TRABAJO INFORMÁTICA

1.Descripción del programa

El programa que hemos hecho consiste en un parking que calcula el precio que han de pagar los usuarios en función del tiempo que hayan estado aparcados.
Al ejecutar el programa se despliega un menú con 3 opciones:

→Entrar al parking

→Salir del parking

→Salir del programa

Si se escoge la primera opción el programa solicitará al usuario su nombre, DNI y la matrícula del vehículo. Una vez introducidos los datos, si estos se han introducido correctamente, el programa te volverá a pedir de nuevo que escojas una opción. Podrás entrar al parking de nuevo hasta completar las plazas del parking introduciendo nuevos datos o por el contrario podrás escoger la opción de salir del parking. Si escoges la segunda opción el programa te pedirá que introduzcas la matrícula de un vehículo que quiera salir del parking y si esta se introduce correctamente, el programa te dirá el tiempo que ha estado el coche aparcado y el dinero que tienes que pagar. Una vez introduzcas el crédito se te devolverá la diferencia en caso de no ser exacto.

2.Código C

```
#define _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_DEPRECATE
#define N 3
#include<stdio.h>
#include<ctime>
#include<vector>
#include<string.h>
struct PARKING { //Declaración de la estructura de un coche cuando quiera entrar al parking
```

```
char matricula[8];
int horaentrada;
int horasalida;
char dni[10];
char nombre[10];
int minentrada;
int minsalida;
char matriculasalida[10];
};
int tiempo(int, int, int, int);
float pago(int);
int hora(int *);
void entrada(void);
void salida(void);
void cambio(float);

int hora(int *p) {
int a;
time_t now = time(0);
tm * time = localtime(&now);
a = time->tm_hour;
*p = time->tm_min;

return(a); //Función que devuelve la hora y los minutos en el presente.
}
int tiempo(int a, int b, int c, int d) {
int x, y, tiempo;
x = c - a;
y = d - b;
tiempo = (x * 60) + y;
return(tiempo); //Esta función devuelve el tiempo total en minutos que ha
estado el coche en el parking.
}
float pago(int x) {
//5 euros el minuto si esta más de 3 minutos se le cobra un 20% más.
float v;
if (x > 3) {
v = 5 * x;
v = v * 1.20;
}
else {
v = 5 * x;
}
return (v); //Esta función devuelve el valor de 'v' que corresponde con el
precio a pagar según el tiempo que haya estado en el parking
}
```

```
int main() {
int m; // m será la opción a elegir del menú del parking, si no corresponde con ninguna, te la pedirá de nuevo.
int plazas = N; //Número de plazas disponibles definidas por #define N 3 que significa que hay 3 plazas
do {
printf("Parking publico\nPlazas libres: %d\nIntroduzca 0 para entrar\nIntroduzca 1 para salir\nIntroduzca 2 para salir del programa\nOpcion:  ", plazas);
scanf("%d", &m);

if ((m == 0) && (plazas == 0)) { //Si m==0 (opción de entrar en el parking) y plazas==0 (no hay plazas) nos dirá que no hay plazas disponibles.
printf("Lo sentimos no hay plazas disponibles vuelva mas tarde\n"); //No se puede entrar en el parking
system("pause");
}

if ((m == 0) && (plazas <= N) && (plazas != 0)) { //Si le damos a la opción de entrar y plazas!=0 nos dejará entrar.
entrada(); // Llamada a la función entrada.
plazas--; //Una vez hayamos entrado, restará uno al número de plazas disponibles.

}


if ((m == 1) && (plazas < N) || (m == 1) && (plazas == 0)) { // Si m==1 (salir del parking) y hay alguna plaza ocupada, el coche sale
salida(); //Llamada a la función salida
plazas++; //Al salir el número de plazas disponibles aumentará 1.

}
if ((m != 1) && (m != 0) && (m != 2)) { //Si se introduce una opción distinta de 0,1,2 te dice que el número introducido es incorrecto y te llevará al inicio del menú.
printf("ATENCION NUMERO INCORRECTO\n");
system("pause");
}
system("cls"); //Método para limpiar la pantalla de la aplicación.

} while (m != 2); //Cuando introduzcamos la opción '2' el programa se cerrará.


}
void entrada(void) {
```

```
struct PARKING coche;
FILE *pf;
errno_t err;
int a;
int i = 0, aux = 1, numero = 0, letra = 0, v;
int t;
float p;
printf("PARKING\n\n");
printf("Nombre: ");
gets_s(coche.nombre); //Nombre del propietario del coche
gets_s(coche.nombre);
do {
printf("DNI: ");
gets_s(coche.dni);
v = strlen(coche.dni); //DNI del conductor
for (i = 0; i < v; i++) {
if ((coche.dni[i] <91) && (coche.dni[i] >64)) { //Si el símbolo esta entre la 'A' y la 'Z'
letra++; //Se añade la letra
}
if ((coche.dni[i] <58) && (coche.dni[i] >47)) { //Si el símbolo esta entre el '0' y el '9'
numero++; //Se añade el numero
}
}
if ((numero != 8) && (letra != 1)) { //Si no se introducen 8 números y una letra te pide introducir el DNI de nuevo
printf("ERROR INTRODUZCA DE NUEVO EL DNI\n");
}
} while ((numero != 8) && (letra != 1)); //Hasta que no se introduzca correctamente (8 números y una letra) no sale del bucle

v = 0, numero = 0; letra = 0;
do {
printf("Matricula: ");
gets_s(coche.matricula);
v = strlen(coche.matricula); //Matricula del conductor
for (i = 0; i < v; i++) {
if ((coche.matricula[i] <91) && (coche.matricula[i] >64)) { //Se añade la letra siempre que este comprendida entre 'A' y 'Z'
letra++;
}
if ((coche.matricula[i] <58) && (coche.matricula[i] >47)) { //Se añade el número si esta entre '0' y '9'
numero++;
}
}
```

```
if ((numero != 4) && (letra != 3)) { //Si no se introducen 4 números y 3 letras se pedira que se vuelva a introducir la matricula
printf("ERROR INTRODUZCA DE NUEVO LA MATRICULA\n");
}
else {
printf("Datos introducidos correctamente.\n");
system("pause");
}
} while ((numero != 4) && (letra != 3)); //Saldrá del bucle cuando se introduzca la matricula correctamente (4 números y 3 letras)
coche.horaentrada = hora(&coche.minentrada);
err = fopen_s(&pf, "PARKING.txt", "a"); //Abre el fichero PARKING
if (err)
printf("No se pudo abrir el archivo");
else
{
fprintf(pf, "%s %s %s %d:%d\n", coche.nombre, coche.dni, coche.matricula, coche.horaentrada, coche.minentrada); //Escribe el nombre del usuario.
fclose(pf);
}
}
void salida(void) {
FILE *pt;
struct PARKING coche;
errno_t err;
int t;
float p;
printf("Introduce tu matricula de salida: ");
gets_s(coche.matriculasalida); //Matricula de salida
gets_s(coche.matriculasalida);
err = fopen_s(&pt, "PARKING.txt", "r"); //Abre el fichero PARKING
if (err)
printf("No se pudo abrir el archivo");
else
{
while (!feof(pt)) { //Lee todo el fichero
fscanf(pt, "%s %s %s %d:%d\n", coche.nombre, coche.dni, coche.matricula, &coche.horaentrada, &coche.minentrada); //Lee los datos del fichero PARKING
if (strcmp(coche.matriculasalida, coche.matricula) == 0) { //Compara la matrícula de salida con la matrícula de entrada para calcular el tiempo que ha permanecido en el parking.
coche.horasalida = hora(&coche.minsalida);
t = tiempo(coche.horaentrada, coche.minentrada, coche.horasalida, coche.minsalida); //Calcula el tiempo que ha estado en el parking
p = pago(t); //Función que indica la cantidad a pagar.
printf("%s debe pagar %.2f euros\n\n", coche.nombre, p);
```

```
cambio(p); //Función que indica el cambio.
printf("\n\n");
printf("GRACIAS POR SU VISITA\nVUELVA PRONTO\n");
system("pause");
}
}
fclose(pt);
}
}
void cambio(float p) {
int credito, cred = 0;
float cambio;
printf("La maquina de cobro solo acepta billetes\nBilletes aceptados: 5
euros, 10 euros, 20 euros, 50 euros\n");
do {
do {
printf("Introduce credito: ");
scanf_s("%d", &credito);
if ((credito != 5) && (credito != 10) && (credito != 20) && (credito != 50)) {
//Solo se acepta crédito de 5,10,20 o 50 euros
printf("CREDITO INCORRECTO\n");
}
} while ((credito != 5) && (credito != 10) && (credito != 20) && (credito !=
50)); //Se repite el bucle hasta que no se introduzca el crédito que se acepta
cred += credito; //Se suma la cantidad total billetes
printf("\nCredito: %d\n", cred);
} while (cred < p); //No se sale del bucle hasta que el crédito introducido sea
superior al precio a pagar
if (cred > p) {
cambio = cred - p; //Cantidad de dinero a devolver
printf("Cambio: %.2f euros", cambio);
}

}
```

3.Componentes de la placa de Arduino

-Placa Arduino uno
-Led verde y led rojo
-Sensor HC-SR04
-Servo motor
-Resistencias 330 ohm
-Placa protoboard

4.Código Arduino

```
#include <Servo.h>

Servo barrera; //Se crea un objeto tipo servo (motor de la barrera)
 // Inicialización de constantes y asignación de pines al sensor y a los leds.
  int Trig = 5;
  int Echo = 6;
  int coches = 0;
 const int LEDV=11;
 const int LEDR=13;
void setup() {

 Serial.begin(9600);
  // Declaración de pines
 pinMode(LEDV,OUTPUT);
 pinMode(LEDR, OUTPUT);
 pinMode(Trig, OUTPUT);
 pinMode(Echo, INPUT);

 // Se inicializa el led rojo encendido y el verde apagado (simulando un
semáforo).
 digitalWrite(LEDV, LOW);
 digitalWrite(LEDR,HIGH);
 barrera.attach(9); // Pin de la barrera 9.
}

void loop() {

 barrera.write(0); //Se inicializa la barrera a 0 grados(bajada)
 // El sensor HC-SR04 calcula el tiempo que tarda en lanzar la señal y
recibirla y mediante una fórmula calculamos la distancia a la cual está el
objeto más cercano.
 int limite=8;
 unsigned long DISTANCIA;
 unsigned long tiempo;
 digitalWrite(Trig, LOW);
 delayMicroseconds(4);
 digitalWrite(Trig, HIGH);
 delayMicroseconds(10);
 digitalWrite(Trig, LOW);
```

```
 tiempo = pulseIn(Echo, HIGH);
 tiempo=tiempo/2;

 DISTANCIA=tiempo/29;

Serial.println(DISTANCIA); // Se imprime en el monitor la distancia en
centímetros.
Serial.print("cm");



if (DISTANCIA<limite) // Si la distancia es inferior al límite (8cm) empezara
el proceso de entrada del vehículo al PARKING.
{

if (coches<3){ //Si ya hay 3 coches en el parking, no tendrá acceso al
siguiente.

  coches=coches+1; // Suma uno cada vez que pasa un coche.
  //Se activa el led verde y apaga la roja del semáforo.
  digitalWrite(LEDR,LOW);
  digitalWrite(LEDV,HIGH);
  delay(500); // Método para esperar 500 microsegundos.
  barrera.write(90); // El motor levantará la barrera 90 grados de su posición
original.
  delay(3000);

}

else
{
  Serial.print("EL PARKING ESTA LLENO");
  }
}
else // Si la distancia no es inferior al límite el semáforo y la barrera
permanecerán en su posición original.
{
  digitalWrite(LEDR,HIGH);
  digitalWrite(LEDV,LOW);
  barrera.write(0);
}

delay(1000);
}
```

Alejandro de la Fuente Arranz 53910
Cesar Curiel Montolio 53904