Universidad de Buenos Aires

Facultad de Ingeniería

Algoritmos y Programación I (95.11)

Curso: 01-Ing. Cardozo

Trabajo Práctico n.º 1 - Visualización de mensajes GPS en formato NMEA

Alumnos:

OCHAGAVIA, Lara	100637	lari.ochagavia13@gmail.com
PINTOS, Gastón Maximiliano	99711	massipintos@gmail.com

Fecha de entrega: 22 de junio de 2019

Fecha de primer reentrega: 29 de junio de 2019

Índice

1. Introducción

En este informe se presenta el desarrollo de aplicativos de consola con comandos de línea de órdenes y escritos en lenguaje C, que permiten obtener mensajes de GPS a través de archivos en formato NMEA y su posterior impresión en un archivo de formato CSV o bien KML según sea indicado. Para la elaboración de este trabajo se requirió el manejo de tipos de datos abstractos, estructuras, memoria dinámica y archivos.

A lo largo del informe se explicitan las consideraciones que se tuvieron en cuenta para el trabajo práctico, junto con la documentación de las funciones implementadas para cumplir los objetivos del proyecto.

En los apéndices del informe se adjuntan el diagrama de flujo del programa, el esquema funcional del programa desarrollado, los códigos y el contenido de los archivos de encabezado correspondientes a cada programa.

Por último, en la conclusión del trabajo, se especifican las dificultades que se presentaron al momento de la elaboración del código, los errores que se cometieron y una descripción del resultado final del trabajo realizado.

2. Desarrollo

Para la implementación del programa, se tuvo en cuenta como estrategia de modularización el uso de diferentes archivos conteniendo el código de las funciones y un homónimo como archivo de inventario, agrupando dichas funciones por funcionalidad.

Para comenzar con la elaboración del trabajo, se implementaron en un principio las funciones básicas y necesarias para la ejecución de un programa en comando en línea de ordenes, para luego enfocarse en el desarrollos de los tipos de datos abstractos utilizados.

El archivo de entrada contiene en cada línea la información de coordenadas de localización terrestre en formato NMEA. Los campos de información están delimitados por el carácter ',' por lo que se implementaron funciones similares al manejo de archivos de formato CSV. Como condición de enunciado, se clasificaron las lineas útiles del archivo según el campo de identificación `$GPGGA`. Como segundo criterio de selección, se verificó que el campo `"suma de verificación"` se corresponda con la suma resultante de la operación *XOR* entre cada carácter entre los caracteres '$' y '*'. Para realizar ambas acciones, se utilizaron funciones que retornan valores booleanos asumiendo que los argumentos que esas funciones reciben fueron previamente validados contra puntero nulo.

Una vez seleccionada la línea, se almacenaron estos campos de información en una estructura que se rige por un tipo de dato abstracto denominado *ADT_coordinate_t*. El almacenamiento de los datos requirió la necesaria atención para respetar los formatos específicos de cada campo. Para los campos numéricos se utilizaron funciones de conversión alfanumérica sujetas a validaciones. Para campos que contienen tanto unidades de medición como de orientación geográfica se guardaron los caracteres originales de la línea. Para el manejo de fechas se utilizó la estructura *time_t* incluida en la biblioteca *"time.h"*.

La lectura del archivo se realizó de manera secuencial, por lo que una vez cargada la estructura, se la almacena en un vector dinámico de estructuras definido por otro tipo de dato abstracto llamado *ADT_vector_t*. Al finalizar la lectura de los datos, se ordenó cada estructura del vector de forma ascendente según su campo UTC, que describe el tiempo universal coordinado. La función encargada de realizar la comparación de estos campos retorna un valor booleano, por lo que se asume que los argumentos que recibe fueron previamente validados contra puntero nulo. Como estrategia de ordenamiento se utilizó método *"Bubble Sort"* considerado el más apropiado por la cantidad de elementos en el vector y el tamaño de las estructuras en él guardadas.

Al considerar la elaboración del archivo de salida, se debe tener en cuenta el argumento recibido por línea de órdenes, el cual indica el formato de impresión que se debe utilizar. Según cuál sea dicho formato, se desarrolló una función específica para cumplir con los aspectos particulares de cada uno.

Para el caso de formato CSV, se inicializaron los campos de encabezado y pie de página del vector *ADT_vector_t* como nulos para evitar su aparición en el archivo de salida. El delimitador utilizado para este formato fue el

carácter '|'. En este archivo se imprimen en cada línea los campos correspondientes a: la fecha, la latitud, la longitud y la altura sobre el nivel del mar de una estructura. Previo a la impresión del campo que contiene la fecha, se completó este campo con los datos del día, mes y año del ordenador utilizando la función *local_time* incluida en la biblioteca *time.h*.

Para el caso en el que el argumento indique que el formato de impresión sea KML, se inicializaron los campos de encabezado y pie de página basándose en el ejemplo presentado en las consignas. Para la impresión de los campos de longitud y latitud, fue necesaria la conversión del dato presentado como sexagesimal al sistema decimal. Se utilizó el carácter ',' como delimitador.

Para ambos formatos de salida, la función que se encarga de exportar el vector, determina la impresión del carácter de salto de linea al finalizar con la impresión de cada estructura.

Las funciones cuentan con validaciones internas para asegurar su correcto funcionamiento, adaptadas a las necesidades de cada una. Si un error se detecta, se informa al usuario cuál es el problema, mediante la implementación de un diccionario de errores.

Al estar todas las funciones validadas, el siguiente paso consiste en cerrar los archivos abiertos, con el detalle de validar esta acción realizada sobre el archivo de salida por haber sido este abierto en modo de escritura.

2.1. Alternativas consideradas y estrategias adoptadas

Se tuvo en cuenta el trabajo realizado para el caso de estudio n.º 2, ya que se consideraron útiles muchas de las funciones desarrolladas en su elaboración. Con el objetivo de reutilizar estas funciones, se pensó en la estrategia de resolución del presente trabajo.

En una primera instancia, se recordó un error cometido al momento de elaborar el caso de estudio n.º 2, el cual fue cargar un arreglo de punteros a estructuras a memoria de forma dinámica, lo que para este previo trabajo era incorrecto. Sin embargo, para el presente proyecto resultó adecuada la implementación de dicha estrategia, para la cual ya estaba desarrollada la función encargada de cargar a memoria las estructuras generadas por cada línea del archivo.

Con el error mencionado para el trabajo previo fue que se definió el desarrollo del presente, ya que salvando diferencias propias del manejo de tipos de dato abstracto, las acciones a llevar a cabo no difieren en grandes rasgos. Una vez cargados los datos a memoria, de procedió a la impresión de ciertos campos, manejando ciertos criterios y respetando el formato del archivo de salida según corresponda, procedimientos explicados en el desarrollo.

2.2. Corridas de prueba

A continuación se presentan capturas de pantalla tomadas a la terminal en diferentes situaciones de ejecución.

```
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ make all
make: Nothing to be done for 'all'.
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ ./make_main -fmt kml -out gpsprueba2.kml hola.txt
Unable to open file.
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$
```

Figura 1: Ejecución con un archivo de entrada inexistente

```
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ make all
make: Nothing to be done for 'all'.
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ ./make_main -fmt csh -out gpsprueba2.txt trayectoriaA.txt
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$
```

Figura 2: Ejecución con un formato no válido

El resultado de la ejecución presentada en la figura 2 es un archivo en blanco, sin nada escrito en él.

```
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ make all
make: Nothing to be done for 'all'.
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$ ./make_main -fmt kml -out gpsprueba2.kml trayectoriaB.txt
lara@Lara:~/Documents/Algoritmos/TP_GPS(new)$
```

Figura 3: Ejecución bien implementada

3. Conclusión

El manejo de tipos de datos abstractos permite definir un nuevo tipo de dato utilizando el manejo de estructuras para englobar distintos tipos de datos como elementos, junto con todas las funciones necesarias para su implementación. El manejo de este tipo de dato aporta seguridad al código al generar un encapsulamiento de la información del dato. Asimismo, al generar actualizaciones en la implementación del tipo de dato, no se altera el código fuente.

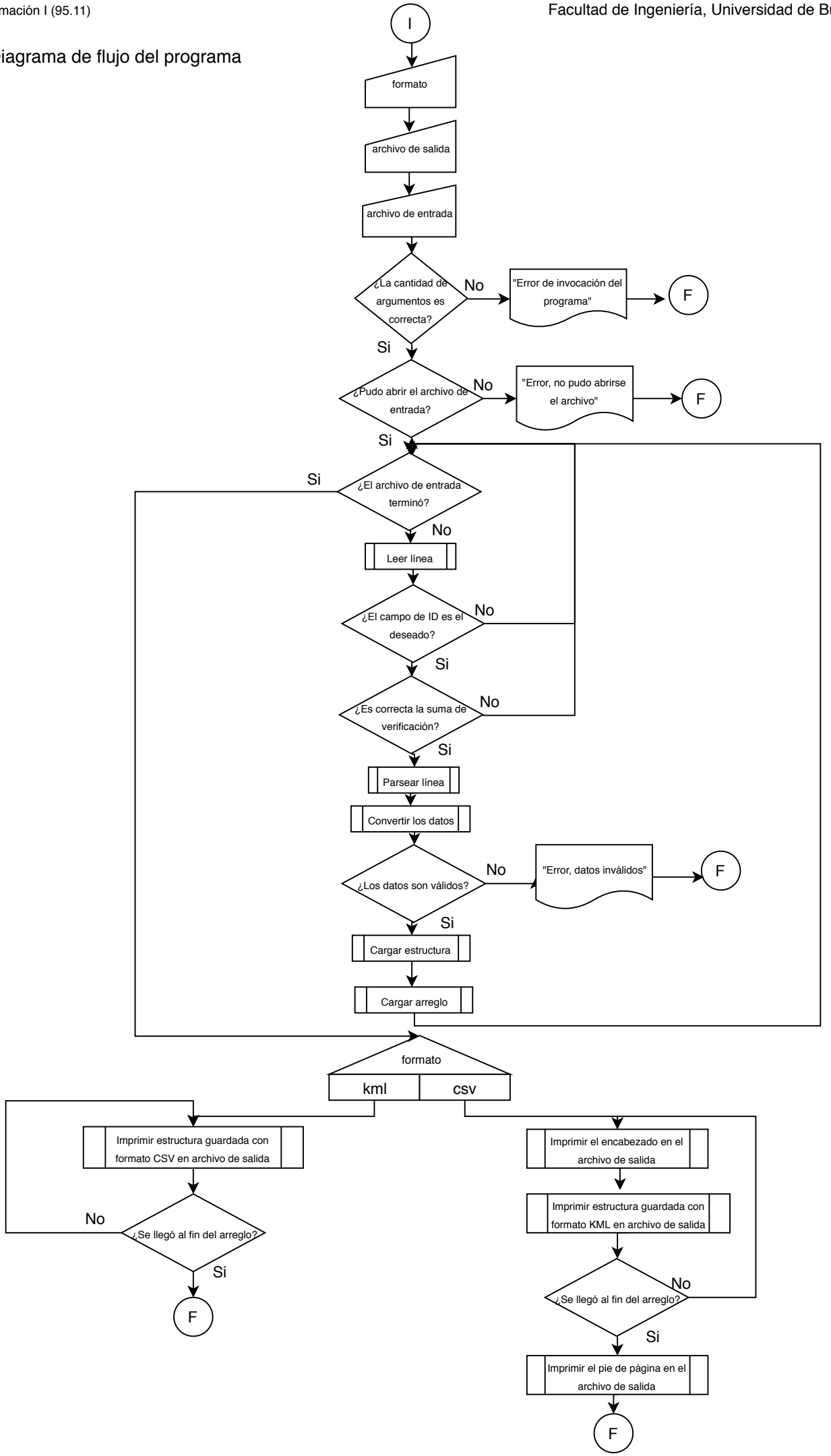
Al utilizar tipos de dato abstracto fue necesario el uso de punteros a función. Con esto se logra independizar la implementación de la función apuntada de la función que la recibe como argumento. Al quedar esta implementación oculta, se desconoce el manejo de los datos dentro de ella sin desconocer la función que cumple.

La compilación del código fuente se llevó a cabo a través de un conjunto de comandos incluidos en un archivo llamado *Makefile*. Esto supone una ventaja al unificar la compilación de los distintos archivos fuentes. Además, se logra independizar la compilación de archivos fuente modificados sin necesidad de compilar nuevamente la totalidad del código.

Las principales dificultades se presentaron con la conversión de los datos numéricos cargados en la estructura al presentar estos números decimales, el manejo de fechas al utilizar funciones y estructuras provenientes de la biblioteca *time.h*, y la modularización del código respetando el manejo de tipo de dato abstracto.

Con este trabajo se profundizó el conocimiento de punteros a función, tipo de dato abstracto y estructuras, ya que en cada función implementada se requiere suficiente abstracción para entender cómo esta función interactúa con las otras.

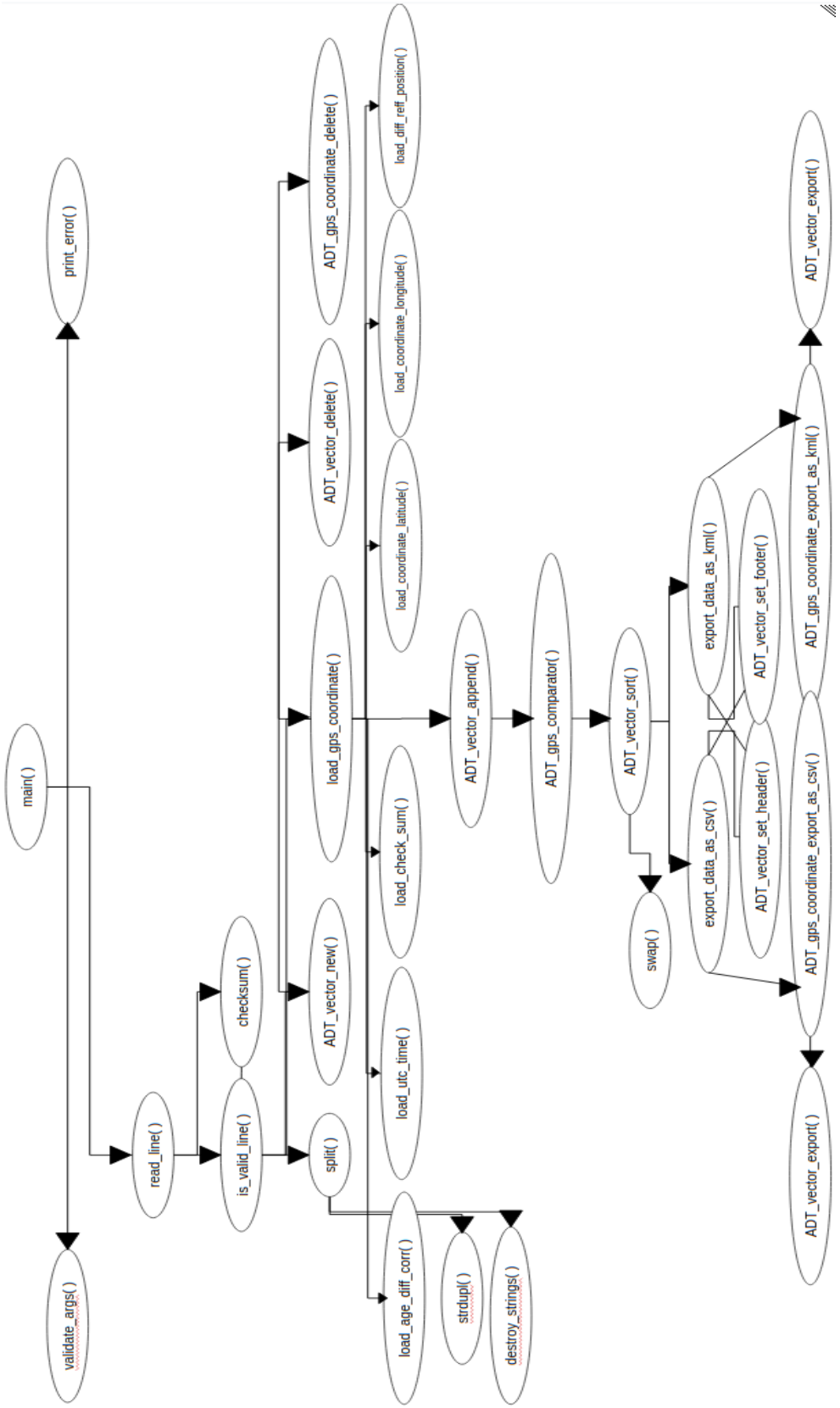
Apéndice I: Diagrama de flujo del programa



PINTOS, Gastón Maximiliano
OCHAGAVIA, Lara

4. Apéndice II: Diagrama estructural del programa

Se incluye el diagrama estructural en la siguiente carilla


main()
print_error()
validate_args()
read_line()
checksum()
is_valid_line()
split()
ADT_vector_new()
load_gps_coordinate()
ADT_gps_coordinate_delete()
ADT_vector_delete()
load_coordinate_latitude()
load_coordinate_longitude()
load_diff_refl_position()
load_check_sum()
load_utc_time()
load_age_diff_corr()
strdupl()
destroy_strings()
ADT_vector_append()
ADT_gps_comparator()
ADT_vector_sort()
swap()
export_data_as_csv()
export_data_as_kml()
ADT_vector_set_header()
ADT_vector_set_footer()
ADT_gps_coordinate_export_as_csv()
ADT_gps_coordinate_export_as_kml()
ADT_vector_export()
ADT_vector_export()

5. Apéndice III: Código

5.1. Tipos de dato abstracto - TDA

5.1.1. vector.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "vector.h"
6  #include "utilities.h"
7
8  status_t ADT_vector_new( ADT_vector_t **vector)
9  {
10         if(vector==NULL)
11                 return ERROR_NULL_POINTER;
12
13         if((*vector=(ADT_vector_t*)malloc(sizeof(ADT_vector_t)))==NULL)
14                 return ERROR_MEMORY;
15
16         if(((*vector)->array_elements=(void**)malloc(INIT_CHOP*sizeof(void*)))==NULL)
               {
17                 free(*vector);
18                 vector=NULL;
19                 return ERROR_MEMORY;
20         }
21
22         (*vector)->size=0;
23
24         (*vector)->alloc_size=INIT_CHOP;
25
26         (*vector)->header=NULL;
27         (*vector)->footer=NULL;
28
29         return OK;
30 }
31
32 status_t ADT_vector_append(ADT_vector_t **vector, void *element)
33 {
34         void **aux;
35
36         if(vector==NULL||element==NULL)
37                 return ERROR_NULL_POINTER;
38
39         if(     (*vector)->size < (*vector)->alloc_size)
40         {
41                 (*vector)->array_elements[(*vector)->size++] = element;
42                 return OK;
43         }
44
```

```
45          if((aux=(void**)realloc((*vector)->array_elements, (*vector)->alloc_size+
                INIT_CHOP*sizeof(void*)))==NULL)
46                  return ERROR_MEMORY;
47
48          (*vector)->alloc_size+=INIT_CHOP;
49
50          (*vector)->array_elements=aux;
51
52          (*vector)->array_elements[(*vector)->size++] = element;
53
54          return OK;
55  }
56
57  status_t ADT_vector_delete(ADT_vector_t **vector, destructor_gps_t pf)
58  {
59          status_t state;
60          size_t i;
61
62          if(vector==NULL || pf ==NULL)
63                  return ERROR_NULL_POINTER;
64
65          for(i=0; i< ((*vector)->size) ; i++)
66          {
67                  if((state= pf((*vector)->array_elements[i]))!=OK)
68                          return state;
69                  (*vector)->array_elements[i]=NULL;
70          }
71          free((*vector)->array_elements);
72
73
74          return OK;
75  }
76
77  status_t ADT_vector_export(const ADT_vector_t *vector, FILE *fo, void *tool_box,
        printer_gps_t pf)
78  {
79          size_t i;
80          status_t state;
81
82          if( vector==NULL || fo==NULL )
83                  return ERROR_NULL_POINTER;
84
85          if((vector->header)!=NULL)
86                  fprintf(fo, "%s\n", vector->header);
87
88          for(i=0; i < vector->size ; i++){
89                  if((state=pf(vector->array_elements[i], tool_box, fo))!=OK)
90                          return state;
91                  fprintf(fo, "\n");
92
93          }
```

```
94
95          if((vector->footer)!=NULL)
96                  fprintf(fo, "%s\n", vector->footer);
97
98          return OK;
99  }
100
101 status_t ADT_vector_set_header(ADT_vector_t *vector, char* string)
102 {
103         if( vector==NULL)
104                 return ERROR_NULL_POINTER;
105
106         if(string!=NULL)
107         {
108                 if((vector->header=(char*)malloc((strlen(string)+2)*sizeof(char)))==
                        NULL)
109                         return ERROR_MEMORY;
110
111                 vector->header=string;
112         }
113
114         return OK;
115 }
116
117 status_t ADT_vector_set_footer(ADT_vector_t *vector, char* string)
118 {
119         if( vector==NULL)
120                 return ERROR_NULL_POINTER;
121
122         if(string!=NULL)
123         {
124                 if((vector->footer=(char*)malloc((strlen(string)+2)*sizeof(char)))==
                        NULL)
125                         return ERROR_MEMORY;
126
127                 vector->footer=string;
128         }
129
130         return OK;
131 }
132
133 status_t export_data_as_csv(ADT_vector_t *vector, FILE *output_file, void* tool_box)
134 {
135         status_t status;
136
137         if(vector==NULL||output_file==NULL||tool_box==NULL)
138                 return ERROR_NULL_POINTER;
139
140         if((status=ADT_vector_set_footer(vector, NULL))!=OK)
141                 return status;
142
```

```
143         if((status=ADT_vector_set_header(vector, NULL))!=OK)
144                 return status;
145
146         if((status=ADT_vector_export(vector, output_file, tool_box,
                ADT_gps_coordinate_export_as_csv))!=OK)
147                 return status;
148
149
150         return OK;
151 }
152
153 status_t export_data_as_kml(ADT_vector_t *vector, FILE *output_file, void* tool_box)
154 {
155         status_t status;
156
157         if(vector==NULL||output_file==NULL||tool_box==NULL)
158                 return ERROR_NULL_POINTER;
159
160         if((status=ADT_vector_set_footer(vector, FMT_KML_FOOTER))!=OK)
161                 return status;
162
163         if((status=ADT_vector_set_header(vector, FMT_KML_HEADER))!=OK)
164                 return status;
165
166         if((status=ADT_vector_export(vector, output_file, tool_box,
                ADT_gps_coordinate_export_as_kml))!=OK)
167                 return status;
168
169         return OK;
170 }
171
172 status_t ADT_vector_sort(ADT_vector_t *vector, comparator_gps_t pf)
173 {
174         size_t i,j,swaps;
175         status_t status;
176
177         if (vector == NULL)
178                 return ERROR_NULL_POINTER;
179
180         for( i=0 ; i < (vector->size) -1 ; i++)
181         {
182                 swaps=0;
183
184                 for( j=0 ; j < ((vector->size) -1 - i); j++)
185                 {
186
187                         if((pf(vector->array_elements[j],vector->array_elements[j+1])
                            )==TRUE)
188                         {
189
```

```
190                                 if((status = swap((vector->array_elements[j]),(vector
                                        ->array_elements[j+1])))!=OK)
191                                         return status;
192
193                                 swaps++;
194                         }
195
196                 }
197                 if(swaps==0) break;
198         }
199         return OK;
200 }
```

5.1.2. vector.h

```
1  #ifndef ADT_VECTOR__H
2  #define ADT_VECTOR__H
3
4  #include "types.h"
5  #include "gps.h"
6
7  #define OUTPUT_FILE_KML_DELIMITER ','
8  #define OUTPUT_FILE_CSV_DELIMITER '|'
9  #define INIT_CHOP 80
10 #define FMT_KML_HEADER "<\?xml version=\"1.0\" encoding=\"UTF-8\"\?>\n<kml xmlns=\"
       http://www.opengis.net/kml/2.2\">\n<Document>\n<name>Rutas</name>\n<description>
       Ejemplos de rutas</description>\n<Style id=\"yellowLineGreenPoly\">\n<LineStyle>\
       n<color>7f00ffff</color>\n<width>4</width>\n</LineStyle>\n<PolyStyle>\n<color>7
       f00ff00</color>\n</PolyStyle>\n</Style>\n<Placemark>\n<name>Relieve absoluto</
       name>\n<description>Pared verde transparente con contornos\namarillos</
       description>\n<styleUrl>#yellowLineGreenPoly</styleUrl>\n<LineString>\n<extrude
       >1</extrude>\n<tessellate>1</tessellate>\n<altitudeMode>absolute</altitudeMode>\n
       <coordinates>"
11
12 #define FMT_KML_FOOTER "</coordinates>\n</LineString>\n</Placemark>\n</Document>\n</
       kml>"
13
14 typedef struct
15 {
16         void **array_elements;
17         size_t size;
18         size_t alloc_size;
19         char *header;
20         char *footer;
21
22 }ADT_vector_t;
23
24
25 status_t ADT_vector_new( ADT_vector_t **vector);
```

```
26 status_t ADT_vector_append(ADT_vector_t **vector, void *element);
27 status_t ADT_vector_delete(ADT_vector_t **vector, destructor_gps_t pf);
28 status_t ADT_vector_export(const ADT_vector_t *vector, FILE *fo, void *tool_box,
       printer_gps_t pf);
29 status_t ADT_vector_set_header(ADT_vector_t *vector, char* string);
30 status_t ADT_vector_set_footer(ADT_vector_t *vector, char* string);
31 status_t export_data_as_csv(ADT_vector_t *vector, FILE *output_file, void* tool_box);
32 status_t export_data_as_kml(ADT_vector_t *vector, FILE *output_file, void* tool_box);
33 status_t ADT_vector_sort(ADT_vector_t *vector, comparator_gps_t pf);
34
35
36 #endif
```

5.1.3. gps.c

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <time.h>
5
6  #include "gps.h"
7
8  status_t load_gps_coordinate(ADT_gps_coordinate_t **coordinate, char ** data)
9  {
10         status_t status;
11         char* temp;
12
13         /*cargo la estructura*/
14         if((*coordinate=(ADT_gps_coordinate_t*)malloc(sizeof(ADT_gps_coordinate_t)))
               ==NULL)
15                 return EXIT_FAILURE;
16
17         /*id_msg*/
18         strcpy((*coordinate)->id_msg,data[ID_MSG_POSITION]);
19
20         /*utc_time*/
21         if((status=load_utc_time(&((*coordinate)->utc_time), data[UTC_POSITION]))!=OK
               )
22         {
23                 return status;
24         }
25
26         /*latitude*/
27         if((status=load_coordinate_latitude(&((*coordinate)->latitude),data[
               LATITUDE_POSITION]))!=OK)
28         {
29                 return status;
30         }
31
```

```
32          /*ns_orientation*/
33          (*coordinate)->ns_orientation=data[NS_ORIENTATION_POSITION][0];
34
35          /*longitude*/
36          if((status=load_coordinate_longitude(&((*coordinate)->longitude),data[
                LONGITUDE_POSITION]))!=OK)
37          {
38                  return status;
39          }
40
41          /*ws_orientation*/
42          (*coordinate)->we_orientation=data[WE_ORIENTATION_POSITION][0];
43
44          /*fix_position*/
45          (*coordinate)->fix_position=strtod(data[FIX_POSITION_POSITION], &temp);
46          if(*temp)
47          {
48                  return ERROR_INVALID_DATA;
49          }
50
51          /*used_satellites*/
52          (*coordinate)->used_satellites=strtoul(data[USED_SATELLITES_POSITION],&temp
                ,10);
53          if(*temp)
54          {
55                  return ERROR_INVALID_DATA;
56          }
57
58          /* HDOP */
59          (*coordinate)->HDOP=strtod(data[HDOP_POSITION],&temp);
60          if(*temp)
61          {
62                  return ERROR_INVALID_DATA;
63          }
64
65          /*sea_level_height*/
66          (*coordinate)->sea_level_height = strtod(data[SEA_LEVEL_POSITION], &temp);
67          if(*temp)
68          {
69                  return ERROR_INVALID_DATA;
70          }
71
72          /*sea_level_height_unit*/
73          (*coordinate)->sea_level_height_unit = data[SEA_LEVEL_UNIT_POSITION][0];
74
75          /*geoid_separation*/
76          (*coordinate)->geoid_separation=strtod(data[GEOID_SEPARATION_POSITION], &temp
                );
77          if(*temp)
78          {
79                  return ERROR_INVALID_DATA;
```

```
80          }
81
82          /*age_diff_corr*/
83          if((status=load_age_diff_corr(&((*coordinate)->age_diff_corr),data[
                AGE_DIFF_CORR_POSITION]))!=OK)
84          {
85                  return status;
86          }
87
88          /*geoid_separation_unit*/
89          (*coordinate)->geoid_separation_unit=data[GEOID_SEPARATION_UNIT_POSITION][0];
90
91          /*diff_reff_position*/
92          if((status=load_diff_reff_position(&((*coordinate)->diff_reff_station_id),
                data[DIFF_REFF_STATION_ID_POSITION]))
93          {
94                  return status;
95          }
96
97          /*check_sum*/
98          if((status=load_check_sum(&((*coordinate)->check_sum),data[
                DIFF_REFF_STATION_ID_POSITION]))
99          {
100                 return status;
101         }
102         return OK;
103 }
104
105 status_t load_coordinate_latitude(coordinate_t *orientation, char* data)
106 {
107         size_t i, position;
108         char *temp, string[MAX_LENGTH];
109
110         if(orientation==NULL||data==NULL)
111                 return ERROR_NULL_POINTER;
112
113         /*latitude*/
114         for(i=0, position=DEGREES_POSITION; i<DEGREES_LATITUDE_WIDTH; i++, position
                ++)
115                 string[i]=data[i];
116
117         string[DEGREES_LATITUDE_WIDTH]='\0';
118
119         orientation->degrees = strtod(string,&temp);;
120
121         if(*temp)
122                 return ERROR_INVALID_DATA;
123
124         /*Minutes*/
125         for(i=0, position=MINUTES_LATITUDE_POSITION; i<MINUTES_LATITUDE_WIDTH; i++,
                position++)
```

```
126                 string[i]=data[position];
127
128         string[MINUTES_LATITUDE_WIDTH]='\0';
129
130         orientation->minutes=strtod(string,&temp);
131
132         if(*temp)
133                 return ERROR_INVALID_DATA;
134
135         return OK;
136 }
137
138 status_t load_coordinate_longitude(coordinate_t *orientation, char* data)
139 {
140         size_t i, position;
141         char *temp, string[MAX_LENGTH];
142
143         if(orientation==NULL||data==NULL)
144                 return ERROR_NULL_POINTER;
145
146         /*longitude*/
147         for(i=0, position=DEGREES_POSITION; i<DEGREES_LONGITUDE_WIDTH; i++, position
                ++)
148                 string[i]=data[i];
149
150         string[DEGREES_LONGITUDE_WIDTH]='\0';
151
152         orientation->degrees = strtod(string,&temp);;
153
154         if(*temp)
155                 return ERROR_INVALID_DATA;
156
157         /*Minutes*/
158         for(i=0, position=MINUTES_LONGITUDE_POSITION; i<MINUTES_LONGITUDE_WIDTH; i++,
                 position++)
159                 string[i]=data[position];
160
161         string[MINUTES_LONGITUDE_WIDTH]='\0';
162
163         orientation->minutes=strtod(string,&temp);
164
165         if(*temp)
166                 return ERROR_INVALID_DATA;
167
168         return OK;
169 }
170
171 status_t load_diff_reff_position(size_t *coordinate, char *data)
172 {
173         size_t i;
174         char string[MAX_LENGTH];
```

```
175         char *temp;
176
177         for(i=0; data[i] != '*'; i++)
178         {
179                 string[i] = data[i];
180         }
181         string[i]='\0';
182
183         *coordinate=strtoul(string,&temp,10);
184
185         if(*temp)
186                 return ERROR_INVALID_DATA;
187
188         return OK;
189 }
190
191 status_t load_check_sum(unsigned char *coordinate, char *data)
192 {
193         size_t position, i;
194         char string[MAX_LENGTH];
195         char *temp;
196
197         for(position=strlen(data)-3, i=0; position < strlen(data) && i<
                CHECK_SUM_WIDTH; position++, i++)
198         {
199                 string[i] = data[position];
200         }
201         string[CHECK_SUM_WIDTH]='\0';
202
203         *coordinate=strtoul(string,&temp,10);
204
205         if(*temp)
206                 return ERROR_INVALID_DATA;
207
208         return OK;
209 }
210
211 status_t load_utc_time(struct tm *utc_time, char *data)
212 {
213         size_t i, position;
214         char string[MAX_LENGTH];
215         char *temp;
216
217         for(i=0, position=HOUR_POSITION; position<HOUR_WIDTH; i++, position++)
218                 string[i]=data[position];
219         string[i]='\0';
220
221         utc_time->tm_hour = strtoul(string, &temp, 10);
222         if(*temp)
223                 return ERROR_INVALID_DATA;
224
```

```
225
226         for(i=0, position = MIN_POSITION; position < MIN_WIDTH; i++, position++)
227         {
228                 string[i]=data[position];
229         }
230         string[i]='\0';
231
232         utc_time->tm_min = strtoul(string, &temp, 10);
233         if(*temp)
234                 return ERROR_INVALID_DATA;
235
236
237         for(i=0, position=SEC_POSITION; position < SEC_WIDTH; i++, position++)
238                 string[i]=data[position];
239         string[i]='\0';
240
241         utc_time->tm_sec = strtoul(string, &temp, 10);
242         if(*temp)
243                 return ERROR_INVALID_DATA;
244
245         return OK;
246 }
247
248 status_t load_age_diff_corr(struct tm *coordinate, char *data)
249 {
250         char string[1];
251         char *temp;
252
253         if(coordinate==NULL)
254                 return ERROR_NULL_POINTER;
255
256         if(data==NULL)
257         {
258                 string[0]='\0';
259                 coordinate->tm_sec =strtoul(string,&temp,10);
260                 if(*temp)
261                         return ERROR_INVALID_DATA;
262         }
263
264         return OK;
265 }
266
267 status_t ADT_gps_coordinate_delete(ADT_gps_coordinate_t *coordinate)
268 {
269         if(coordinate == NULL)
270                 return ERROR_NULL_POINTER;
271
272         free(coordinate);
273         coordinate = NULL;
274         return OK;
275 }
```

```
276
277 status_t ADT_gps_coordinate_export_as_csv(const ADT_gps_coordinate_t *coordinate,
        char delim, FILE *fo)
278 {
279         time_t aux;
280         struct tm* filled_time;
281
282         if(coordinate == NULL || fo == NULL)
283                 return ERROR_NULL_POINTER;
284
285         /*Manejo de la fecha*/
286         time(&aux);
287         filled_time = localtime(&aux);
288
289         /*Impresion de datos*/
290         fprintf(fo,"%i " ,1900+filled_time->tm_year);
291
292         fprintf(fo,"%i  ", filled_time->tm_mon);
293
294         fprintf(fo, "%i ", filled_time->tm_mday);
295
296         fprintf(fo, "%i ", coordinate->utc_time.tm_hour);
297
298         fprintf(fo, "%i ", coordinate->utc_time.tm_min);
299
300         fprintf(fo, "%i ", coordinate->utc_time.tm_sec);
301
302         fprintf(fo, "%c", delim);
303
304         fprintf(fo, "%f %f ", coordinate->latitude.degrees, coordinate->latitude.
            minutes);
305
306         fprintf(fo, "%c", delim);
307
308         fprintf(fo, "%f %f ", coordinate->longitude.degrees, coordinate->longitude.
            minutes);
309
310         fprintf(fo, "%c", delim);
311
312         fprintf(fo, "%f ", coordinate->sea_level_height);
313
314         return OK;
315 }
316
317 status_t ADT_gps_coordinate_export_as_kml(const ADT_gps_coordinate_t *coordinate,
        char delim, FILE *fo)
318 {
319         double decimal_coordinate;
320
321         if(coordinate == NULL || fo == NULL)
322                 return ERROR_NULL_POINTER;
```

```
323
324         decimal_coordinate = coordinate->longitude.degrees + ((coordinate->longitude.
                minutes)/60);
325
326         if(coordinate->ns_orientation == SOUTH_INDICATOR)
327                 decimal_coordinate=-decimal_coordinate;
328
329         fprintf(fo, "%.11f", decimal_coordinate);
330
331         fprintf(fo, "%c", delim);
332
333
334         decimal_coordinate=coordinate->latitude.degrees + (coordinate->latitude.
                minutes)/60;
335
336         if(coordinate->we_orientation == WEST_INDICATOR)
337                 decimal_coordinate=-decimal_coordinate;
338
339         fprintf(fo, "%.11f", decimal_coordinate);
340
341         fprintf(fo, "%c", delim);
342
343         fprintf(fo, "%.11f", coordinate->sea_level_height);
344
345         return OK;
346 }
347
348 bool_t ADT_gps_comparator(ADT_gps_coordinate_t *coordinate_1, ADT_gps_coordinate_t  *
        coordinate_2)
349 {
350         int ret1, ret2;
351         double dif_time;
352
353         ret1 = mktime(&(coordinate_1 ->utc_time));
354         ret2 = mktime(&(coordinate_2 ->utc_time));
355
356         if(ret1 == (-1) || ret2 == (-1))
357         {
358                 return FALSE;
359         }
360
361         dif_time = difftime(ret1,ret2);
362
363         if(dif_time == 0 || dif_time < 0)
364                 return FALSE;
365         else
366                 return TRUE;
367 }
```

5.1.4. gps.h

```
#ifndef TDA_GPS__H
#define TDA_GPS__H

#include <time.h>

#include "types.h"

#define MAX_LENGTH 10

#define ID_MSG_POSITION 0
#define UTC_POSITION 1
#define LATITUDE_POSITION 2
#define NS_ORIENTATION_POSITION 3
#define LONGITUDE_POSITION 4
#define WE_ORIENTATION_POSITION 5
#define FIX_POSITION_POSITION 6
#define USED_SATELLITES_POSITION 7
#define HDOP_POSITION 8
#define SEA_LEVEL_POSITION 9
#define SEA_LEVEL_UNIT_POSITION 10
#define GEOID_SEPARATION_POSITION 11
#define GEOID_SEPARATION_UNIT_POSITION 12
#define AGE_DIFF_CORR_POSITION 14
#define DIFF_REFF_STATION_ID_POSITION 13

#define ID_WIDTH 6
#define DIFF_REFF_STATION_ID_WIDTH 4
#define CHECK_SUM_WIDTH 3

#define HOUR_POSITION 0
#define HOUR_WIDTH 2
#define MIN_POSITION 2
#define MIN_WIDTH 4
#define SEC_POSITION 4
#define SEC_WIDTH 6

#define DEGREES_POSITION 0

#define DEGREES_LATITUDE_WIDTH 2
#define MINUTES_LATITUDE_POSITION 2
#define MINUTES_LATITUDE_WIDTH 9

#define DEGREES_LONGITUDE_WIDTH 3
#define MINUTES_LONGITUDE_POSITION 3
#define MINUTES_LONGITUDE_WIDTH 10


#define SOUTH_INDICATOR 'S'
```

```
49 #define WEST_INDICATOR 'W'
50
51
52 typedef struct{
53         double degrees;
54         double minutes;
55 }coordinate_t;
56
57 typedef struct{
58         char id_msg[ID_WIDTH];
59         struct tm utc_time;
60         coordinate_t latitude;
61         char ns_orientation;
62         coordinate_t longitude;
63         char we_orientation;
64         int fix_position;
65         size_t used_satellites;
66         float HDOP;
67         double sea_level_height;
68         char sea_level_height_unit;
69         double geoid_separation;
70         char geoid_separation_unit;
71         struct tm age_diff_corr;
72         size_t diff_reff_station_id;
73         unsigned char check_sum;
74         char eof;
75 }ADT_gps_coordinate_t;
76
77 typedef status_t (*destructor_gps_t)(ADT_gps_coordinate_t *coordinate);
78 typedef status_t (*printer_gps_t)(const ADT_gps_coordinate_t *coordinate, char delim,
       FILE *fo);
79 typedef bool_t (*comparator_gps_t)(ADT_gps_coordinate_t *coordinate_1,
      ADT_gps_coordinate_t *coordinate_2);
80
81 status_t load_gps_coordinate(ADT_gps_coordinate_t **coordinate, char ** data);
82 status_t load_coordinate_latitude(coordinate_t *orientation, char* data);
83 status_t load_coordinate_longitude(coordinate_t *orientation, char* data);
84 status_t load_diff_reff_position(size_t *coordinate, char* data);
85 status_t load_check_sum(unsigned char *coordinate, char* data);
86 status_t load_utc_time(struct tm *utc_time, char *data);
87 status_t load_age_diff_corr(struct tm *coordinate, char *data);
88 status_t ADT_gps_coordinate_delete(ADT_gps_coordinate_t *coordinate);
89 status_t ADT_gps_coordinate_export_as_csv(const ADT_gps_coordinate_t *coordinate,
      char delim, FILE *fo);
90 status_t ADT_gps_coordinate_export_as_kml(const ADT_gps_coordinate_t *coordinate,
      char delim, FILE *fo);
91 bool_t ADT_gps_comparator(ADT_gps_coordinate_t *coordinate_1, ADT_gps_coordinate_t *
      coordinate_2);
92
93
94
```

```
95 #endif
```

5.2. Implementación

5.2.1. arguments.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "arguments.h"
6  #include "error.h"
7
8  status_t validate_args(char * argv[], size_t arg_amount)
9  {
10         if (argv == NULL)
11                 return ERROR_NULL_POINTER;
12
13         if( arg_amount != MAX_CMD_ARGS)
14                 return ERROR_PROG_INVOCATION;
15
16         return OK;
17 }
```

5.2.2. arguments.h

```
1  #ifndef ARGUMENTS_H
2  #define ARGUMENTS_H
3
4  #include "types.h"
5
6  #define MAX_CMD_ARGS 6
7  #define CMD_ARG_PRINTING_FMT_POSITION 2
8  #define CMD_ARG_OUTPUT_FILE_POSITION 4
9  #define CMD_ARG_INPUT_FILE_POSITION 5
10 #define PRINTING_FMT_CSV "csv"
11 #define PRINTING_FMT_KML "kml"
12
13
14 status_t validate_args(char * argv[], size_t arg_amount );
15
16 #endif
```

5.2.3. utilities.c

```
1  #include <stdio.h>
```

```
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "utilities.h"
6
7
8  status_t read_line(FILE * fi, char ** created_string, bool_t *eof)
9  {
10         char c;
11         size_t alloc_size, used_size;
12         char * aux;
13
14         if (created_string == NULL || fi == NULL)
15                 return ERROR_NULL_POINTER;
16
17         if((*created_string=(char *)malloc(INIT_SIZE*sizeof(char)))== NULL)
18                 return ERROR_MEMORY;
19
20         alloc_size = INIT_SIZE;
21         used_size = 0;
22
23         while((c = fgetc(fi)) != '\n' && c != EOF)
24         {
25                 if(used_size == alloc_size-1)
26                 {
27                         if((aux=(char*)realloc(*created_string,alloc_size + INIT_SIZE
                               )) == NULL)
28                         {
29                                 free(*created_string);
30                                 return ERROR_MEMORY;
31                         }
32                         *created_string = aux;
33                         alloc_size += INIT_SIZE;
34                 }
35                 (*created_string)[used_size ++] = c;
36         }
37
38         *eof=(c==EOF)?TRUE:FALSE;
39
40         (*created_string)[used_size]='\0';
41
42         return OK;
43 }
44
45 bool_t is_valid_line(char *string)
46 {
47         if(!(strncmp(string,SELECTED_ID,ID_LENGHT)))
48         {
49                 if(strstr(string,EMPTY_CSV_FILEDS_LINE)==NULL){
50                         if(check_sum(string)==TRUE)
51                         {
```

```
52                                  return TRUE;
53                          }
54                  }
55          }
56          else
57                  return FALSE;
58  }
59
60  bool_t check_sum(char * string)
61  {
62          unsigned char sum;
63          unsigned char check_sum;
64          size_t position, next, i;
65          char aux[CHECK_SUM_WIDTH];
66          char *temp;
67
68          for(position=1 , next=position+1, sum=string[position]; next<strlen(string)-4
                ; next++){
69
70                  sum=string[next]^sum;
71          }
72
73          for(position=strlen(string)-3, i=0; position < strlen(string) && i<
                CHECK_SUM_WIDTH; position++, i++)
74          {
75                  aux[i]=string[position];
76          }
77          aux[CHECK_SUM_WIDTH]='\0';
78
79          check_sum=strtoul(aux,&temp,16);
80
81          return (check_sum==sum)?TRUE:FALSE;
82  }
83
84  status_t split (const char * s, char del, size_t * amount_fields , char ***
        string_array)
85  {
86          char *str, *q, *p;
87          char delims[2];
88          size_t i;
89
90          if ( s == NULL || amount_fields == NULL || string_array == NULL)
91                  return ERROR_NULL_POINTER;
92
93          delims[0] = del;
94
95          delims[1] = '\0';
96
97          if(strdupl (s,&str) != OK )
98          {
99                  *amount_fields = 0;
```

```
100                     return ERROR_MEMORY;
101             }
102
103             for(i = 0, *amount_fields = 0; str[i]; i++)
104             {
105                     if(str[i] == del)
106                             (*amount_fields)++;
107
108             }
109             (*amount_fields)++;
110
111             if((* string_array = (char **)malloc((*amount_fields) * sizeof(char *))) ==
                    NULL)
112             {
113                     free(str);
114                     *amount_fields=0;
115                     return ERROR_MEMORY;
116             }
117
118             for( i=0, q=str; (p = strtok(q, delims))!= NULL; q=NULL, i++)
119             {
120                     if(strdupl(p, &(*string_array)[i]) != OK)
121                     {
122                             free(str);
123                             destroy_strings(string_array, amount_fields);
124                             *amount_fields=0;
125                             return ERROR_MEMORY;
126                     }
127             }
128             free(str);
129
130             return OK;
131     }
132
133     status_t strdupl(const char *s, char **t)
134     {
135             size_t i;
136
137             if( s == NULL || t == NULL)
138                     return ERROR_NULL_POINTER;
139
140             if((*t =(char *)malloc((strlen(s)+sizeof(char))*sizeof(char))) == NULL)
141                     return ERROR_MEMORY;
142
143             for( i=0; ((*t)[i] = s[i]) ; i++);
144
145             return OK;
146     }
147
148     status_t destroy_strings(char *** string_array, size_t *l)
149     {
```

```
150         size_t i;
151
152         if(string_array == NULL)
153                 return ERROR_NULL_POINTER;
154
155         for(i=0; i < *l; i++)
156         {
157                 free(*string_array[i]);
158                 (*string_array)[i]=NULL;
159         }
160
161         free(*string_array);
162
163         *string_array = NULL;
164
165         *l=0;
166
167         return OK;
168 }
169
170 status_t swap(ADT_gps_coordinate_t *data_1, ADT_gps_coordinate_t *data_2)
171 {
172         ADT_gps_coordinate_t aux;
173
174         if(data_1==NULL || data_2==NULL)
175                 return ERROR_NULL_POINTER;
176
177         aux = *data_1;
178         *data_1 = *data_2;
179         *data_2 = aux;
180
181         return OK;
182 }
```

5.2.4. utilities.h

```
1  #ifndef UTILITIES__H
2  #define UTILITIES__H
3
4  #include "gps.h"
5  #include "types.h"
6
7  #define INIT_SIZE 10
8  #define GROWTH_FACTOR 5
9  #define ID_LENGHT 6
10 #define SELECTED_ID "$GPGGA"
11 #define MASK_CHECKSUM 0x7F
12 #define CHECK_SUM_WIDTH 3
13 #define INPUT_CSV_DELIMITER ','
```

```
14 #define EMPTY_CSV_FILEDS_LINE ",,,,,"
15
16 status_t split (const char * s, char del, size_t * amount_fields , char ***
       string_array);
17 status_t strdupl(const char *s, char **t);
18 status_t read_line(FILE * fi, char ** created_string, bool_t *eof);
19 status_t destroy_strings(char *** string_array, size_t *l);
20 bool_t check_sum(char * string);
21 bool_t is_valid_line(char *string);
22 status_t swap(ADT_gps_coordinate_t *data_1, ADT_gps_coordinate_t *data_2);
23
24 #endif
```

5.3. Errores

5.3.1. error.c

```
1  #include <stdio.h>
2
3  #include "error.h"
4
5  void print_error(status_t status)
6  {
7          char * errors_dictionary[MAX_ERRORS]=
8          {
9                  "Null pointer.",
10                 "Insufficient memory.",
11                 "Incorrect program invocation.",
12                 "Unable to open file.",
13                 "Unable to write in file",
14                 "Invalid data"
15         };
16
17         fprintf(stderr, "%s\n", errors_dictionary[status]);
18 }
```

5.3.2. error.h

```
1  #ifndef ERRORS__H
2  #define ERRORS__H
3
4  #include "types.h"
5
6  #define MAX_ERRORS 6
7
8  void print_error(status_t status);
9
10 #endif
```

5.4. Ejecución

5.4.1. main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "utilities.h"
6  #include "vector.h"
7  #include "gps.h"
8  #include "error.h"
9  #include "arguments.h"
10 #include "types.h"
11
12 int main(int argc, char* argv[])
13 {
14         FILE *input_file, *output_file;
15         status_t status;
16         ADT_vector_t *vector;
17         ADT_gps_coordinate_t *coordinate;
18         char *created_string;
19         bool_t eof;
20         size_t amount_fields, num_linea;
21         char ** string_array;
22
23
24         if((status=validate_args(argv,argc))!=OK)
25         {
26                 print_error(status);
27                 return EXIT_FAILURE;
28         }
29
30         if((input_file=fopen(argv[CMD_ARG_INPUT_FILE_POSITION],"rt"))==NULL)
31         {
32                 status=ERROR_CORRUPT_FILE;
33                 print_error(status);
34                 return EXIT_FAILURE;
35         }
36
37         if((output_file=fopen(argv[CMD_ARG_OUTPUT_FILE_POSITION],"wt"))==NULL)
38         {
39                 fclose(input_file);
40                 status=ERROR_CORRUPT_FILE;
41                 print_error(status);
42                 return EXIT_FAILURE;
43         }
44
```

```
45          if((status=ADT_vector_new(&vector))!=OK)
46          {
47                  ADT_vector_delete(&vector,ADT_gps_coordinate_delete);
48                  fclose(input_file);
49                  fclose(output_file);
50                  print_error(status);
51                  return EXIT_FAILURE;
52          }
53
54          num_linea=0;
55
56          while((status=read_line(input_file, &created_string, &eof))==OK && eof!=TRUE)
57          {
58
59                  if((is_valid_line(created_string))==TRUE)
60                  {
61                          if((status = split (created_string, INPUT_CSV_DELIMITER, &
                                amount_fields , &string_array))!=OK)
62                          {
63                                  ADT_vector_delete(&vector, ADT_gps_coordinate_delete)
                                        ;
64                                  fclose(input_file);
65                                  fclose(output_file);
66                                  print_error(status);
67                                  return EXIT_FAILURE;
68                          }
69
70                          if((status=load_gps_coordinate(&coordinate,string_array))!=OK
                                )
71                          {
72                                  ADT_gps_coordinate_delete(coordinate);
73                                  ADT_vector_delete(&vector, ADT_gps_coordinate_delete)
                                        ;
74                                  fclose(input_file);
75                                  fclose(output_file);
76                                  print_error(status);
77                                  return EXIT_FAILURE;
78                          }
79
80                          if((status=ADT_vector_append(&vector,coordinate))!=OK)
81                          {
82                                  ADT_vector_delete(&vector, ADT_gps_coordinate_delete)
                                        ;
83                                  fclose(input_file);
84                                  fclose(output_file);
85                                  print_error(status);
86                                  return EXIT_FAILURE;
87                          }
88                  }
89          }
90
```

```
91          if((status = ADT_vector_sort( vector, ADT_gps_comparator))!=OK)
92          {
93                  ADT_vector_delete(&vector, ADT_gps_coordinate_delete);
94                  fclose(input_file);
95                  fclose(output_file);
96                  print_error(status);
97                  return EXIT_FAILURE;
98          }
99
100         if(!strcmp(argv[CMD_ARG_PRINTING_FMT_POSITION],PRINTING_FMT_CSV))
101         {
102                 if((status=export_data_as_csv(vector,output_file,
                        OUTPUT_FILE_CSV_DELIMITER ))!=OK)
103                 {
104                         ADT_vector_delete(&vector, ADT_gps_coordinate_delete);
105                         fclose(input_file);
106                         fclose(output_file);
107                         print_error(status);
108                         return EXIT_FAILURE;
109                 }
110         }
111
112         if(!strcmp(argv[CMD_ARG_PRINTING_FMT_POSITION],PRINTING_FMT_KML))
113         {
114                 if((status=export_data_as_kml(vector,output_file,
                        OUTPUT_FILE_KML_DELIMITER ))!=OK)
115                 {
116
117                         ADT_vector_delete(&vector, ADT_gps_coordinate_delete);
118                         fclose(input_file);
119                         fclose(output_file);
120                         print_error(status);
121                         return EXIT_FAILURE;
122                 }
123         }
124
125         if((status=ADT_vector_delete(&vector, ADT_gps_coordinate_delete))!=OK)
126         {
127                 fclose(input_file);
128                 fclose(output_file);
129                 print_error(status);
130                 return EXIT_FAILURE;
131         }
132         fclose(input_file);
133
134         if(fclose(output_file)==EOF)
135         {
136                 status=ERROR_CORRUPT_FILE;
137                 print_error(status);
138                 return EXIT_FAILURE;
139         }
```

```
140
141         return EXIT_SUCCESS;
142 }
```

5.4.2. types.h

```
1  #ifndef TYPES__H
2  #define TYPES__H
3
4  typedef enum
5  {
6          ERROR_NULL_POINTER,
7          ERROR_MEMORY,
8          ERROR_PROG_INVOCATION,
9          ERROR_CORRUPT_FILE,
10         ERROR_WRITING_FILE,
11         ERROR_INVALID_DATA,
12         OK
13 }status_t;
14
15 typedef enum{
16         FALSE,
17         TRUE
18 }bool_t;
19
20 #endif
```

5.4.3. makefile

```
1  CFLAGS = -Wall -ansi -pedantic -o2
2  CC = gcc
3
4  all: make_main
5
6  make_main : main.o utilities.o vector.o gps.o error.o arguments.o
7          $(CC) $(CFLAGS) -o make_main main.o utilities.o vector.o gps.o error.o
               arguments.o
8
9  main.o : main.c utilities.h vector.h gps.h error.h arguments.h types.h
10         $(CC) $(CFLAGS) -o main.o -c main.c
11
12 utilities.o: utilities.c utilities.h types.h
13         $(CC) $(CFLAGS) -o utilities.o -c utilities.c
14
15 vector.o: vector.c vector.h types.h
16         $(CC) $(CFLAGS) -o vector.o -c vector.c
17
18 gps.o: gps.c gps.h types.h
```

```
19          $(CC) $(CFLAGS) -o gps.o -c gps.c
20
21  error.o: error.c error.h types.h
22          $(CC) $(CFLAGS) -o error.o -c error.c
23
24  arguments.o: arguments.c arguments.h types.h
25          $(CC) $(CFLAGS) -o arguments.o -c arguments.c
```

6. Apéndice IV: Resultados de ejecución

6.1. Archivos CSV

6.1.1. Con trayectoria A

```
2019 5  28 21 14 47 |34.000000 37.835521 |58.000000 22.403242 |17.700000
2019 5  28 21 14 48 |34.000000 37.837772 |58.000000 22.391915 |5.500000
2019 5  28 21 14 49 |34.000000 37.837163 |58.000000 22.392834 |3.800000
2019 5  28 21 14 50 |34.000000 37.836751 |58.000000 22.392113 |5.000000
2019 5  28 21 14 51 |34.000000 37.835735 |58.000000 22.391892 |6.400000
2019 5  28 21 14 52 |34.000000 37.835370 |58.000000 22.390773 |9.000000
2019 5  28 21 14 53 |34.000000 37.834453 |58.000000 22.391315 |7.500000
2019 5  28 21 14 54 |34.000000 37.834058 |58.000000 22.390139 |8.300000
2019 5  28 21 14 55 |34.000000 37.833346 |58.000000 22.388798 |8.800000
2019 5  28 21 14 56 |34.000000 37.832764 |58.000000 22.388066 |8.700000
2019 5  28 21 14 57 |34.000000 37.831057 |58.000000 22.387065 |8.900000
2019 5  28 21 14 58 |34.000000 37.830194 |58.000000 22.386583 |9.400000
2019 5  28 21 14 59 |34.000000 37.830068 |58.000000 22.385824 |9.300000
2019 5  28 21 15 0 |34.000000 37.829798 |58.000000 22.385074 |9.700000
2019 5  28 21 15 1 |34.000000 37.829083 |58.000000 22.383613 |8.800000
2019 5  28 21 15 2 |34.000000 37.827777 |58.000000 22.382493 |9.900000
2019 5  28 21 15 3 |34.000000 37.827069 |58.000000 22.382111 |10.800000
2019 5  28 21 15 4 |34.000000 37.826323 |58.000000 22.381977 |11.700000
2019 5  28 21 15 5 |34.000000 37.825335 |58.000000 22.381969 |11.200000
2019 5  28 21 15 6 |34.000000 37.824622 |58.000000 22.380659 |10.900000
2019 5  28 21 15 7 |34.000000 37.824008 |58.000000 22.380042 |10.600000
2019 5  28 21 15 8 |34.000000 37.823368 |58.000000 22.379543 |9.200000
2019 5  28 21 15 9 |34.000000 37.821954 |58.000000 22.378794 |10.600000
2019 5  28 21 15 10 |34.000000 37.821218 |58.000000 22.378132 |9.100000
2019 5  28 21 15 11 |34.000000 37.820526 |58.000000 22.377524 |8.200000
2019 5  28 21 15 12 |34.000000 37.819521 |58.000000 22.376822 |7.500000
2019 5  28 21 15 13 |34.000000 37.818945 |58.000000 22.375521 |6.800000
2019 5  28 21 15 14 |34.000000 37.818167 |58.000000 22.373567 |4.500000
2019 5  28 21 15 15 |34.000000 37.817909 |58.000000 22.372277 |3.800000
2019 5  28 21 15 16 |34.000000 37.817802 |58.000000 22.371015 |3.200000
2019 5  28 21 15 17 |34.000000 37.817802 |58.000000 22.369900 |2.900000
2019 5  28 21 15 18 |34.000000 37.817846 |58.000000 22.368279 |3.400000
2019 5  28 21 15 19 |34.000000 37.817672 |58.000000 22.367011 |3.300000
2019 5  28 21 15 20 |34.000000 37.818117 |58.000000 22.365964 |2.900000
2019 5  28 21 15 21 |34.000000 37.818421 |58.000000 22.364708 |2.800000
2019 5  28 21 15 22 |34.000000 37.819148 |58.000000 22.363400 |3.000000
2019 5  28 21 15 23 |34.000000 37.819129 |58.000000 22.363032 |3.400000
2019 5  28 21 15 24 |34.000000 37.819204 |58.000000 22.362941 |3.300000
2019 5  28 21 15 25 |34.000000 37.819397 |58.000000 22.362704 |3.400000
2019 5  28 21 15 26 |34.000000 37.819217 |58.000000 22.362457 |3.400000
2019 5  28 21 15 27 |34.000000 37.819378 |58.000000 22.362375 |3.700000
2019 5  28 21 15 28 |34.000000 37.819927 |58.000000 22.360180 |3.200000
2019 5  28 21 15 29 |34.000000 37.820398 |58.000000 22.357902 |2.900000
2019 5  28 21 15 30 |34.000000 37.826270 |58.000000 22.358290 |0.400000
2019 5  28 21 15 31 |34.000000 37.825584 |58.000000 22.356983 |0.300000
```

```
2019 5  28 21 15 32 |34.000000 37.826338 |58.000000 22.355880 |-0.000000
2019 5  28 21 15 33 |34.000000 37.826597 |58.000000 22.355333 |-1.500000
2019 5  28 21 15 34 |34.000000 37.827637 |58.000000 22.353294 |-1.700000
2019 5  28 21 15 35 |34.000000 37.828700 |58.000000 22.351391 |-2.000000
2019 5  28 21 15 36 |34.000000 37.828863 |58.000000 22.349736 |-1.200000
2019 5  28 21 15 37 |34.000000 37.828468 |58.000000 22.348193 |-2.500000
2019 5  28 21 15 38 |34.000000 37.829283 |58.000000 22.347571 |-3.200000
2019 5  28 21 15 39 |34.000000 37.829664 |58.000000 22.346320 |-3.300000
2019 5  28 21 15 40 |34.000000 37.830153 |58.000000 22.345877 |-3.500000
2019 5  28 21 15 41 |34.000000 37.830678 |58.000000 22.345003 |-4.200000
2019 5  28 21 15 42 |34.000000 37.831027 |58.000000 22.344246 |-3.100000
2019 5  28 21 15 43 |34.000000 37.831295 |58.000000 22.343426 |-2.700000
2019 5  28 21 15 44 |34.000000 37.831863 |58.000000 22.342112 |-3.200000
2019 5  28 21 15 45 |34.000000 37.831561 |58.000000 22.341217 |-3.600000
2019 5  28 21 15 46 |34.000000 37.832179 |58.000000 22.340382 |-4.800000
2019 5  28 21 15 47 |34.000000 37.832796 |58.000000 22.338597 |-5.500000
2019 5  28 21 15 48 |34.000000 37.832507 |58.000000 22.336929 |-5.000000
2019 5  28 21 15 49 |34.000000 37.832334 |58.000000 22.335625 |-4.800000
2019 5  28 21 15 50 |34.000000 37.832401 |58.000000 22.334442 |-5.200000
2019 5  28 21 15 51 |34.000000 37.833165 |58.000000 22.333403 |-4.600000
2019 5  28 21 15 52 |34.000000 37.834052 |58.000000 22.332450 |-4.800000
2019 5  28 21 15 53 |34.000000 37.834425 |58.000000 22.331592 |-3.500000
2019 5  28 21 15 54 |34.000000 37.834779 |58.000000 22.330262 |-2.800000
2019 5  28 21 15 55 |34.000000 37.835502 |58.000000 22.329502 |-2.000000
2019 5  28 21 15 56 |34.000000 37.835801 |58.000000 22.328375 |-1.300000
2019 5  28 21 15 57 |34.000000 37.836421 |58.000000 22.327586 |-0.400000
2019 5  28 21 15 58 |34.000000 37.836666 |58.000000 22.326879 |0.100000
2019 5  28 21 15 59 |34.000000 37.837319 |58.000000 22.326159 |1.300000
2019 5  28 21 16 0 |34.000000 37.837302 |58.000000 22.324890 |2.400000
2019 5  28 21 16 1 |34.000000 37.837800 |58.000000 22.324034 |2.600000
2019 5  28 21 16 2 |34.000000 37.838086 |58.000000 22.323228 |5.500000
2019 5  28 21 16 3 |34.000000 37.838909 |58.000000 22.322934 |6.800000
2019 5  28 21 16 4 |34.000000 37.840080 |58.000000 22.322618 |7.400000
2019 5  28 21 16 5 |34.000000 37.841118 |58.000000 22.322626 |7.400000
2019 5  28 21 16 6 |34.000000 37.842541 |58.000000 22.323009 |8.000000
```

6.1.2. Con trayectoria B

```
2019 5  28 21 29 48 |34.000000 37.843365 |58.000000 22.303517 |43.600000
2019 5  28 21 29 49 |34.000000 37.847693 |58.000000 22.310842 |32.600000
2019 5  28 21 29 50 |34.000000 37.850327 |58.000000 22.316112 |25.500000
2019 5  28 21 29 51 |34.000000 37.852133 |58.000000 22.320239 |16.600000
2019 5  28 21 29 52 |34.000000 37.853489 |58.000000 22.324068 |11.800000
2019 5  28 21 29 53 |34.000000 37.854090 |58.000000 22.324500 |12.600000
2019 5  28 21 29 54 |34.000000 37.854755 |58.000000 22.322733 |17.700000
2019 5  28 21 29 55 |34.000000 37.855914 |58.000000 22.323074 |17.500000
2019 5  28 21 29 56 |34.000000 37.856637 |58.000000 22.322333 |20.100000
2019 5  28 21 29 57 |34.000000 37.855699 |58.000000 22.322766 |21.900000
2019 5  28 21 29 58 |34.000000 37.857084 |58.000000 22.322928 |22.200000
```

```
2019 5  28 21 29 59 |34.000000 37.858537 |58.000000 22.323211 |22.500000
2019 5  28 21 30 0 |34.000000 37.859933 |58.000000 22.324000 |20.200000
2019 5  28 21 30 1 |34.000000 37.860761 |58.000000 22.324487 |21.000000
2019 5  28 21 30 2 |34.000000 37.861402 |58.000000 22.325137 |20.600000
2019 5  28 21 30 3 |34.000000 37.862647 |58.000000 22.326920 |19.200000
2019 5  28 21 30 4 |34.000000 37.863775 |58.000000 22.327890 |18.300000
2019 5  28 21 30 5 |34.000000 37.864110 |58.000000 22.327856 |17.200000
2019 5  28 21 30 6 |34.000000 37.864260 |58.000000 22.329072 |16.800000
2019 5  28 21 30 7 |34.000000 37.864901 |58.000000 22.329797 |16.100000
2019 5  28 21 30 8 |34.000000 37.865518 |58.000000 22.330958 |14.900000
2019 5  28 21 30 9 |34.000000 37.866372 |58.000000 22.331641 |14.200000
2019 5  28 21 30 10 |34.000000 37.867172 |58.000000 22.332311 |13.800000
2019 5  28 21 30 11 |34.000000 37.867864 |58.000000 22.333327 |14.100000
2019 5  28 21 30 12 |34.000000 37.868997 |58.000000 22.332757 |14.100000
2019 5  28 21 30 13 |34.000000 37.869008 |58.000000 22.333045 |14.700000
2019 5  28 21 30 14 |34.000000 37.869310 |58.000000 22.333737 |16.300000
2019 5  28 21 30 15 |34.000000 37.869685 |58.000000 22.334642 |16.600000
2019 5  28 21 30 16 |34.000000 37.870286 |58.000000 22.335977 |15.900000
2019 5  28 21 30 17 |34.000000 37.871349 |58.000000 22.335447 |15.600000
2019 5  28 21 30 18 |34.000000 37.872198 |58.000000 22.335908 |15.600000
2019 5  28 21 30 19 |34.000000 37.873145 |58.000000 22.335912 |15.400000
2019 5  28 21 30 20 |34.000000 37.873988 |58.000000 22.337513 |15.200000
2019 5  28 21 30 21 |34.000000 37.874896 |58.000000 22.339446 |14.300000
2019 5  28 21 30 22 |34.000000 37.875945 |58.000000 22.340124 |14.100000
2019 5  28 21 30 23 |34.000000 37.876776 |58.000000 22.340683 |13.700000
2019 5  28 21 30 24 |34.000000 37.878120 |58.000000 22.340736 |14.400000
2019 5  28 21 30 25 |34.000000 37.878577 |58.000000 22.340242 |14.200000
2019 5  28 21 30 26 |34.000000 37.879090 |58.000000 22.341032 |13.400000
2019 5  28 21 30 27 |34.000000 37.880308 |58.000000 22.341555 |13.500000
2019 5  28 21 30 28 |34.000000 37.881184 |58.000000 22.341979 |13.200000
2019 5  28 21 30 29 |34.000000 37.881953 |58.000000 22.342232 |13.000000
2019 5  28 21 30 30 |34.000000 37.882170 |58.000000 22.342154 |12.300000
2019 5  28 21 30 31 |34.000000 37.882984 |58.000000 22.342773 |12.700000
2019 5  28 21 30 32 |34.000000 37.884106 |58.000000 22.345231 |12.400000
2019 5  28 21 30 33 |34.000000 37.884809 |58.000000 22.346668 |12.500000
2019 5  28 21 30 34 |34.000000 37.886137 |58.000000 22.348737 |12.800000
2019 5  28 21 30 35 |34.000000 37.885932 |58.000000 22.353113 |9.100000
2019 5  28 21 30 36 |34.000000 37.886820 |58.000000 22.355372 |7.700000
2019 5  28 21 30 37 |34.000000 37.886901 |58.000000 22.356705 |7.200000
2019 5  28 21 30 38 |34.000000 37.888292 |58.000000 22.357912 |7.000000
2019 5  28 21 30 39 |34.000000 37.886402 |58.000000 22.359714 |9.600000
2019 5  28 21 30 40 |34.000000 37.886198 |58.000000 22.360897 |11.300000
2019 5  28 21 30 41 |34.000000 37.886227 |58.000000 22.362202 |10.900000
2019 5  28 21 30 42 |34.000000 37.885640 |58.000000 22.363641 |12.800000
2019 5  28 21 30 43 |34.000000 37.885111 |58.000000 22.364510 |14.100000
2019 5  28 21 30 44 |34.000000 37.884490 |58.000000 22.365232 |14.600000
2019 5  28 21 30 45 |34.000000 37.883561 |58.000000 22.366432 |15.100000
2019 5  28 21 30 46 |34.000000 37.883711 |58.000000 22.367303 |14.800000
2019 5  28 21 30 47 |34.000000 37.883523 |58.000000 22.368698 |11.500000
2019 5  28 21 30 48 |34.000000 37.883334 |58.000000 22.370006 |10.800000
2019 5  28 21 30 49 |34.000000 37.882813 |58.000000 22.373311 |7.200000
```

```
2019 5  28 21 30 50 |34.000000 37.882845 |58.000000 22.374848 |7.200000
2019 5  28 21 30 51 |34.000000 37.882014 |58.000000 22.375489 |8.700000
2019 5  28 21 30 52 |34.000000 37.880934 |58.000000 22.376409 |11.400000
2019 5  28 21 30 53 |34.000000 37.880288 |58.000000 22.378278 |9.600000
2019 5  28 21 30 54 |34.000000 37.878685 |58.000000 22.380719 |10.100000
2019 5  28 21 30 55 |34.000000 37.877751 |58.000000 22.382364 |10.900000
2019 5  28 21 30 56 |34.000000 37.876611 |58.000000 22.383021 |12.300000
2019 5  28 21 30 57 |34.000000 37.876153 |58.000000 22.382666 |14.100000
2019 5  28 21 30 58 |34.000000 37.876602 |58.000000 22.384072 |15.000000
2019 5  28 21 30 59 |34.000000 37.877044 |58.000000 22.386502 |10.500000
2019 5  28 21 31 0 |34.000000 37.877649 |58.000000 22.388134 |8.900000
2019 5  28 21 31 1 |34.000000 37.876557 |58.000000 22.388228 |7.800000
2019 5  28 21 31 2 |34.000000 37.876092 |58.000000 22.388695 |6.800000
2019 5  28 21 31 3 |34.000000 37.875691 |58.000000 22.390278 |6.400000
2019 5  28 21 31 4 |34.000000 37.874845 |58.000000 22.391102 |6.100000
2019 5  28 21 31 5 |34.000000 37.874527 |58.000000 22.391444 |4.900000
2019 5  28 21 31 6 |34.000000 37.873915 |58.000000 22.392062 |4.000000
2019 5  28 21 31 7 |34.000000 37.873009 |58.000000 22.393534 |2.800000
2019 5  28 21 31 8 |34.000000 37.872383 |58.000000 22.394231 |1.800000
2019 5  28 21 31 9 |34.000000 37.871738 |58.000000 22.394848 |1.800000
2019 5  28 21 31 10 |34.000000 37.871967 |58.000000 22.395051 |1.100000
2019 5  28 21 31 11 |34.000000 37.870610 |58.000000 22.396682 |-0.200000
2019 5  28 21 31 12 |34.000000 37.870141 |58.000000 22.396940 |-1.600000
2019 5  28 21 31 13 |34.000000 37.870451 |58.000000 22.397134 |-1.900000
2019 5  28 21 31 14 |34.000000 37.870336 |58.000000 22.397142 |-1.900000
2019 5  28 21 31 15 |34.000000 37.869501 |58.000000 22.397870 |-2.600000
2019 5  28 21 31 16 |34.000000 37.867652 |58.000000 22.398861 |-2.200000
2019 5  28 21 31 17 |34.000000 37.866657 |58.000000 22.399655 |-2.700000
2019 5  28 21 31 18 |34.000000 37.867432 |58.000000 22.400283 |-4.100000
2019 5  28 21 31 19 |34.000000 37.866042 |58.000000 22.400727 |-2.700000
2019 5  28 21 31 20 |34.000000 37.864373 |58.000000 22.401795 |-0.300000
2019 5  28 21 31 21 |34.000000 37.862098 |58.000000 22.402064 |1.900000
2019 5  28 21 31 22 |34.000000 37.860268 |58.000000 22.401956 |2.800000
2019 5  28 21 31 23 |34.000000 37.858732 |58.000000 22.401388 |3.200000
2019 5  28 21 31 24 |34.000000 37.857390 |58.000000 22.401005 |3.300000
2019 5  28 21 31 25 |34.000000 37.855903 |58.000000 22.400561 |3.400000
2019 5  28 21 31 26 |34.000000 37.855278 |58.000000 22.399032 |2.900000
2019 5  28 21 31 27 |34.000000 37.854179 |58.000000 22.398016 |2.400000
2019 5  28 21 31 28 |34.000000 37.853265 |58.000000 22.396884 |1.800000
2019 5  28 21 31 29 |34.000000 37.852172 |58.000000 22.396287 |2.100000
2019 5  28 21 31 30 |34.000000 37.851394 |58.000000 22.395455 |2.300000
2019 5  28 21 31 31 |34.000000 37.850575 |58.000000 22.394602 |2.700000
2019 5  28 21 31 32 |34.000000 37.849830 |58.000000 22.394386 |3.500000
2019 5  28 21 31 33 |34.000000 37.848879 |58.000000 22.393811 |5.000000
2019 5  28 21 31 34 |34.000000 37.847761 |58.000000 22.393067 |5.200000
```

6.2. Archivos KML

6.2.1. Con trayectoria A

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<name>Rutas</name>
<description>Ejemplos de rutas</description>
<Style id="yellowLineGreenPoly">
<LineStyle>
<color>7f00ffff</color>
<width>4</width>
</LineStyle>
<PolyStyle>
<color>7f00ff00</color>
</PolyStyle>
</Style>
<Placemark>
<name>Relieve absoluto</name>
<description>Pared verde transparente con contornos
amarillos</description>
<styleUrl>#yellowLineGreenPoly</styleUrl>
<LineString>
<extrude>1</extrude>
<tessellate>1</tessellate>
<altitudeMode>absolute</altitudeMode>
<coordinates>
-58.37338736667,-34.63059201667,17.70000000000
-58.37319858333,-34.63062953333,5.50000000000
-58.37321390000,-34.63061938333,3.80000000000
-58.37320188333,-34.63061251667,5.00000000000
-58.37319820000,-34.63059558333,6.40000000000
-58.37317955000,-34.63058950000,9.00000000000
-58.37318858333,-34.63057421667,7.50000000000
-58.37316898333,-34.63056763333,8.30000000000
-58.37314663333,-34.63055576667,8.80000000000
-58.37313443333,-34.63054606667,8.70000000000
-58.37311775000,-34.63051761667,8.90000000000
-58.37310971667,-34.63050323333,9.40000000000
-58.37309706667,-34.63050113333,9.30000000000
-58.37308456667,-34.63049663333,9.70000000000
-58.37306021667,-34.63048471667,8.80000000000
-58.37304155000,-34.63046295000,9.90000000000
-58.37303518333,-34.63045115000,10.80000000000
-58.37303295000,-34.63043871667,11.70000000000
-58.37303281667,-34.63042225000,11.20000000000
-58.37301098333,-34.63041036667,10.90000000000
-58.37300070000,-34.63040013333,10.60000000000
-58.37299238333,-34.63038946667,9.20000000000
-58.37297990000,-34.63036590000,10.60000000000
```

```
-58.37296886667,-34.63035363333,9.10000000000
-58.37295873333,-34.63034210000,8.20000000000
-58.37294703333,-34.63032535000,7.50000000000
-58.37292535000,-34.63031575000,6.80000000000
-58.37289278333,-34.63030278333,4.50000000000
-58.37287128333,-34.63029848333,3.80000000000
-58.37285025000,-34.63029670000,3.20000000000
-58.37283166667,-34.63029670000,2.90000000000
-58.37280465000,-34.63029743333,3.40000000000
-58.37278351667,-34.63029453333,3.30000000000
-58.37276606667,-34.63030195000,2.90000000000
-58.37274513333,-34.63030701667,2.80000000000
-58.37272333333,-34.63031913333,3.00000000000
-58.37271720000,-34.63031881667,3.40000000000
-58.37271568333,-34.63032006667,3.30000000000
-58.37271173333,-34.63032328333,3.40000000000
-58.37270761667,-34.63032028333,3.40000000000
-58.37270625000,-34.63032296667,3.70000000000
-58.37266966667,-34.63033211667,3.20000000000
-58.37263170000,-34.63033996667,2.90000000000
-58.37263816667,-34.63043783333,0.40000000000
-58.37261638333,-34.63042640000,0.30000000000
-58.37259800000,-34.63043896667,-0.00000000000
-58.37258888333,-34.63044328333,-1.50000000000
-58.37255490000,-34.63046061667,-1.70000000000
-58.37252318333,-34.63047833333,-2.00000000000
-58.37249560000,-34.63048105000,-1.20000000000
-58.37246988333,-34.63047446667,-2.50000000000
-58.37245951667,-34.63048805000,-3.20000000000
-58.37243866667,-34.63049440000,-3.30000000000
-58.37243128333,-34.63050255000,-3.50000000000
-58.37241671667,-34.63051130000,-4.20000000000
-58.37240410000,-34.63051711667,-3.10000000000
-58.37239043333,-34.63052158333,-2.70000000000
-58.37236853333,-34.63053105000,-3.20000000000
-58.37235361667,-34.63052601667,-3.60000000000
-58.37233970000,-34.63053631667,-4.80000000000
-58.37230995000,-34.63054660000,-5.50000000000
-58.37228215000,-34.63054178333,-5.00000000000
-58.37226041667,-34.63053890000,-4.80000000000
-58.37224070000,-34.63054001667,-5.20000000000
-58.37222338333,-34.63055275000,-4.60000000000
-58.37220750000,-34.63056753333,-4.80000000000
-58.37219320000,-34.63057375000,-3.50000000000
-58.37217103333,-34.63057965000,-2.80000000000
-58.37215836667,-34.63059170000,-2.00000000000
-58.37213958333,-34.63059668333,-1.30000000000
-58.37212643333,-34.63060701667,-0.40000000000
-58.37211465000,-34.63061110000,0.10000000000
-58.37210265000,-34.63062198333,1.30000000000
-58.37208150000,-34.63062170000,2.40000000000
```

```
-58.37206723333,-34.63063000000,2.60000000000
-58.37205380000,-34.63063476667,5.50000000000
-58.37204890000,-34.63064848333,6.80000000000
-58.37204363333,-34.63066800000,7.40000000000
-58.37204376667,-34.63068530000,7.40000000000
-58.37205015000,-34.63070901667,8.00000000000
</coordinates>
</LineString>
</Placemark>
</Document>
</kml>
```

Figura 4: Trayectoria A

6.2.2. Con trayectoria B

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<name>Rutas</name>
<description>Ejemplos de rutas</description>
<Style id="yellowLineGreenPoly">
<LineStyle>
<color>7f00ffff</color>
<width>4</width>
</LineStyle>
<PolyStyle>
<color>7f00ff00</color>
</PolyStyle>
</Style>
<Placemark>
```

```
<name>Relieve absoluto</name>
<description>Pared verde transparente con contornos
amarillos</description>
<styleUrl>#yellowLineGreenPoly</styleUrl>
<LineString>
<extrude>1</extrude>
<tessellate>1</tessellate>
<altitudeMode>absolute</altitudeMode>
<coordinates>
-58.37172528333,-34.63072275000,43.60000000000
-58.37184736667,-34.63079488333,32.60000000000
-58.37193520000,-34.63083878333,25.50000000000
-58.37200398333,-34.63086888333,16.60000000000
-58.37206780000,-34.63089148333,11.80000000000
-58.37207500000,-34.63090150000,12.60000000000
-58.37204555000,-34.63091258333,17.70000000000
-58.37205123333,-34.63093190000,17.50000000000
-58.37203888333,-34.63094395000,20.10000000000
-58.37204610000,-34.63092831667,21.90000000000
-58.37204880000,-34.63095140000,22.20000000000
-58.37205351667,-34.63097561667,22.50000000000
-58.37206666667,-34.63099888333,20.20000000000
-58.37207478333,-34.63101268333,21.00000000000
-58.37208561667,-34.63102336667,20.60000000000
-58.37211533333,-34.63104411667,19.20000000000
-58.37213150000,-34.63106291667,18.30000000000
-58.37213093333,-34.63106850000,17.20000000000
-58.37215120000,-34.63107100000,16.80000000000
-58.37216328333,-34.63108168333,16.10000000000
-58.37218263333,-34.63109196667,14.90000000000
-58.37219401667,-34.63110620000,14.20000000000
-58.37220518333,-34.63111953333,13.80000000000
-58.37222211667,-34.63113106667,14.10000000000
-58.37221261667,-34.63114995000,14.10000000000
-58.37221741667,-34.63115013333,14.70000000000
-58.37222895000,-34.63115516667,16.30000000000
-58.37224403333,-34.63116141667,16.60000000000
-58.37226628333,-34.63117143333,15.90000000000
-58.37225745000,-34.63118915000,15.60000000000
-58.37226513333,-34.63120330000,15.60000000000
-58.37226520000,-34.63121908333,15.40000000000
-58.37229188333,-34.63123313333,15.20000000000
-58.37232410000,-34.63124826667,14.30000000000
-58.37233540000,-34.63126575000,14.10000000000
-58.37234471667,-34.63127960000,13.70000000000
-58.37234560000,-34.63130200000,14.40000000000
-58.37233736667,-34.63130961667,14.20000000000
-58.37235053333,-34.63131816667,13.40000000000
-58.37235925000,-34.63133846667,13.50000000000
-58.37236631667,-34.63135306667,13.20000000000
-58.37237053333,-34.63136588333,13.00000000000
```

```
-58.37236923333,-34.63136950000,12.30000000000
-58.37237955000,-34.63138306667,12.70000000000
-58.37242051667,-34.63140176667,12.40000000000
-58.37244446667,-34.63141348333,12.50000000000
-58.37247895000,-34.63143561667,12.80000000000
-58.37255188333,-34.63143220000,9.10000000000
-58.37258953333,-34.63144700000,7.70000000000
-58.37261175000,-34.63144835000,7.20000000000
-58.37263186667,-34.63147153333,7.00000000000
-58.37266190000,-34.63144003333,9.60000000000
-58.37268161667,-34.63143663333,11.30000000000
-58.37270336667,-34.63143711667,10.90000000000
-58.37272735000,-34.63142733333,12.80000000000
-58.37274183333,-34.63141851667,14.10000000000
-58.37275386667,-34.63140816667,14.60000000000
-58.37277386667,-34.63139268333,15.10000000000
-58.37278838333,-34.63139518333,14.80000000000
-58.37281163333,-34.63139205000,11.50000000000
-58.37283343333,-34.63138890000,10.80000000000
-58.37288851667,-34.63138021667,7.20000000000
-58.37291413333,-34.63138075000,7.20000000000
-58.37292481667,-34.63136690000,8.70000000000
-58.37294015000,-34.63134890000,11.40000000000
-58.37297130000,-34.63133813333,9.60000000000
-58.37301198333,-34.63131141667,10.10000000000
-58.37303940000,-34.63129585000,10.90000000000
-58.37305035000,-34.63127685000,12.30000000000
-58.37304443333,-34.63126921667,14.10000000000
-58.37306786667,-34.63127670000,15.00000000000
-58.37310836667,-34.63128406667,10.50000000000
-58.37313556667,-34.63129415000,8.90000000000
-58.37313713333,-34.63127595000,7.80000000000
-58.37314491667,-34.63126820000,6.80000000000
-58.37317130000,-34.63126151667,6.40000000000
-58.37318503333,-34.63124741667,6.10000000000
-58.37319073333,-34.63124211667,4.90000000000
-58.37320103333,-34.63123191667,4.00000000000
-58.37322556667,-34.63121681667,2.80000000000
-58.37323718333,-34.63120638333,1.80000000000
-58.37324746667,-34.63119563333,1.80000000000
-58.37325085000,-34.63119945000,1.10000000000
-58.37327803333,-34.63117683333,-0.20000000000
-58.37328233333,-34.63116901667,-1.60000000000
-58.37328556667,-34.63117418333,-1.90000000000
-58.37328570000,-34.63117226667,-1.90000000000
-58.37329783333,-34.63115835000,-2.60000000000
-58.37331435000,-34.63112753333,-2.20000000000
-58.37332758333,-34.63111095000,-2.70000000000
-58.37333805000,-34.63112386667,-4.10000000000
-58.37334545000,-34.63110070000,-2.70000000000
-58.37336325000,-34.63107288333,-0.30000000000
```

```
-58.37336773333,-34.63103496667,1.90000000000
-58.37336593333,-34.63100446667,2.80000000000
-58.37335646667,-34.63097886667,3.20000000000
-58.37335008333,-34.63095650000,3.30000000000
-58.37334268333,-34.63093171667,3.40000000000
-58.37331720000,-34.63092130000,2.90000000000
-58.37330026667,-34.63090298333,2.40000000000
-58.37328140000,-34.63088775000,1.80000000000
-58.37327145000,-34.63086953333,2.10000000000
-58.37325758333,-34.63085656667,2.30000000000
-58.37324336667,-34.63084291667,2.70000000000
-58.37323976667,-34.63083050000,3.50000000000
-58.37323018333,-34.63081465000,5.00000000000
-58.37321778333,-34.63079601667,5.20000000000
</coordinates>
</LineString>
</Placemark>
</Document>
</kml>
```

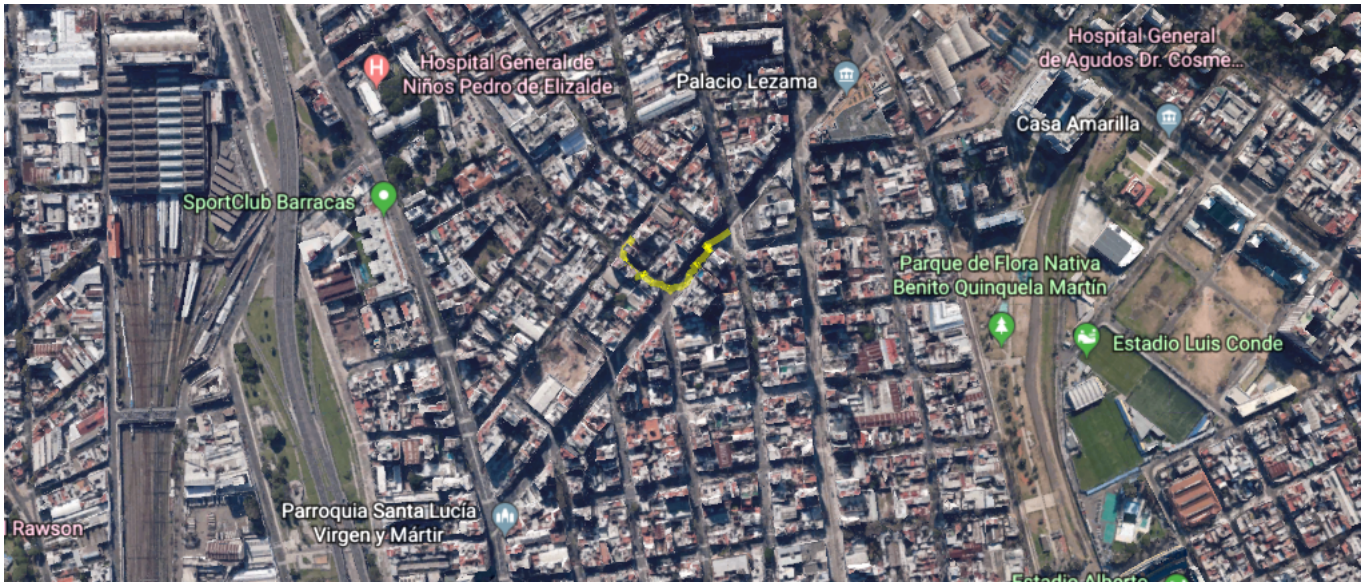


Figura 5: Trayectoria B