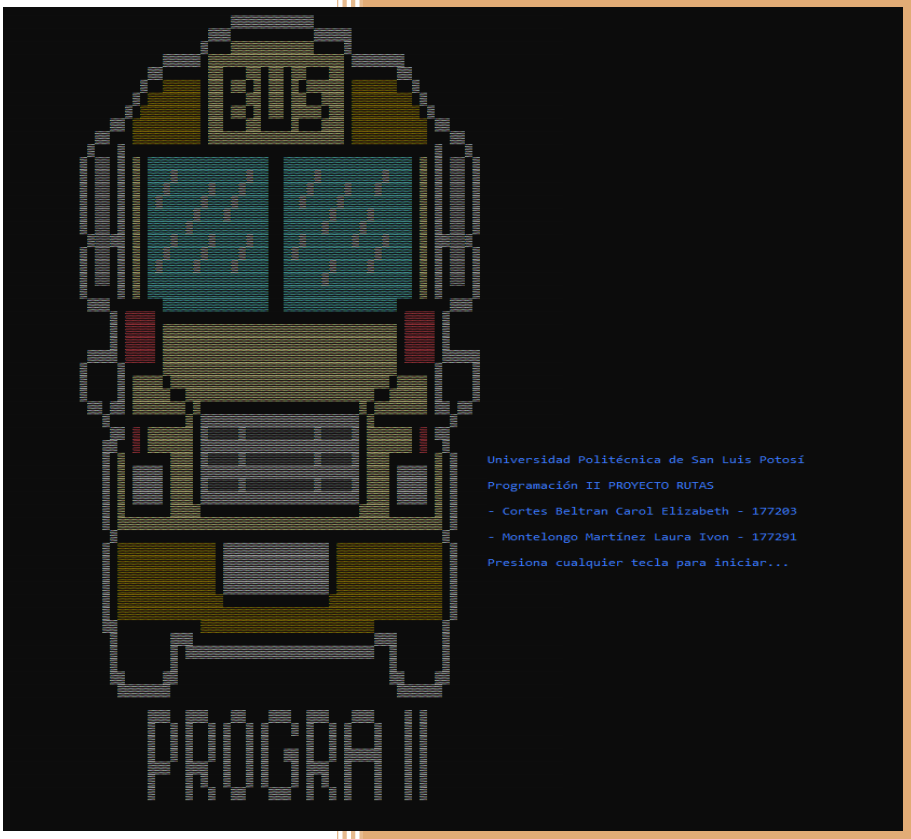
UNIVERSIDAD POLITÉCNICA DE SAN LUIS POTOSÍ

2022

Manual del Programador

“RUTAS DE TRANSPORTE EN SAN LUIS POTOSÍ”



DESARROLLADORAS

Cortes Beltran Carol Elizabeth 177203

Montelongo Martínez Laura Ivon 177291

MATERIA:

Programación II

PROFESORA:

Profa. Guadalupe Ledesma Ramos

Otoño 2022

Indice

Tabla de ilustraciones

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

INTRODUCCION

El presente documento es un manual que busca describir un programa de “Rutas”, desarrollado por Cortés Beltrán Carol Elizabeth y Montelongo Martínez Laura Ivon, como proyecto final de la materia de Programación II en la Universidad Politécnica de San Luis Potosí de la carrera de ingeniería en Tecnologías de la información.

Utilizando listas (simples y dobles), estructuras, archivos y apuntadores proporcionadas por la profesora Guadalupe Ledesma Ramos, docente de la materia. El cual nos sirvió de ayuda para adaptarlo a nuestro tema para poder llevar acabo el presente proyecto.

El programa consiste en solicitar y guardar información en archivos sobre las rutas de los diferentes usuarios , así como también el poder modificar y consultar la información de las estructuras con las que contamos.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Marco teórico

Objetivo

Aprender las bases de la programación en C y C++ para desarrollar lógica con lo comprendido en el periodo primavera-verano 2022, conllevando así un dominio en los temas: listas (dobles y simples), archivos, estructuras y apuntadores.

Para con ello, poder implementarlo en programas que fácilmente pueda comprender un programador y un usuario, con la mejor funcionalidad.

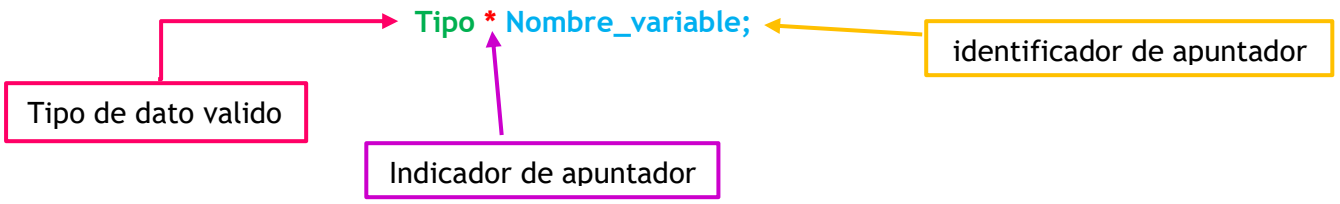
Fundamento teórico

A continuación, se presentan los temas de relevancia para una mayor comprensión al proyecto.

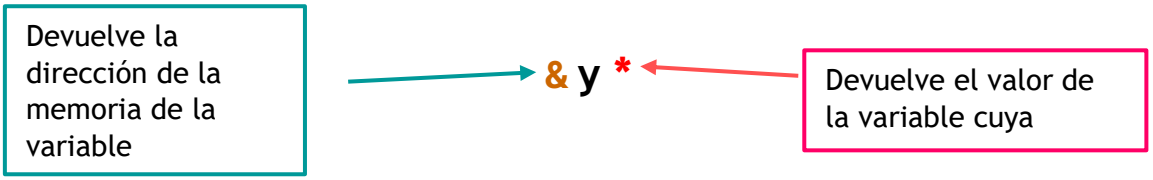
Apuntadores

Los apuntadores son variables que contienen direcciones de memoria en C y C++.

Se declaran de la siguiente manera:



Sus operadores son:



Su referenciación:

La referenciación es la obtención de la dirección de una variable, se hace a través del operador & aplicado a la variable a la cual se desea saber su dirección.

El valor puede variar durante cada ejecución del programa, debido a que el programa puede reservar distintos espacios de memoria durante cada ejecución.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Referenciación

```
int var1=6;
printf("\nLa direcci%cn de &var1 es: %d\n", 162, &var1);
```

Diferenciación:

Es la obtención del valor almacenado en el espacio de memoria donde apunta un apuntador, esto se hace a través del operador ‘*’ aplicado al apuntador que contiene la dirección del valor.

```
int var1=6;
int *pin= &var1;
printf("\nEl valor de var1 es: %d\n", *pin);
```

Estructuras/Uniones

Las estructuras toman parte de la memoria y se la reparten entre sus miembros. Cada miembro tiene reservado un espacio para él solo.

El tamaño total que ocupa una estructura en memoria es la suma del tamaño que ocupa cada uno de sus miembros.

Lo que diferencia una estructura de una unión es que, en una estructura, los miembros ocupan diferentes áreas de la memoria, pero en una unión, los miembros ocupan la misma área de memoria.

Inicialización de Estructuras

principal.cpp librerias.h inicializadas.h validaciones.h

```
#include "librerias.h"

int main(){
    Informacion dato;
    registrar(dato);
    imprimir(dato);
    return 0;
}
```

principal.cpp librerias.h inicializadas.h validaciones.h

```
void imprimirEstructura(){}

void registrar(Informacion &dato){
        dato.matricula=validaEntero("Matricula: ");
        printf("Nombre: ");
        validaCadena(dato.nombre);
        printf("Carrera: ");
        validaCadena(dato.carrera);
        dato.calif[0] = validaFlotante("Calif. 1: ");
        dato.calif[1] = validaFlotante("Calif 2: ");
        dato.calif[2] = validaFlotante("Calif 3: ");
        system("cls");
}

void imprimir(Informacion &dato){
    printf("%d\t", dato.matricula);
    printf("%s\t", dato.nombre);
    printf("%s\t", dato.carrera);
    printf("%.2f\t", dato.calif[0]);
    printf("%.2f\t",dato.calif[1]);
    printf("%.2f\t",dato.calif[2]);
    printf("%.2f\n", (dato.calif[0] + dato.calif[1] + dato.calif[2]) / 3);
}
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Estructuras Estáticas

principal.cpp librerias.h inicializadas.h funciones.h validaciones.h

```
#include "librerias.h"

int main(){
    registro cliente[3];
    registro_clientes(cliente);
    imprime_clientes(cliente);
    return 0;
}
```

principal.cpp librerias.h inicializadas.h funciones.h validaciones.h

```
#include<stdlib.h>//malloc, free
#include<conio.h>
#include <string.h>
#include <ctype.h>
#include <stdio.h>
#include <locale.h>

typedef struct{
    int dia;
    int mes;
    int anio;
}fecha;

typedef struct{
    char nombre[20];
    int edad;
    fecha f_nac;
}registro;
```

principal.cpp librerias.h inicializadas.h funciones.h validaciones.h valida.h

```
#include "librerias.h"

int main(){
    registro cliente2, *pCliente;
    pCliente = &cliente2;
    registro_clientes2(pCliente);
    imprime_clientes2(pCliente);
    return 0;
}
```

```
#include "librerias.h"

int main(){
    registro *pCliente;

    pCliente = (registro *)malloc(sizeof(registro));

    registro_clientes2(pCliente);
    imprime_clientes2(pCliente);
    free(pCliente);
    return 0;
}
```

principal.cpp [*] librerias.h inicializadas.h funciones.h validaciones.h

```
typedef struct{
    int dia;
    int mes;
    int anio;
}fecha;

typedef struct{
    char nombre[20];
    int edad;
    fecha f_nac;
}registro;
```

Estructuras Dinámicas

Archivos

Un archivo se define como un conjunto de registros, mismos que guardan una relación entre sí. Un registro es una colección de campos de información sobre una entidad particular.

La asociación de campos, se pueden hacer por medio de estructuras que vienen siendo la definición de un registro.

Funciones para manipular los archivos son: **fopen, fclose, feof, fprinf, fscanf.**

Apertura y cierre de los archivos:

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Se debe establecer un área en el buffer para almacenar la información temporalmente.
FILE *arch;

•**FILE**, definido en la librería stdio.h. Palabra reservada que indica el tipo de estructura a manejar

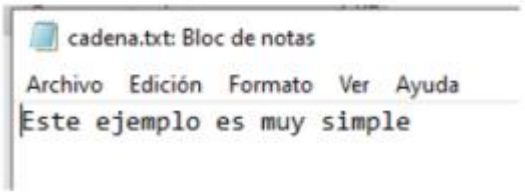
•***arch** , puntero que indica el principio del área en la que se almacena la información temporalmente.

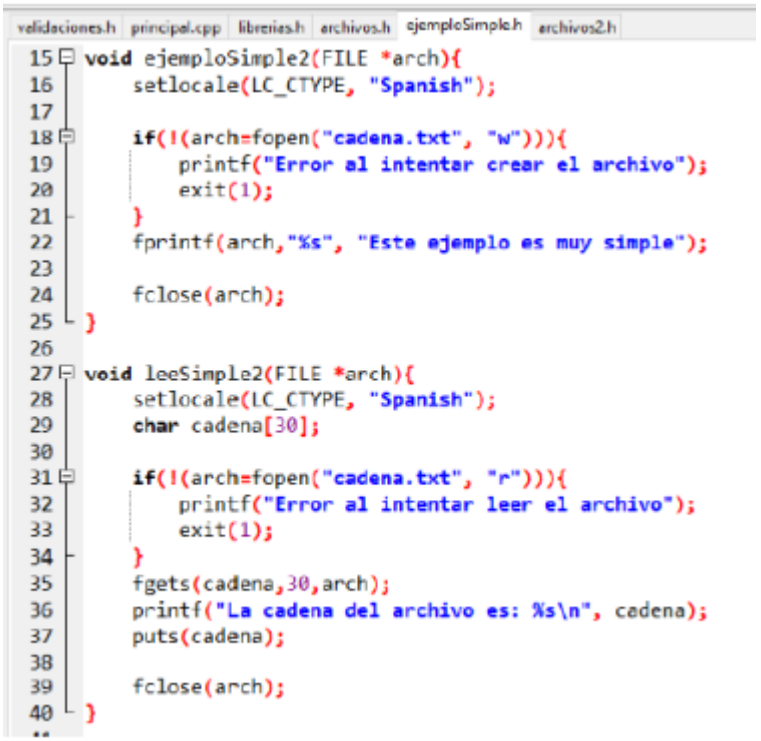
•Abrir archivo **arch=fopen**(nombre-archivo, tipo-archivo)

•**fopen**, devuelve un puntero al principio del área del buffer asociada al archivo. Devuelve un valor NULL si no se puede abrir el archivo o cuando no existe.

•**fclose**(arch), cierra el archivo.

Ejemplo Archivos





Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Listas

Las Listas son un tipo de estructura lineal y dinámica de datos. Lineal porque a cada elemento le puede seguir sólo otro elemento, dinámica porque se puede manejar la memoria de manera flexible, sin necesidad de reservar espacio con antelación.

Las listas ligadas son **colecciones de elementos llamados nodos**; el orden entre éstos se establece por medio de un tipo de datos denominado punteros o apuntadores, direcciones o referencias a otros nodos. Por lo tanto, siempre es importante distinguir entre un dato de tipo apuntador y el dato contenido en la celda al cual éste apunta.

Lista Simple Ligada/Enlazada

Una lista simplemente ligada constituye una colección de elementos llamados Nodos. El orden entre éstos se establece por medio de punteros; es decir, direcciones o referencias a otros nodos.

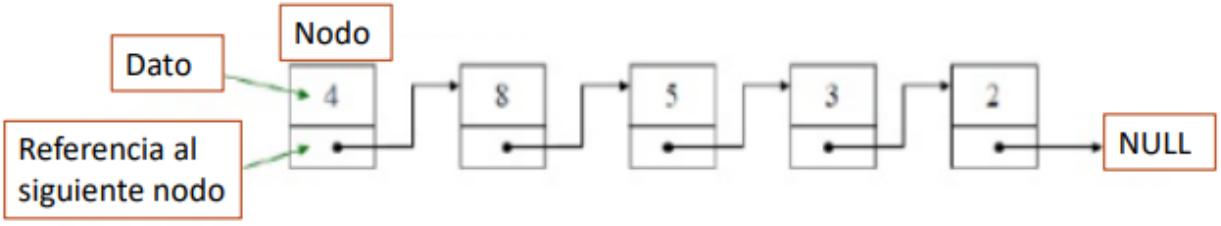
En general un nodo consta de **dos partes:**

▪Un **campo INFORMACIÓN**, que será del tipo de datos que se

requiera almacenar.

▪Un **campo LIGA**, de tipo puntero, que se utiliza para establecer la liga o el enlace con otro nodo de la lista. Si el nodo fuera el último de la lista, este campo

campo tendría como valor NULL.



Listas Dobles

Una lista doblemente ligada constituye una colección de elementos llamados Nodos. El orden entre éstos se establece por medio de punteros; es decir, direcciones o referencias a otros nodos. En general un nodo consta de dos partes:

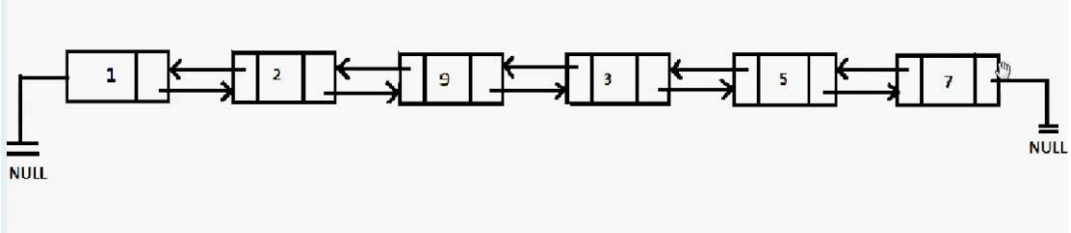
▪Un **campo INFORMACIÓN**, que será del tipo de datos que se requiera almacenar.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

▪Dos campos LIGA

***siguiente**, de tipo puntero, que se utiliza para establecer la liga o el enlace con otro nodo de la lista. Si el nodo fuera el último de la lista, este campo tendría como valor NULL.

***anterior**, de tipo puntero, que se utiliza para establecer la liga o el enlace con otro nodo de la lista. Si el nodo fuera el primero de la lista, este campo tendría como valor NULL



Funciones

Portada.h

Ilustración 1Función portada

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
1  void portada(){
2      int aux = 0;
3      FILE *archivo = fopen("portada.txt", "r");
4
5      for(int i=0; i<63; i++){
6          for(int j=0; j<75; j++){
7              fscanf(archivo, "%d\t", &aux);
8              SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), aux);
9              printf("%c", 177);
10         }
11         printf("\n");
12     }
13
14     fclose(archivo);
15 }
```

La función “portada” nos ayuda a imprimir nuestra imagen de rutas , mediante un archivo el cual contiene los colores a imprimir y además a través el uso de for llevamos a cabo la impresión de un símbolo especial, mediante el código ASCII.

Validación.h

Ilustración 2 Función validaId

```
funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

26  int validaId(const char mensaje[]){
27      struct Usuario *auxiliar = primLista;
28      int continuar=1,continuar2=0;;
29      int id=0,i;
30      bool bandera = true;
31
32      do{
33          continuar=0;
34          auxiliar = primLista;
35          id=validaEntero("Id Usuario: ");
36
37          while(auxiliar != NULL){
38              if(id==auxiliar->idUsuario){
39                  continuar = 1;
40                  break;
41              }else{
42                  continuar = 0;
43              }
44
45              auxiliar = auxiliar->sig;
46          }
47
48
49      }while(continuar ==1);
50
51      return id;
52  }
```

Esta función se invoca cada que queremos evitar que cada que el usuario meta un Id este se repita, así controlamos que se tenga un Id único.

Ilustración 3 Función validaFlotante

```
funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

1   float validaFlotante(const char mensaje[]){
2       int continuar=0;
3       float flotante=0;
4
5       do{
6           printf("\n%s",mensaje);
7           continuar = scanf("%f",&flotante);
8           fflush(stdin);
9       }while(continuar !=1);
10
11      return flotante;
12  }
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Esta función se invoca cada que queremos obtener un valor flotante del usuario, pero que este no ingrese otro valor que no permanezca a float.

Ilustración 4 Función validaEntero

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
14  int validaEntero(const char mensaje[]){
15      int continuar=0;
16      int entero=0;
17
18      do{
19          printf("\n%s",mensaje);
20          continuar = scanf("%i",&entero);
21          fflush(stdin);
22      }while(continuar !=1);//si es 0 el usuario no metio un entero
23      return entero;
24  }
25
```

Esta función se invoca cada que queremos obtener un valor entero del usuario, pero que este no ingrese otro valor que no permanezca a entero.

Ilustración 5 Función validaCadena

```
54  void validaCadena(const char mensaje[],const char cadena[]){
55      bool bandera = true;
56      while(bandera){
57          printf("\n%s", mensaje);
58          scanf("%[^\n]",cadena);
59          fflush(stdin);
60          if(strlen(cadena)>29){
61              continue;
62          }else{
63              for(int i = 0; i<strlen(cadena); i++){
64                  if(isalpha(cadena[i]) || cadena[i]==' '){
65                      if(i== (strlen(cadena)-1)){
66                          bandera=false;
67                      }
68                  }else{
69                      break;
70                  }
71              }
72          }
73      }
74  }
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Esta función se invoca cada que queremos obtener una cadena del usuario, pero que este no ingrese otro valor que no permanezca a los requeridos por dato solicitado. Esta función puede modificarse, cambiando la condición para solicitar diferentes caracteres.

Estructuras

Ilustración 6 Estructura admin

```
27  struct admin{
28      char nombre[20];
29      char contra[20];
30      struct admin *siguiente;
31      struct admin *anterior;
32  };
```

La estructura “admin” es la que contendrá las variables que se usaran para el registro de tipo Administrador.

Ilustración 7 Estructura usuario

```
34  struct usuario{
35      char nombre[20];
36      char contra[20];
37      struct usuario *siguienteb;
38      struct usuario *anteriorb;
39  };
```

La estructura “usuario” es la que contendrá las variables que se usaran para el registro de tipo Usuario.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 8 estructura MenuRutas

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
45  struct MenuRutas{
46      char nombreRuta[30];
47      char paradaRuta[30];
48      char horarioRuta[30];
49      char destinosRuta[30];
50      struct MenuRutas *siguienteRuta;
51  }*primi;
52
```

La estructura “MenuRutas” es la que contendrá los datos y variables que se usaran para el mostrar y registrar nuevas rutas, acción que solo podrá llevar acabo el administrador.

Ilustración 9 Estructura Rutas

```
54  struct Rutas{
55      int nodo;
56      int idRuta;
57      int estado;
58      int tomarBajar;
59      char destino[40];
60      int horaEntrada;
61      int horaSalida;
62      //string dia;
63      char parada[40];
64      Rutas *sigR;
65
66  }*primeroR, *ultimoR;
67
```

La estructura “Rutas” es la que contendrá las variables que se usaran para el mostrar y registrar nuevas Rutas por usuario.

Ilustración 10 Estructura Usuario

```
68  struct Usuario{
69      int idUsuario;
70      Rutas *lista;
71      Usuario *sig;
72  }*primLista,*ultLista;
73
```

La estructura “Usuario” es la que contendrá las variables que se usaran para el mostrar y registrar nuevas Listas de Listas.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Menu.h

Ilustración 11 Función menuRutas

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | [*] librerias.h

```
void menuRutas(MenuRutas *regis,FILE *file){
    HANDLE hd = GetStdHandle(STD_OUTPUT_HANDLE);

    if(!(file=fopen("rutas.txt","a+"))){
        printf("Error al intentar leer el archivo");
        exit(1);
    }

    while(!feof(file)){
        fscanf(file,"%s\t",regis->nombreRuta);
        fscanf(file,"%s\t",regis->paradaRuta);
        fscanf(file,"%s\t",regis->horarioRuta);
        fscanf(file,"%s\t",regis->destinosRuta);

        SetConsoleTextAttribute(hd, 2);
        printf("%s\t",regis->nombreRuta);
        SetConsoleTextAttribute(hd, 1);
        printf("%s\t",regis->paradaRuta);
        SetConsoleTextAttribute(hd, 13);
        printf("%s",regis->horarioRuta);
        SetConsoleTextAttribute(hd, 14);
        printf("%s\t\n",regis->destinosRuta);
        SetConsoleTextAttribute(hd, 15);
    }

    fclose(file);
}
```

Esta función es utilizada para leer de nuestro archivo “rutas.txt” los datos que corresponden a las especificaciones o detalles de cada ruta como su hora y destino.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 12 Función registroRutas

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | [*] librerias.h

```
29  void registroRutas(MenuRutas *regis, FILE *arch){
30
31      setlocale(LC_CTYPE,"Spanish");
32      SetConsoleCP(1252);
33      SetConsoleOutputCP(1252);//para acentos,
34      //apuntador de flecha, guarda direcciopnes de memoria
35      if(!(arch=fopen("rutas.txt","a+"))){
36              printf("Error al intentar crear el archivo");
37              exit(1);
38      }
39
40      validaCadena("Nombre Ruta: ", regis->nombreRuta);
41      validaCadena("Parada: ", regis->paradaRuta);
42      validaCadena("Horario: ", regis->horarioRuta);
43      validaCadena("Destino: ", regis->destinosRuta);
44
45      fprintf(arch,"\n%s\t",regis->nombreRuta);
46      fprintf(arch,"%s\t",regis->paradaRuta);
47      fprintf(arch,"%s\t",regis->horarioRuta);
48      fprintf(arch,"%s\t",regis->destinosRuta);
49
50
51      fclose(arch);
52
53  }
```

Esta función es utilizada para registrar/agregar a nuestro archivo “rutas.txt” nuevas caracteresticas de las rutas que queremos ingresar.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 13 Función menuPasajero

Con esta función "menuPasajero" podemos desplazarnos por las opciones dadas en el programa, específicamente para aquellos que se logearon como Usuarios. Esta función de menú se divide en todas las diversas opciones usando el mismo formato, y se usan diferentes para condicionar dependiendo a las opciones.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 14 Función menuAdmi

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | [*] librerias.h

```
125 void menuAdmi(){
126         FILE *file=NULL;
127         Rutas *primeroR=NULL;
128         int sigue,sigue2=2;
129         int opcP;
130 
131         system("cls");
132         cout<<"\n\n----- MENÚ -----\n";
133         cout<<"1.-Agregar cuenta Usuario/Asministrador";
134         cout<<"\n2.- Consultar Usuarios";
135         cout<<"\n3.- Consultar Administradores";
136         cout<<"\n4.- Consultar Rutas";
137         cout<<"\n5.- Registrar Rutas";
138         cout<<"\n8.- Salir";
139         printf("\n\nOpcion: ");
140         scanf("%d",&opcP);
141         switch(opcP){
142 
143             case 1:
144                 agregarCuenta();
145                 break;
146 
147             case 2:
148                 consultaUsuarios();
149                 break;
150 
151             case 3:
152                 consultaAdmi();
153                 break;
154 
155             case 4:
156                 MenuRutas *regis;
157                 regis=(MenuRutas *)malloc(sizeof(MenuRutas));
158                 menuRutas(regis,file);
159                 break;
160 
161             case 5:
162                 regis=(MenuRutas *)malloc(sizeof(MenuRutas));
163                 menuRutas(regis,file);
164                 regristroRutas(regis,file);
165                 break;
166 
167             case 6:
168                 break;
169 
170             case 7:
171                 break;
172 
173             case 8:
174                 system("cls");
175                 printf("HASTA LUEGO :)");
176                 break;
177         }
178 }
```

Con esta función “menuAdmi” podemos desplazarnos por las opciones dadas en el programa, específicamente para aquellos que se logearon como Administradores. Esta función de menú se divide en todas las diversas opciones usando el mismo formato, y se usan diferentes para condicionar dependiendo a las opciones.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

login.h

Ilustración 15 Función leerarchivo

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
1  void leerarchivo(){
2
3      if(!(file=fopen("admin.txt", "r"))){
4          printf("Error al intentar leer el archivo");
5          exit(1);
6      }
7      //leer del archivo, vaciar hacia un nodo
8      while(!feof(file)){
9
10         admin *nuevo = new admin;
11
12         fscanf(file,"%s\t", nuevo->nombre);
13         fscanf(file,"%s\n", nuevo->contra);
14
15         if(primero==NULL){
16             primero=nuevo;
17             primero->siguiente=NULL;
18             primero->anterior=NULL;
19             ultimo=primero;
20         }else{
21             ultimo->siguiente=nuevo;
22             nuevo->siguiente=NULL;
23             nuevo->anterior=ultimo;
24             ultimo=nuevo;
25         }
26
27     }
28     fclose(file);
29 }
```

Con esta función “leerarchivo” leemos a las personas que ya habían sido previamente registradas como Administradores.

Ilustración 16 Función leerarchivo2

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
31  void leerarchivo2(){
32
33      if(!(file=fopen("usuario.txt", "r"))){
34          printf("Error al intentar leer el archivo");
35          exit(1);
36      }
37      //leer del archivo, vaciar hacia un nodo
38      while(!feof(file)){
39
40          usuario *nuevo = new usuario;
41
42          fscanf(file,"%s\t", nuevo->nombre);
43          fscanf(file,"%s\n", nuevo->contra);
44
45          if(prim==NULL){
46              prim=nuevo;
47              prim->siguienteb=NULL;
48              prim->anteriorb=NULL;
49              ult=prim;
50          }else{
51              ult->siguienteb=nuevo;
52              nuevo->siguienteb=NULL;
53              nuevo->anteriorb=ult;
54              ult=nuevo;
55          }
56
57      }
58      fclose(file);
59  }
```

Con esta función “leerarchivo2” leemos a las personas que ya habían sido previamente registradas como Usuarios.

Ilustración 17 Función login

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
61  void login(){
62      int sigue=1,sigue2=1;
63      struct admin *auxiliar=NULL;
64      struct usuario *aux=NULL;
65      int opc,r;
66      char nom[20],con[20];
67
68  HANDLE hd = GetStdHandle(STD_OUTPUT_HANDLE);
69  SetConsoleTextAttribute(hd, 15);
70  printf("\t\t\t\t*****************************\n");
71      printf("\t\t\t\t*\t   BIENVENIDOS       *\n");
72      printf("\t\t\t\t*\t1.-ADMINISTRADOR     *\n ");
73      printf("\t\t\t\t*\t2.-USUARIO           *\n");
74      printf("\t\t\t\t*\t3.-CREAR CUENTA      *\n");
75      printf("\t\t\t\t*****************************\n");
76      opc = validaEntero("Opcion: ");
77      if(opc==1){
78          do{
79              printf("Nombre de administrador: \n");
80              scanf("%s",&nom);
81              for(auxiliar=primero;auxiliar!=NULL;auxiliar=auxiliar->siguiente){
82                  if(strcmp(nom,auxiliar->nombre)==0){
83                      r=1;
84                      break;
85                  }
86              }
87          }while(r==0);
88          r=0;
89          do{
```

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
89          do{
90              printf("Contrasena: \n");
91              scanf("%s",&con);
92                  if(strcmp(con,auxiliar->contra)==0){
93                      r=1;
94                  }
95          }while(r==0);
96          printf("\nBIENVENIDO...");
97          do{
98              Sleep(200);
99              menuAdmi();
100             sigue=validaEntero("Teclea 1 para escoger otra opcion del menu: ") ;
101         }while(sigue2==1);
102     }
103
104     if(opc==2){
105         do{
106             printf("Nombre de usuario: \n");
107             scanf("%s",&nom);
108             for(aux=prim;aux!=NULL;aux=aux->siguienteb){
109                 if(strcmp(nom,aux->nombre)==0){
110                     r=1;
111                     break;
112                 }
113             }
114         }while(r==0);
115         r=0;
116         do{
117             printf("Contrasena: \n");
```

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
117             printf("Contrasena: \n");
118             scanf("%s",&con);
119                 if(strcmp(con,aux->contra)==0){
120                     r=1;
121                 }
122         }while(r==0);
123         printf("\nBIENVENIDO...");
124         do{
125             Sleep(200);
126             menuPasajero();
127             sigue=validaEntero("Teclea 1 para escoger otra opcion del menu: ") ;
128         }while(sigue==1);
129     }
130
131
132     if(opc==3){
133         int opc;
134
135     printf("\n1.ADMINISTRADOR\n2.USUARIO");
136     scanf("%d",&opc);
137     if(opc==1){
138     int n=0;
139
140     if(!(file=fopen("admin.txt", "a+"))){              //Si el archivo no se pued
141         printf("Error al intentar leer el archivo");
142         exit(1);
143     }
144         admin *nuevo = new admin;
145
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
146         if(nuevo==NULL){
147             printf("No hay memoria disponible!!");
148         }
149
150         printf("\nNuevo usuario\n\n");
151         printf("\nUsuario: ");
152         scanf("%s",&nuevo->nombre);
153         printf("\nContrasena: ");
154         scanf("%s",&nuevo->contra);
155
156         fprintf(file,"\n%s\t", nuevo->nombre);
157         fprintf(file,"%s\n", nuevo->contra);
158
159         nuevo->siguiente=NULL;
160         nuevo->anterior=NULL;
161
162         if(primero==NULL){
163             primero=nuevo;
164             primero->siguiente=NULL;
165             primero->anterior=NULL;
166             ultimo=primero;
167         }else{
168             ultimo->siguiente=nuevo;
169             nuevo->siguiente=NULL;
170             nuevo->anterior=ultimo;
171             ultimo=nuevo;
172         }
173     fclose(file);
174     }
```

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
176     if(opc==2){
177
178     int n=0;
179
180     if(!(file=fopen("usuario.txt", "a+"))){
181         printf("Error al intentar leer el archivo");
182         exit(1);
183     }
184         usuario *nuevo = new usuario;
185
186         if(nuevo==NULL){
187             printf("No hay memoria disponible!!");
188         }
189
190         printf("\nNuevo usuario\n\n");
191         printf("\nUsuario: ");
192         scanf("%s",&nuevo->nombre);
193         printf("\nContrasena: ");
194         scanf("%s",&nuevo->contra);
195
196         fprintf(file,"\n%s\t", nuevo->nombre);
197         fprintf(file,"%s\n", nuevo->contra);
198
199         nuevo->siguienteb=NULL;
200         nuevo->anteriorb=NULL;
201
202         if(prim==NULL){
203             prim=nuevo;
204             prim->siguienteb=NULL;
```

```
201
202         if(prim==NULL){
203             prim=nuevo;
204             prim->siguienteb=NULL;
205             prim->anteriorb=NULL;
206             ult=prim;
207         }else{
208             ult->siguienteb=nuevo;
209             nuevo->siguienteb=NULL;
210             nuevo->anteriorb=ult;
211             ult=nuevo;
212         }
213     fclose(file);
214     }
215     system("cls");
216     login();
217
218     }
219 }
```

La función “login” es utilizada para registrar a cualquier usuario que se quiera registrar a nuestro programa, dividiéndose en dos secciones administrador y usuario. Además si

este ya se encontraba registrado solo verificamos si el nombre de la persona y su contraseña coinciden en la sección en la que se registro.

funciones.h

Ilustración 18 Función archivoEscrituraRutas

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
1  void archivoEscrituraRutas(){
2      Rutas *aux = primeroR;
3      file=fopen("Rutas.xls","w");
4      if(file==NULL){
5          printf("No se encontro el archivo");
6          exit(1);
7      }
8      else{
9          while(aux!=NULL){
10             fprintf(file,"%d\t",&aux->idRuta);
11             fprintf(file,"%d\t",&aux->estado);
12             fprintf(file,"%s\t",aux->parada);
13             fprintf(file,"%s\t",aux->destino);
14             fprintf(file,"%d\t",&aux->horaEntrada);
15             fprintf(file,"%d\t",&aux->horaSalida);
16             fprintf(file,"%d\t",&aux->tomarBajar);
17             aux=aux->sigR;
18         }
19         fclose(file);
20     }
21 }
```

La función “archivoEscrituraRutas” sirve para escribir en el archivo las nuevas actualizaciones o registros que el usuario hace.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 19 Función archivoLecturaRutas

funciones.h | login.h | menus.h | portada.h | principal.cpp | validacion.h | librerias.h

```
23  void archivoLecturaRutas(){
24      Rutas *lecturaR;
25      file2=fopen("Rutas.xls","r");
26      if(file2==NULL){
27          printf("ARCHIVO NO ENCONTRADO");
28          exit(1);
29      }
30      else{
31          while(!feof(file2)){
32              fscanf(file2,"%d\t",&lecturaR->idRuta);
33              fscanf(file2,"%d\t",&lecturaR->estado);
34              fscanf(file2,"%s\t",&lecturaR->parada);
35              fscanf(file2,"%s\t",&lecturaR->destino);
36              fscanf(file2,"%d\t",&lecturaR->horaEntrada);
37              fscanf(file2,"%d\n",&lecturaR->horaSalida);
38              fscanf(file2,"%d\n",&lecturaR->tomarBajar);
39
40              Rutas *nuevo = new Rutas;
41
42              nuevo->idRuta = lecturaR->idRuta;
43              nuevo->estado = lecturaR->estado;
44              strcpy(nuevo->parada, lecturaR->parada);
45              strcpy(nuevo->destino, lecturaR->destino);
46              nuevo->horaEntrada = lecturaR->horaEntrada;
47              nuevo->horaSalida = lecturaR->horaSalida;
48              nuevo->horaSalida = lecturaR->tomarBajar;
49              nuevo->sigR = NULL;
50
51              if(primeroR == NULL){
52                  primeroR = nuevo;
53                  ultimoR = nuevo;
54              }
55              else{
56                  ultimoR->sigR = nuevo;
57                  ultimoR = nuevo;
58              }
59          }
60          fclose(file2);
61      }
62  }
```

La función “archivoEscrituraRutas” sirve para escribir en el archivo las nuevas actualizaciones o registros que el lee o escanea propiamente nuestro archivo.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 20 Función registroRuta

```
64  Rutas *registrarRuta() {
65      Rutas *nuevo = new Rutas;
66      nuevo->idRuta = validaEntero("Ruta: ");
67      nuevo->estado = validaEntero("Estado(ACTIVO 1 - INACTIVO - 0): ");//Uno es ACTIVO , dos INACTIVO
68      if(nuevo->estado==1 || nuevo->estado==0){
69          printf("Dato Correcto!\n");
70      }else{
71          printf("Estado NO valido..Solo 1 o 0");
72          nuevo->estado = validaEntero("Estado: ");
73      }
74      cout<<"Parada: ";
75      cin>>nuevo->parada;
76      cout<<"Destino: ";
77      cin>>nuevo->destino;
78      /*cout<<"Hora de Entrada: ";
79      cin>>nuevo->horaEntrada;*/
80      nuevo->horaEntrada = validaEntero("Hora de Entrada: ");
81      nuevo->horaSalida = validaEntero("Hora de Salida: ");
82      /*cout<<"Hora de Salida: ";
83      cin>>nuevo->horaSalida;*/
84      nuevo->tomarBajar = validaEntero("Tomar Autobus=1 - Bajar Autobus=2 - Ninguno(Por el momento) - 0): ");
85      if(nuevo->tomarBajar==1 || nuevo->tomarBajar==2 || nuevo->tomarBajar==0){
86          printf("Dato Correcto!\n");
87      }else{
88          printf("Dato No valido..Solo 1,2 o 0");
89          nuevo->tomarBajar=validaEntero("Tomar Autobus=1 - Bajar Autobus=2 - Ninguno(Por el momento) - 0): ");
90      }
91      nuevo->sigR = NULL;
92      return nuevo;
93  }
94
```

La función "registroRuta" sirve para que el usuario pueda registrar o añadir a su lista alguna de las rutas que aparecen por defecto.

Ilustración 21 Función eliminaEstado

```
95  void eliminaEstado(){
96      Rutas *aux = primeroR;
97      int id=0, veri;//true
98      id=validaEntero("Ruta: ");
99
100     veri=0;
101         while(aux != NULL){
102             if(id==aux->idRuta){
103                 cout<<"Cambiando Su estado.. ";
104                 aux->estado=0;
105                 veri = 1;
106                 break;
107             }
108             aux = aux->sigR;
109         }
110         if(veri == 0){
111             printf("\nRUTA NO REGISTRADA\n");
112         }
113
114 }
115
```

La función "eliminaEstado" sirve para cambiar el estado de cada ruta. Lo que hace es pasar de un estado Activo = 1 a inactivo =0-

Ilustración 22 Función modifEstado

```
116 void modifEstado(){
117     Rutas *aux = primeroR;
118     int id=0, veri;//true
119     id=validaEntero("Ruta: ");
120 
121     veri=0;
122         while(aux != NULL){
123             if(id==aux->idRuta){
124                 cout<<"Ingrese Su Nuevo Estado: ";
125                 cin>>aux->estado;
126                 veri = 1;
127                 break;
128             }
129             aux = aux->sigR;
130         }
131         if(veri == 0){
132             printf("\nRUTA NO REGISTRADA\n");
133         }
134 
135 }
```

```
137 void modifHorarioEntrada(){
138     Rutas *aux = primeroR;
139     int id=0, veri;//true
140     id=validaEntero("Ruta: ");
141 
142     veri=0;
143         while(aux != NULL){
144             if(id==aux->idRuta){
145                 cout<<"Ingrese Su Nueva Hora de entrada: ";
146                 cin>>aux->horaEntrada;
147                 veri = 1;
148                 break;
149             }
150             aux = aux->sigR;
151         }
152         if(veri == 0){
153             printf("\nRUTA NO REGISTRADA\n");
154         }
155 
156 }
```

Las funciones “modifEstado” y “modifHorarioEntrada” tiene la misma funcionalidad la cual es modificar o cambiar un dato especifico de nuestro registro de rutas. Solo dependerá del dato a cambiar.

Ilustración 23 Función newRuta

```
179 void newRuta() {
180     Rutas *nuevo = registrarRuta();
181     if(primeroR == NULL) {
182         primeroR = nuevo;
183         ultimoR = nuevo;
184     } else {
185         ultimoR->sigR = nuevo;
186         ultimoR = nuevo;
187     }
188 }
```

La función “newRuta” tiene la función de añadir un nuevo registro de alguna Ruta.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 24 Función agregarLista

```
190 void agregarLista(){
191     Usuario *nuevo = new Usuario;
192     int sigue = 1;
193
194     primeroR = NULL;
195     ultimoR = NULL;
196
197     while(sigue==1){
198         newRuta();
199         sigue = validaEntero("Tecla 1 para agregar otra Ruta: ");
200     }
201     nuevo->sig = NULL;
202     nuevo->lista = primeroR;
203
204     if(primLista==NULL){
205         primLista = nuevo;
206         ultLista = nuevo;
207     }else{
208         ultLista->sig=nuevo;
209         ultLista = nuevo;
210     }
211 }
```

La función “agregarLista” sirve para agregar una Lista Doble a nuestro programa, a través de un nuevo usuario.

Ilustración 25 Función consultarRutasInfo

```
452 void consultaUsuarios(){
453     admin *aux=primero;
454     cout<<"USUARIOS..\n"<<setw(0);
455     cout<<"Nombres"<<setw(15);
456     cout<<"Contraseñas"<<endl;
457     while(aux!=NULL){
458         cout<<aux->nombre<<setw(15);
459         cout<<aux->contra<<endl;
460         aux=aux->siguiente;
461     }
462 }
463
464 void consultaAdmi(){
465     usuario *aux=prim;
466     cout<<"ADMINISTRADORES..\n"<<setw(0);
467     cout<<"Nombres"<<setw(15);
468     cout<<"Contraseñas"<<endl;
469     while(aux!=NULL){
470         cout<<aux->nombre<<setw(15);
471         cout<<aux->contra<<endl;
472         aux=aux->siguienteb;
473     }
474 }
475
```

```
214 void consultarRutasInfo(Rutas *primeroR){
215     Rutas *aux=primeroR;
216     cout<<"\nMostrando Sus Registros"<<endl;
217     cout<<"Ubicacion"<<setw(20);
218     cout<<"Ruta"<<setw(20);
219     cout<<"Estado"<<setw(20);
220     cout<<"Parada"<<setw(20);
221     cout<<"Destino"<<setw(20);
222     cout<<"Hora entrada"<<setw(20);
223     cout<<"Hora salida"<<setw(20);
224     cout<<"Tomar/Bajar Autobus"<<setw(20);
225     cout<<"Siguiente"<<endl;
226     while(aux!=NULL){
227         cout<<aux<<setw(20);
228         cout<<aux->idRuta<<setw(20);
229         cout<<aux->estado<<setw(20);
230         cout<<aux->parada<<setw(20);
231         cout<<aux->destino<<setw(20);
232         cout<<aux->horaEntrada<<setw(20);
233         cout<<aux->horaSalida<<setw(20);
234         cout<<aux->tomarBajar<<endl;
235         cout<<aux->sigR<<endl;
236         aux=aux->sigR;
237     }
238 }
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

```
281 void consultahour(){
282     Rutas *aux=primeroR;
283     cout<<"Ruta.."<<setw(20);
284     cout<<"Hora entrada"<<setw(20);
285     cout<<"Hora salida"<<endl;
286     while(aux!=NULL){
287         cout<<aux->idRuta<<setw(20);
288         cout<<aux->horaEntrada<<setw(20);
289         cout<<aux->horaSalida<<endl;
290         aux=aux->sigR;
291     }
292 }
```

```
313 void consultaDestino(){
314     Rutas *aux=primeroR;
315     cout<<"Ruta.."<<setw(20);
316     cout<<"Destino"<<endl;
317     while(aux!=NULL){
318         cout<<aux->idRuta<<setw(20);
319         cout<<aux->destino<<endl;
320         aux=aux->sigR;
321     }
322 }
323
324 void consultaParadas(){
325     Rutas *aux=primeroR;
326     cout<<"Ruta.."<<setw(20);
327     cout<<"Parada"<<endl;
328     while(aux!=NULL){
329         cout<<aux->idRuta<<setw(20);
330         cout<<aux->parada<<endl;
331         aux=aux->sigR;
332     }
333 }
```

Las funciones de “consultar” tiene la funcionalidad de imprimir la información a la que queremos acceder, cada una de ellas tiene el mismo propósito solo cambiará la información que la contiene.

Ilustración 26 Función mostrarListas

```
240 void mostrarListas(){
241     system("cls");
242     struct Usuario *aux = primLista;
243     cout<<"Ubicacion Lista"<<setw(20);
244     cout<<"Lista(Primer nodo)"<<endl;
245     while(aux!=NULL){
246         cout<<aux<<setw(20);
247         cout<<aux->lista<<endl;
248         aux = aux->sig;
249     }
250     cout<<endl<<"Primer LISTA: "<<primLista;
251     cout<<endl<<"Ultima LISTA: "<<ultLista;
252     aux = primLista;
253     while(aux!=NULL){
254         consultarRutasInfo(aux->lista);
255         aux = aux->sig;
256     }
257 }
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

La función “mostrarListas” es aquella que nos ayuda a imprimir todos los registros que se han hecho es decir todas las listas de listas registradas.

Ilustración 27 Función checarRuta

```
258 void checarRuta(){
259     Rutas *aux=primeroR;
260     cout<<"\nMostrando sus Registros"<<endl;
261     cout<<"Ruta"<<setw(20);
262     cout<<"Estado"<<setw(20);
263     cout<<"Parada"<<setw(20);
264     cout<<"Destino"<<setw(20);
265     cout<<"Hora De Entrada"<<setw(20);
266     cout<<"Hora De Salida"<<setw(20);
267     cout<<"Tomar/Bajar Autobus"<<endl;
268     while(aux!=NULL){
269         cout<<aux->idRuta<<setw(20);
270         cout<<aux->estado<<setw(20);
271         cout<<aux->parada<<setw(20);
272         cout<<aux->destino<<setw(20);
273         cout<<aux->horaEntrada<<setw(20);
274         cout<<aux->horaSalida<<setw(20);
275         cout<<aux->tomarBajar<<endl;
276         aux=aux->sigR;
277     }
278     getch();
279 }
```

La función “checarRuta” imprime los registros generados por cada usuario, es decir, despliega la información de todo aquello que se haya registrado.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

Ilustración 28 Función agregarCuenta

```
void agregarCuenta(){
        int opc;

    printf("\n1.ADMINISTRADOR\n2.USUARIO");
    scanf("%d",&opc);
    if(opc==1){
    int n=0;

    if(!(file=fopen("admin.txt", "a+"))){
        printf("Error al intentar leer el archivo");
        exit(1);
    }
        admin *nuevo = new admin;

        if(nuevo==NULL){
            printf("No hay memoria disponible!!");
        }

        printf("\nNuevo usuario\n\n");
        printf("\nUsuario: ");
        scanf("%s",&nuevo->nombre);
        printf("\nContrasena: ");
        scanf("%s",&nuevo->contra);

        fprintf(file,"\n%s\t", nuevo->nombre);
        fprintf(file,"%s\n", nuevo->contra);

        nuevo->siguiente=NULL;
        nuevo->anterior=NULL;

        if(primero==NULL){
            primero=nuevo;
            primero->siguiente=NULL;
            primero->anterior=NULL;
            ultimo=primero;
        }else{
```

```
            ultimo=primero;
        }else{
            ultimo->siguiente=nuevo;
            nuevo->siguiente=NULL;
            nuevo->anterior=ultimo;
            ultimo=nuevo;
        }
    fclose(file);
    }

    if(opc==2){

    int n=0;

    if(!(file=fopen("usuario.txt", "a+"))){
        printf("Error al intentar leer el archivo");
        exit(1);
    }
        usuario *nuevo = new usuario;

        if(nuevo==NULL){
            printf("No hay memoria disponible!!");
        }

        printf("\nNuevo usuario\n\n");
        printf("\nUsuario: ");
        scanf("%s",&nuevo->nombre);
        printf("\nContrasena: ");
        scanf("%s",&nuevo->contra);

        fprintf(file,"\n%s\t", nuevo->nombre);
        fprintf(file,"%s\n", nuevo->contra);

        nuevo->siguienteb=NULL;
        nuevo->anteriorb=NULL;

        if(prim==NULL){
            prim=nuevo;
```

```
        nuevo->siguienteb=NULL;
        nuevo->anteriorb=NULL;

        if(prim==NULL){
            prim=nuevo;
            prim->siguienteb=NULL;
            prim->anteriorb=NULL;
            ult=prim;
        }else{
            ult->siguienteb=nuevo;
            nuevo->siguienteb=NULL;
            nuevo->anteriorb=ult;
            ult=nuevo;
        }
    fclose(file);
    }
    system("cls");
}
```

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx

La función "agregarCuenta" imprime los registros generados por cada usuario, es decir, despliega la información de todo aquello que se haya registrado.

Cortés Beltrán Carol Elizabeth 177203@upslp.edu.mx
Montelongo Martínez Laura Ivon 177291@upslp.edu.mx