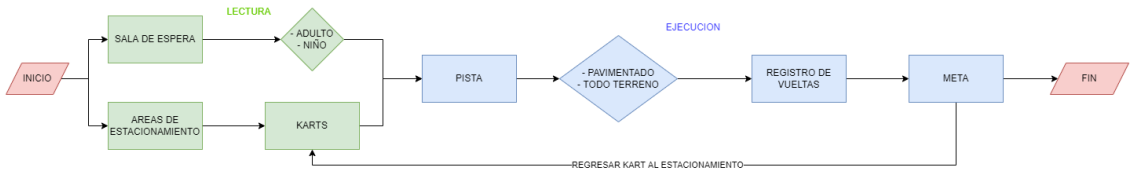
CICLO DE INTERRUPCIONES

BOLETRIA



CICLO QUE REALIZA LA BOLETERIA: 1) En la taquilla que puede ser en linea o presencial y en diferentes jornadas las personas adquieren el boleto, 2) Se vende el boleto en los tipos de precio, 3) FInaliza con esa persona y se repite para vender nuevo boleto

CIRCUITO

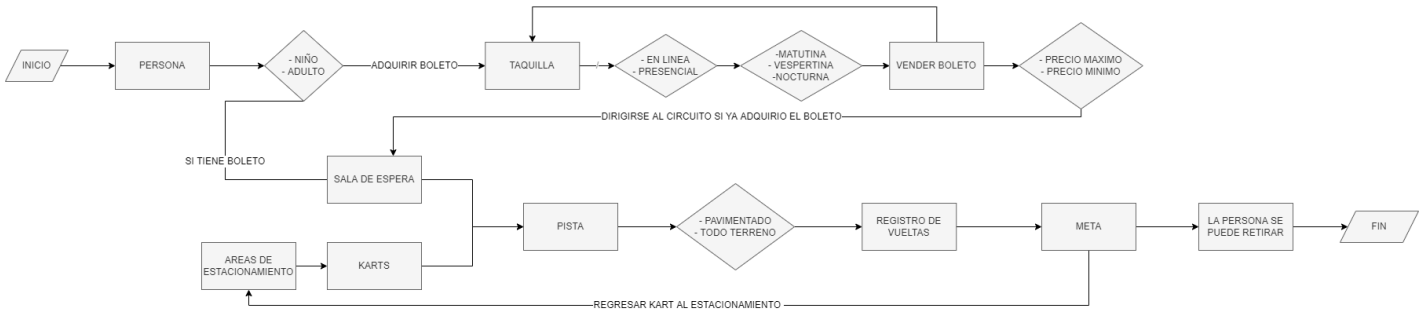


PERSONA

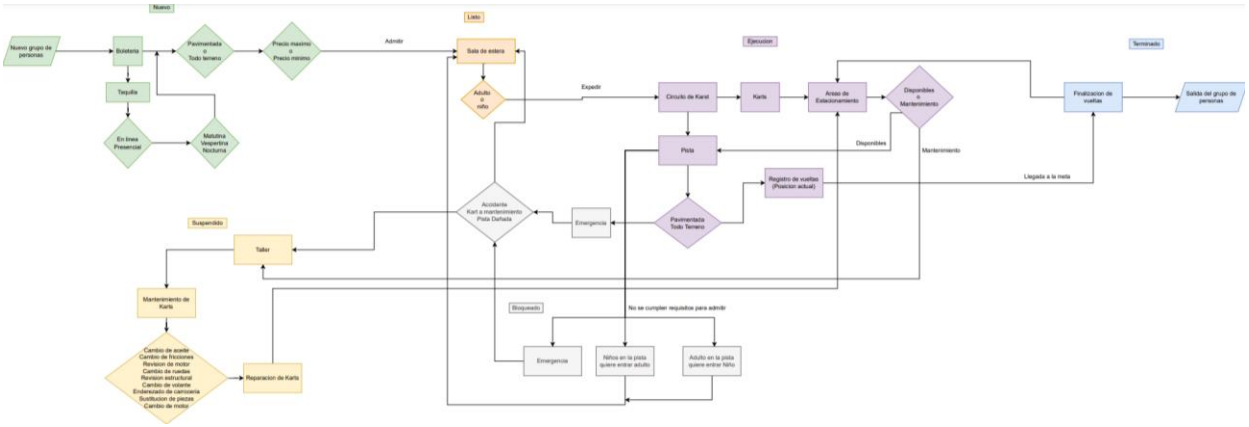
CICLO QUE SE REALIZA EN EL CIRCUITO: 1) De la sala de espera se elige los tipos de persona, del estacionamiento se eligen los karts, 2) Personas y Karts se lleva a los diferentes tipos de pistas,3) Se realiza el registro de vueltas y una vez llegan a la meta se realiza un interrupcion en el estacionamiento para llevar el kart que se ha desocupado.

CICLO QUE SIGUE UNA PERSONA: 1) Puede ser adulto o niño, 2) Adquiere boleto, 3) Se dirige a la sala de espera, 4) realiza su carrera en la pista, 5) se retira.

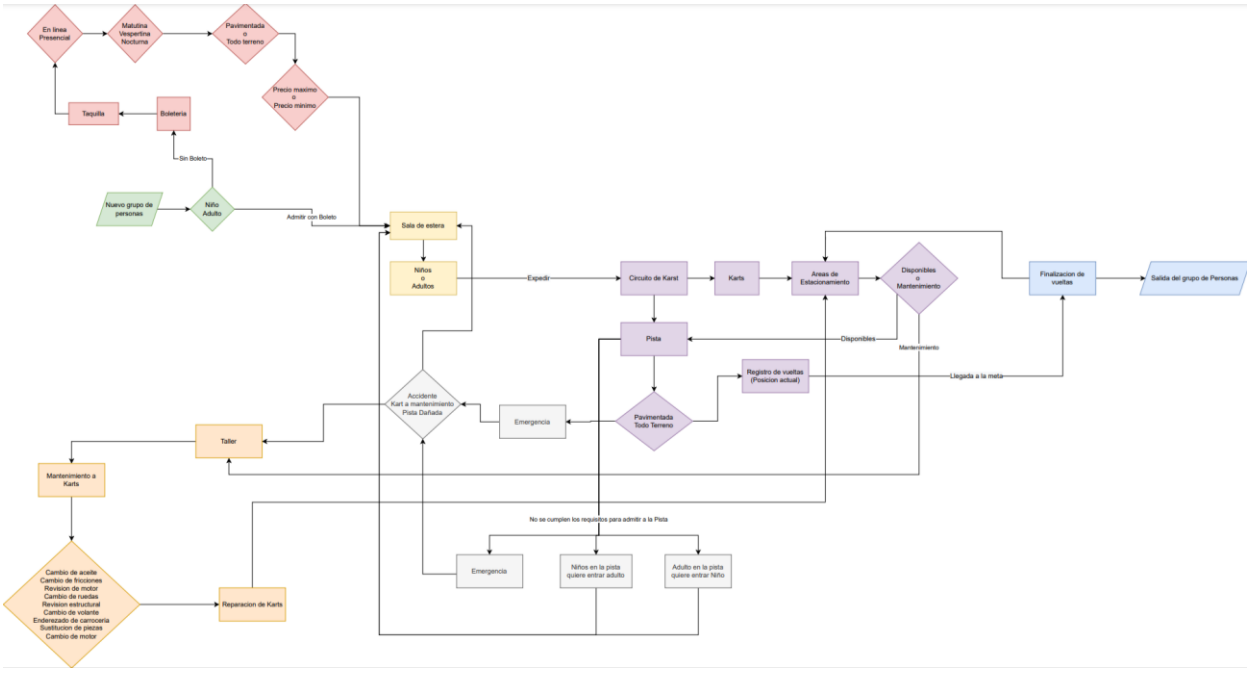
CICLO DE INTERRUPCIONES UNIDO



MODELO DE 5 ESTADOS



MODELO DE 7 ESTADOS



3. IMPLEMENTACIÓN DE SEMÁFOROS BINARIOS

Los semáforos son una herramienta de sincronización que ofrece una solución al problema de la sección crítica. Un semáforo provee una simple pero útil abstracción para controlar el acceso de múltiples procesos a un recurso común en programación paralela.

Para la implementación de semaforos binarios en la simulación del circuito de karts:

- Primero se define las variables (sem_espera, sem_corriendoKart, sem_taquilla, sem_ninos) y con valor binario que se decidirá al azar.
  Se crean los métodos para cada variable y se simulan los procesos.
- Se crea un semáforo para la Taquilla y si el sistema no pudo asignar recursos para su creación se imprime en pantalla un mensaje y se finaliza la simulación y así sucesivamente para las siguientes variables.
- Posteriormente se comienza con la iniciación de los semáforos.
- Se le define un valor para el semáforo de la carrera de los karts, si no se puede crear se imprime un mensaje de error y se finaliza la simulación y así sucesivamente con las siguientes variables.
- Da inicio a la simulación para cada grupo de persona.
- Se crea el proceso y en caso de error se retorna en pantalla un mensaje critico
- Llega una persona a la cola, compra su boleto y entra a la sala de espera y asi sucesivamente con todo el grupo de personas.
- En paralelo a este proceso entra una persona a correr en un kart, pero se verifica si entro niño (Si un adulto quiere entrar deberá esperar en la sala).
- Si entro a correr en el kart (adulto o niño) se hace la modificación en la sala de espera. corre unos segundos en el kart y se termina la carrera. Se hace la modificación en cuantos hay en la pista actualmente.
- En paralelo a este último proceso, va entrando personas a la sala de espera y saliendo a correr en el kart. Siempre se verifica que o son solo niños o adultos en la pista.
- Termina el grupo de personas y se pregunta si desea ejecutarse otra vez.

CÓDIGO

```
#include <iostream>
#include<time.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>
#include <wait.h>
#include <sys/sem.h>
#include <sys/stat.h>

using namespace std;

#define PERSONAS 1+rand()%10

union senum {
    int val;
};

// Metodos para crear e inicializar los semaforos

int crearSemEspera(){
    return semget(IPC_PRIVATE, 1, IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR);
}

int crearSemCorriendo(){
    return semget(IPC_PRIVATE,1,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR);
}

int crearSemTaquilla(){
    return semget(IPC_PRIVATE,1,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR);
}

int crearSemNinos(){
    return semget(IPC_PRIVATE,1,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR);
}

int inicializarValorSemNinos(int sem_Ninos){
    return semctl(sem_Ninos, 0, SETVAL, 1);
}

int inicializarValorSemTaquilla(int sem_taquilla){
    return semctl(sem_taquilla, 0, SETVAL, 6);
}
```

```c
int inicializarValorSemEspera(int sem_espera){
    return semctl(sem_espera, 0, SETVAL, 4);
}

int inicializarValorSemCorriendoKart(int sem_corriendoKart){
    return semctl(sem_corriendoKart, 0, SETVAL, 5);
}

//Funcion main de la ejecucion de la simulacion
int main() {

    int continuar;
    pid_t pid;
    int sem_espera, sem_corriendoKart, sem_taquilla, sem_ninos;
    int r, identificador;
    int hayNinos, esNino;

    union senum arg;
    struct sembuf arriba = {0, 1, 0};
    struct sembuf abajo = {0, -1, 0};

//-----------------------------------CREACION DE SEMAFOROS----------------------
----------------

    /**
     *Se crea un semaforo para la Taquilla y si el sistema no pudo asignar
recursos para su creacion
     *se imprime en pantalla un mensaje y se finaliza la simulacion
     */
    sem_taquilla= crearSemTaquilla();
    if (sem_espera == -1) {
        perror("Imposible crear semaforo para la entrada a la sala de espera. ");
        return -1;
    }

    /**
     *Se crea un semaforo para la sala de espera y si el sistema no pudo asignar
recursos para su creacion
     *se imprime en pantalla un mensaje y se finaliza la simulacion
     */
    sem_espera= crearSemEspera();
    if (sem_espera == -1) {
        perror("Imposible crear semaforo para la entrada a la sala de espera. ");
        return -1;
```

```
    }

    /**
     *Se crea un semaforo para la carrera de los karts y si el sistema no pudo
asignar recursos para su creacion
     *se imprime en pantalla un mensaje y se finaliza la simulacion
     */
    sem_corriendoKart= crearSemCorriendo();
    if (sem_corriendoKart == -1) {
        perror("Imposible crear semaforo para la carrera. ");
        return -1;
    }

    /**
     *Se crea un semaforo para evitar que entren adultos si hay ninos en la pista
y si el sistema no pudo asignar recursos para su creacion
     *se imprime en pantalla un mensaje y se finaliza la simulacion
     */
    sem_ninos= crearSemNinos();
    if (sem_corriendoKart == -1) {
        perror("Imposible crear semaforo para la carrera. ");
        return -1;
    }

//-----------------------------------INICIALIZACION DE SEMAFOROS----------------
----------------------

    /**
     * Se le define un valor para el semaforo de la carrera de los karts, si no
se puede crear se
     * imprime un mensaje de error y se finaliza la simulacion
     */
    r = inicializarValorSemTaquilla(sem_taquilla);

    if (r == -1) {
        perror("Error inicializando semaforo de la boleteria.");
        return -1;
    }

    /**
     * Se le define un valor para el semaforo de la sala de espera, si no se
puede crear se
     * imprime un mensaje de error y se finaliza la simulacion
     */
    r = inicializarValorSemEspera(sem_espera);
```

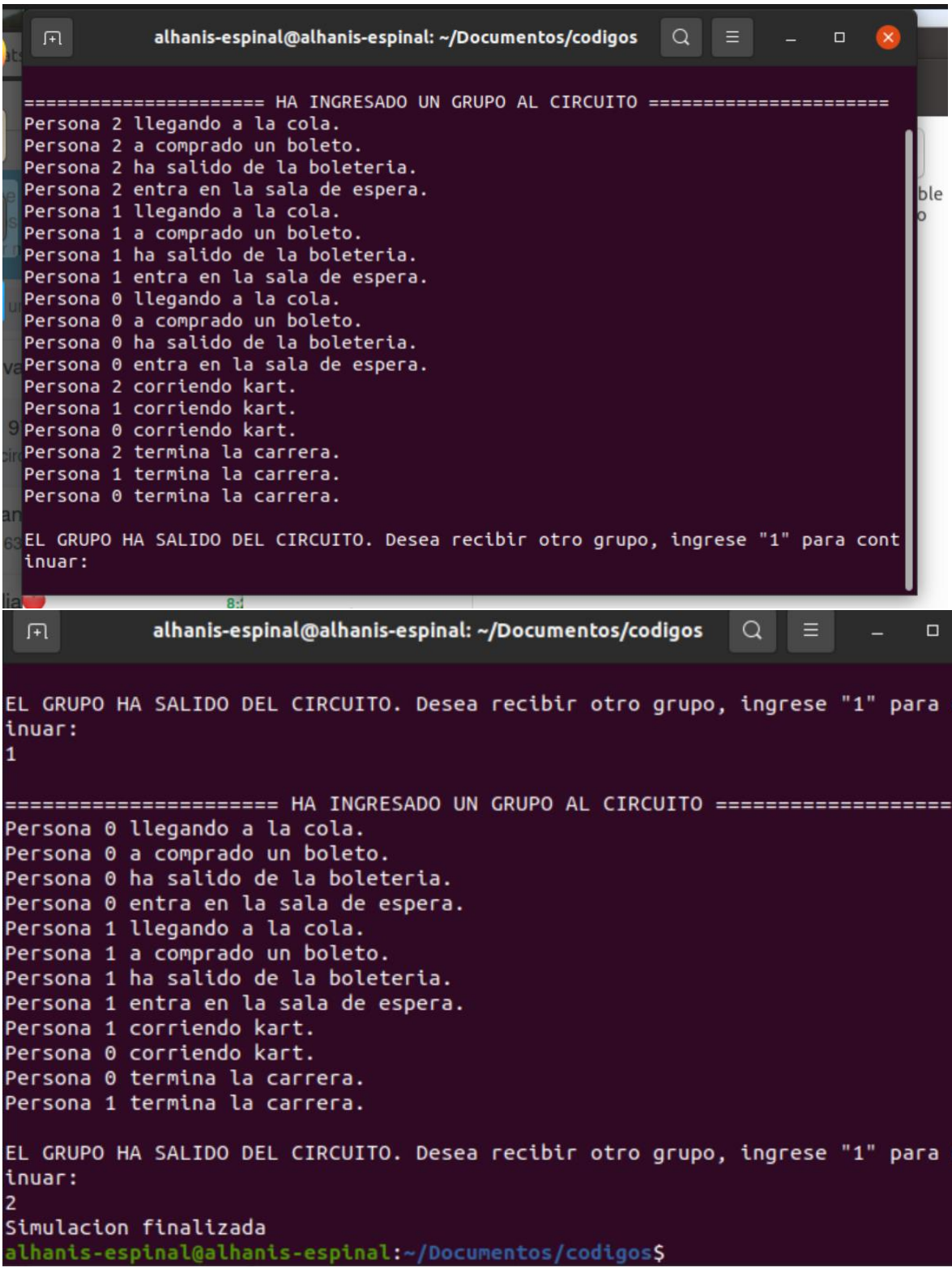
```
    if (r == -1) {
        perror("Error inicializando semaforo de la sala de espera.");
        return -1;
    }

    /**
     * Se le define un valor para el semaforo de la taquilla, si no se puede
crear se
     * imprime un mensaje de error y se finaliza la simulacion
     */
    r = inicializarValorSemCorriendoKart(sem_corriendoKart);

    if (r == -1) {
        perror("Error inicializando semaforo de la carrera.");
        return -1;
    }

    /**
     * Se le define un valor para el semaforo de ninos en la pista, si no se
puede crear se
     * imprime un mensaje de error y se finaliza la simulacion
     */
    r = inicializarValorSemNinos(sem_ninos);

    if (r == -1) {
        perror("Error inicializando semaforo de ninos en la pista.");
        return -1;
    }




//--------------------------------INICIO DE LA SIMULACION PARA CADA GRUPO DE
PERSONA----------------------------------
    printf("\n====================== HA INGRESADO UN GRUPO AL CIRCUITO
======================\n");
    for (int i = 0; i < PERSONAS; i++) {
        hayNinos = rand()%1;
        esNino = rand()%1;
        //creacion de proceso
        pid = fork();
        //En caso de error en la creacion de los subproceso
        if (pid == -1) {
            perror("Error creando procesos hijos.");
```

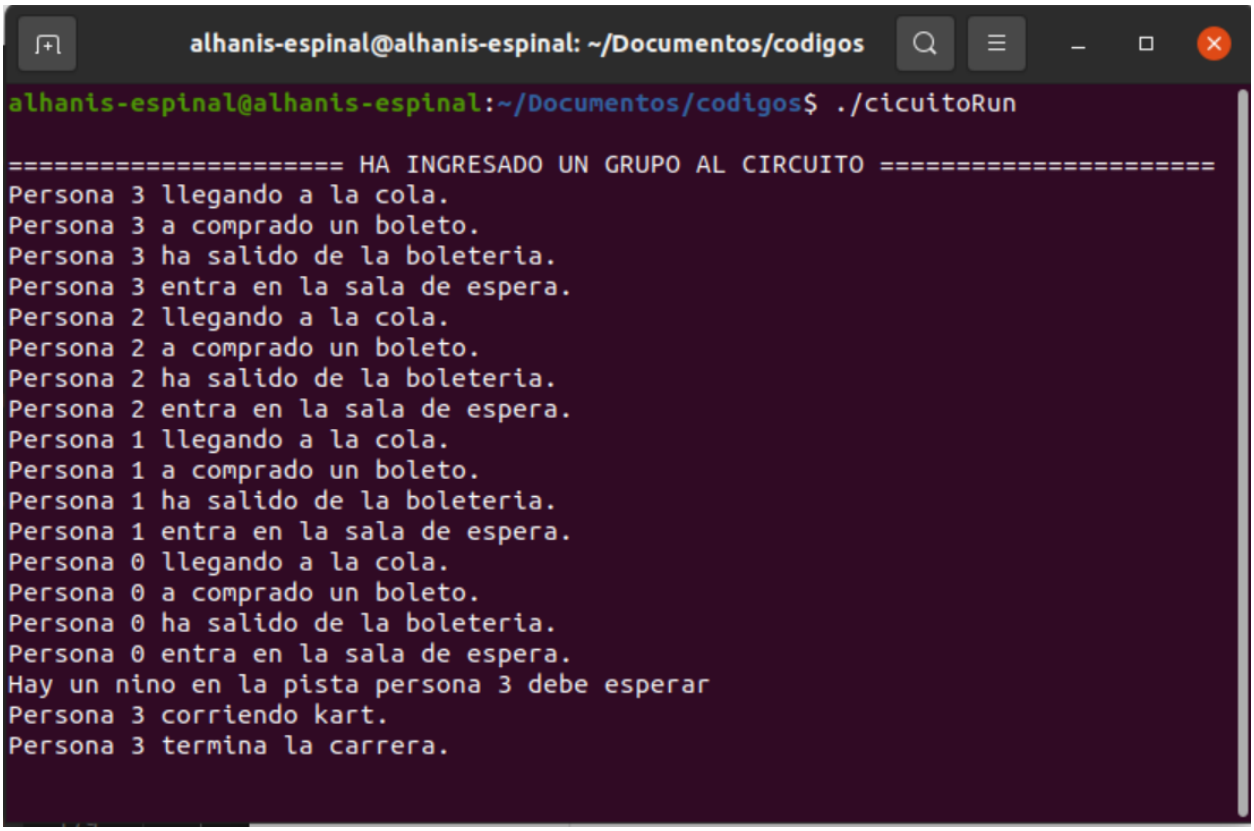
```
            return -1;
        }
        //En caso de exito en la creacion de los procesos
        if (pid == 0) {

            identificador = i;
            //Persona i llega a la cola
            printf("Persona %i llegando a la cola.\n", identificador);

            //Persona i compra boleto
            r = semop(sem_taquilla, &abajo, 1);
            if (r==-1) {
                perror("Error bajando semaforo");
            }
            //Persona i entra a la sala de espera
            printf("Persona %i a comprado un boleto.\n", identificador);

            r= semop(sem_espera, &arriba, 1);
            if (r == -1) {
                perror("Error subiendo semaforo de la sala de espera.");
            }
            printf("Persona %i ha salido de la boleteria.\n", identificador);

            //Entrar a la sala de espera
            r = semop(sem_espera, &abajo, 1);
            if (r==-1) {
                perror("Error bajando semaforo");
            }

            //Persona i entra a la sala de espera
            printf("Persona %i entra en la sala de espera.\n", identificador);
            sleep(1); //espera un rato

            //Entrar a correr un kart
            r = semop(sem_corriendoKart, &abajo, 1);
            if (r==-1) {
                perror("Error bajando semaforo");
            }
            //si entro un nino, se determina de manera aleatoria gracias a la
variable num
            if(hayNinos ==1 && esNino ==0){
                r = semop(sem_ninos, &abajo, 1);
                if (r==-1) {
                    perror("Error bajando semaforo");
                }
```

```
                printf("Hay un nino en la pista persona %i debe esperar\n",
identificador);
                sleep(2);
            }

            //Si entro a correr en el kart salir de la sala de espera
            r= semop(sem_espera, &arriba, 1);

            if (r == -1) {
                perror("Error subiendo semaforo de la sala de espera.");
            }

            printf("Persona %i corriendo kart.\n", identificador);
            //Corre unos seg el kart asi que se espera un momento
            sleep(2);
            //una vez termina de correr
            r= semop(sem_corriendoKart, &arriba, 1);

            if (r == -1) {
                perror("Error subiendo semaforo de la sala de espera.");
            }

            printf("Persona %i termina la carrera.\n", identificador);

            if(hayNinos == 2){
                r = semop(sem_ninos, &arriba, 1);
                if (r==-1) {
                    perror("Error bajando semaforo");
                }
            }


            exit(0);

        }

    }

//Gestion de los procesos
    pid = wait(NULL);

    while ( (pid != -1) ||
        ( (pid == -1) && (errno ==EINTR)))

            pid = wait(NULL);
```

```
            r = semctl(sem_taquilla, 0, IPC_RMID);

            if ( r==-1) {
                perror("Error al eliminar el semaforo de la taquilla.");
            }

            r = semctl(sem_espera, 0, IPC_RMID);

            if ( r==-1) {
                perror("Error al eliminar el semaforo de la sala de espera.");
            }

            r = semctl(sem_corriendoKart, 0, IPC_RMID);

            if ( r==-1) {
                perror("Error al eliminar el semaforo de la carrera.");
            }

            r = semctl(sem_ninos, 0, IPC_RMID);

            if ( r==-1) {
                perror("Error al eliminar el semaforo de ninos en la pista.");
            }

    printf("\nEL GRUPO HA SALIDO DEL CIRCUITO. Desea recibir otro grupo, ingrese
\"1\" para continuar:\n");
    cin >> continuar;
    if(continuar==1){
        main();
        exit(0);
    }

    printf("Simulacion finalizada\n");

    return 0;

}
```

EJECUCIÓN DE LA SIMULACIÓN




alhanis-espinal@alhanis-espinal: ~/Documentos/codigos
alhanis-espinal@alhanis-espinal:~/Documentos/codigos$ ./cicuitoRun
====================== HA INGRESADO UN GRUPO AL CIRCUITO ======================
Persona 3 llegando a la cola.
Persona 3 a comprado un boleto.
Persona 3 ha salido de la boleteria.
Persona 3 entra en la sala de espera.
Persona 2 llegando a la cola.
Persona 2 a comprado un boleto.
Persona 2 ha salido de la boleteria.
Persona 2 entra en la sala de espera.
Persona 1 llegando a la cola.
Persona 1 a comprado un boleto.
Persona 1 ha salido de la boleteria.
Persona 1 entra en la sala de espera.
Persona 0 llegando a la cola.
Persona 0 a comprado un boleto.
Persona 0 ha salido de la boleteria.
Persona 0 entra en la sala de espera.
Hay un nino en la pista persona 3 debe esperar
Persona 3 corriendo kart.
Persona 3 termina la carrera.

CONCLUSIONES

Se ha realizar un código que aplica un algoritmo de semáforos binarios al caso de estudio del proyecto de una simulación circuito de karts.

Los semáforos binarios funcionan como una herramienta que asegura la exclusión entre procesos concurrentes para acceder a un recurso compartido. Se realizo el uso de un algoritmo que los implementa y sirvió para manejar de manera sincronizada los procesos, de modo que su ejecución se realizó de forma ordenada y sin conflictos entre ellos

- Alhanis Carolina Espinal Flores.

Los semáforos binarios ayudaron a encontrar comportamientos en un grupo controlado de personas. El problema este que al ser controlado y ser binarios, cada proceso solo podemos esperar dos resultados “ocupado” o “libre”.

El semáforo binario resulta conveniente una vez que se debe defender un recurso que tienen la posibilidad de compartir diversos procesos, sin embargo, una vez que lo cual se debe defender es un grupo de recursos semejantes.

- Kevin Alejandro Ponce.