```
#include <iostream>
#include <string>
#include <stdlib.h>
#include <iomanip>
#include <stdio.h>
#include <string.h>

#define BLACK_COLOR "\033[1;30m"
#define RED_COLOR "\033[1;31m"
#define GREEN_COLOR "\033[1;32m"
#define YELLOW_COLOR "\033[1;33m"
#define BLUE_COLOR "\033[1;34m"
#define PURPLE_COLOR "\033[1;35m"
#define CYAN_COLOR "\033[1;36m"
#define WHITE_COLOR "\033[1;39m"
#define RESTORE_DEFAULT_COLOR "\033[0m"

using namespace std;
/*************************************************************
 ** Definición de sus struct del Tipo de Dato Foto **
 *************************************************************/
struct Foto{
        string ruta;
        string tipo;
        int tamanio;

};
/*************************************************************
 ** Definición de sus struct del Tipo de Dato Usuario **
 *************************************************************/
struct Usuario{
        string login;
        string nombre;
        string apellido;
        string perfilusuario;
        Foto *v_foto;
        int dim_v_foto;
        int totalfotosusuario;
};
/*************************************************************
 ** Definición de sus struct del Tipo de tabla Usuario **
 *************************************************************/
struct TablaUsuarios{
        Usuario **tabla;
        int totaltuplas;
};
/*************************************************************
 ** Definición de los prototipos
 *************************************************************/
void setruta(Foto *f, string ruta);
void settipo(Foto *f, string tipo);
void settamanio(Foto *f, int tamanio);
```

```
string getruta(Foto *f);
string gettipo(Foto *f);
int gettamanio(Foto *f);
void setlogin(Usuario *u, string login);
void setnombre(Usuario *u, string nombre);
void setapellido(Usuario *u, string apellido);
void setaperfilusuario(Usuario *u, string perfilusuario);
string getlogin(Usuario *u);
string getnombre(Usuario *u);
string getapellido(Usuario *u);
string getaperfilusuario(Usuario *u);
Foto* getv_foto(Usuario *u);
int getdim_v_foto(Usuario *u);
int gettotalfotosusuario(Usuario *u);
void settotaltuplas(TablaUsuarios *tablausuario, int totaltuplas);
int gettotaltuplas(TablaUsuarios *tablausuario);
void addUsuario (TablaUsuarios *tablausuario, Usuario *u);
TablaUsuarios* InicializarTabla ();
Usuario* crearusuario (string login, string nombre, string apellido, string perfilusuario , int
dim_v_foto,int totalfotosusuario);
void imprimirtabla(TablaUsuarios *tabla_u);
void imprimirusuario(Usuario *u);
void anadirusuarios(TablaUsuarios *tablausuario,string login,string nombre,string apellido,string
perfilusuario, int dimensionfoto, int totalfotos);
Usuario** resize(Usuario **u, int util_v, int &dim_v_actual, int dim_nueva);
Foto* resizeFotos(Foto *u, int util_v, int &dim_v_actual, int dim_nueva);
Foto* anadir_foto_usuario(TablaUsuarios *ta, Usuario *usuario_Crear,string ruta,string tipo, int
tamanio);
void imprimir_una_foto(Foto *f);
void imprimir_las_fotos(Usuario *u);
void borrarusuarios(TablaUsuarios *u, string login);
void buscarusuario_imprimir_foto(TablaUsuarios *u, string login);
bool comprobar_si_existe(TablaUsuarios *u, string login);
Usuario* comprobar_si_existe_foto(TablaUsuarios *u);
void datos_crear_usuario(TablaUsuarios *tablausuario, string nombre,string apellido,string
usuario,string tipo,string login, int util_foto, int dim_fotos);
void datos_crear_foto(TablaUsuarios *tu);
void borrar_tabla(TablaUsuarios *ta);
void burbuja(TablaUsuarios *ta);
void burbuja_total_fotos(TablaUsuarios *ta);
void menu_en_pantalla();
/*************************************************************
 ** Definición de los set y get del Tipo de Dato foto
 *************************************************************/

void setruta(Foto *f, string ruta){
        f -> ruta = ruta;
}
void settipo(Foto *f, string tipo){
        f -> tipo = tipo;
}
void settamanio(Foto *f, int tamanioanio){
```

```
	f -> tamanio = tamanioanio;
}
string getruta(Foto *f){
	return f -> ruta;
}
string gettipo(Foto *f){
	return f -> tipo;
}
int gettamanio(Foto *f){
	return f -> tamanio;
}
/*************************************************************
 ** Definición de sus set get del Tipo de Dato Usuario
 *************************************************************/
void setlogin(Usuario *u, string login){
	u -> login = login;
}
void setnombre(Usuario *u, string nombre){
	u -> nombre = nombre;
}
void setapellido(Usuario *u, string apellido){
	u -> apellido = apellido ;
}
void setaperfilusuario(Usuario *u, string perfilusuario){
	u -> perfilusuario = perfilusuario;
}
void setv_foto(Usuario *u, Foto *v_foto){
	u -> v_foto = v_foto;
}
void setdim_v_foto(Usuario *u, int dim_v_foto){
	u -> dim_v_foto = dim_v_foto;
}
void settotalfotosusuario(Usuario *u, int totalfotosusuario){
	u -> totalfotosusuario = totalfotosusuario;
}
string getlogin(Usuario *u){
	return u -> login;
}
string getnombre(Usuario *u){
	return u -> nombre;
}
string getapellido(Usuario *u){
	return u -> apellido;
}
string getaperfilusuario(Usuario *u){
	return u -> perfilusuario;
}
Foto* getv_foto(Usuario *u){
	return  u -> v_foto;
}
int getdim_v_foto(Usuario *u){
	return u -> dim_v_foto;
```

```
}
int gettotalfotosusuario(Usuario *u){
        return u -> totalfotosusuario;
}
/*************************************************************
 ** Definición de sus Prototipos del Tipo de tabla
 *************************************************************/
void settotaltuplas(TablaUsuarios *tablausuario, int totaltuplas){
        tablausuario -> totaltuplas = totaltuplas;
}
int gettotaltuplas(TablaUsuarios *tablausuario){
        return tablausuario -> totaltuplas;
}
/*
*@info vamos añadir a tabla todos los usuarios.
*@info le añadimos un usuario a totaltuplas. es un puntero de tipo usuario.
*@param u es un usuario que lo vamos a guardar en la tabla.
*@param tablausuario es la tabla donde vamos a guardar todos los usuarios.
*@return no de vuelve nada ya que es de tipo void.
*/

void addUsuario (TablaUsuarios *tablausuario, Usuario *u){
        //vamos añadir un usuario a la tabla de usuarios. acedo al struc y el struct tiene el totaltuplas.
        tablausuario -> tabla[tablausuario -> totaltuplas] = u ;
        tablausuario -> totaltuplas++;
}
/**
 * @brief Modulo que aumenta o disminuye la dimensión del vector de punteros de foto, cuya dirección se le pasa por copia.
 * @info Si las dim_nueva < dim_v_actual
 * @pre util_v <= dim_v_actual
 * @post util_v y dim_v_actual se modifican apropiadamente en el interior, por lo tanto fuera se veran también afectadas
 * @return La dirección de memoria del vector nuevo que ya tiene la dimensión nueva, y copiados los elementos del vector antiguo.
 */
Foto* resizeFotos(Foto *u, int util_v, int &dim_v_actual, int dim_nueva){
        Foto *nuevo = 0;
   nuevo = new Foto[dim_nueva];
   if (nuevo == 0){
      cerr << "Error. No hay memoria suficiente. Se abortará la ejecución" << endl;
      exit(-1);
   }

   //1.1) Discusión acerca de hasta donde copio en el escenario de disminución
   if (util_v > dim_nueva) //recorto
      util_v = dim_nueva;


   //2º Copio las componentes del vector antiguo en el nuevo
   for(int i =0; i < util_v; i++)
      nuevo[i] = u[i];
```

```
    //3º Libero la memoria del antiguo vector
    delete [] u;
    //4º Devuelvo el puntero del nuevo vector, para que se guarde en la variable del módulo llamante y cambio la dimensión del vector actual
    dim_v_actual = dim_nueva;

    return nuevo;
}
/*
*@info vamos a crear el usuario con un puntero de tipo Usuario.
*@pre le pasamos todos los datos ya introducidos en otro modulo.
*@param le pasamos de tipo string el login, nombre, apellido y el perfilusuario dim_v_foto(util), totalfotosusuario(dimension total del vector)
*@param creo u usuario de de tipo diinamico
*@param creo f que es vector de punteros,  el vector de fotos que se las voy añadir al usuario
*@post devuelve un usuario el usuario con su vector de fotos dinamico y todos los datos añadidos
*@return devuelvo u = usuario dinamico.
*/
Usuario* crearusuario (string login, string nombre, string apellido, string perfilusuario , int dim_v_foto,int totalfotosusuario){
        Usuario *u = new Usuario;
        Foto *f = new Foto [100];
        if (u == 0)
        {
                cerr << " Error. No hay memoria suficiente. Se abortará la ejecución" << endl;
      exit(-1);
    }
    u-> v_foto = f;
    setlogin (u, login);
    setnombre(u, nombre);
    setapellido(u, apellido);
    setaperfilusuario(u, perfilusuario);
    setv_foto(u, f);
    setdim_v_foto(u, totalfotosusuario);
    settotalfotosusuario(u, totalfotosusuario);

    return u;
}
/**
 * @brief modulo que inicializa una foto normal como dinamica.
 * @info no le paso nada ya que lo voy a inicializar en la funcion
 * @post devuelve una foto dinamica
 * @return devuelvo f que es una foto dinamica
 */
Foto* iniciar_foto(){
        Foto *f = new Foto;

        if (f == 0){
      cerr << " Error. No hay memoria suficiente. Se abortará la ejecución" << endl;
      exit(-1);
    }
    return f;
```

```
}
/**
 * @brief Módulo que inserta la foto, en el Vector Dinámico de fotos v.
 * @info En los escenarios necesarios realiza una llamada al módulo resize para que l nueva foto p
pueda introducirse sin problemas y sea transparente para el programado que utilice este módulo
 * @param aumento la util en una posicion cuando cree una foto
 */
void insertarFotoVector(Foto* v, int &util_actual, int &dim_actual, Foto *p){
   cout << " Insertar persona: " << endl;

   v[util_actual] = *p;
   util_actual++;

}
/**
 * @brief modulo que añade a una foto los datos ruta, tipo y tamaño
 * @param f es el puntero de fotos donde vamos a guardar la ruta, tipo y tamaño de la foto
 * @pre paso una foto ya creada como dinamica y los datos introducidos por el usuario
 * @post no devuelvo nada ya que lo voy añadir con los set en este modulo
 */
void introducir_una_foto (Foto *f, string ruta, string tipo,int tamanio){
        setruta(f,ruta);
        settipo(f,tipo);
        settamanio(f,tamanio);
}
/*
* @info vamos a introducir las fotos
*@pre le pasamos ya el usuario que exista en el programa. y usuaremos el resize ya previamente
creado
*@param le vamos a sumar una posicion a la dimension de foto(util) dim_v_foto
*@param hago un settotalfotos para que la util sea igual de grande que la dimension del vector
*@post le abremos añadido una posicion mas al vector de fotos
*/
void introducirfoto(Usuario *u ,string login, string ruta, string tipo,int tamanio, Foto *f){

        u->v_foto = resizeFotos(u->v_foto, getdim_v_foto(u), u->dim_v_foto, u-
>totalfotosusuario+1);
        settotalfotosusuario(u, getdim_v_foto(u));
        u->v_foto[getdim_v_foto(u)-1] = *f;

}

/**
 * @brief Modulo que aumenta o disminuye la dimensión del vector de punteros a usuario, cuya
dirección se le pasa por copia.
 * @info Si las dim_nueva < dim_v_actual
 * @pre util_v <= dim_v_actual
 * @post util_v y dim_v_actual se modifican apropiadamente en el interior, por lo tanto fuera se
veran también afectadas
 * @return La dirección de memoria del vector nuevo que ya tiene la dimensión nueva, y copiados
los elementos del vector antiguo.
 */
```

```
Usuario** resize(Usuario **u, int util_v, int &dim_v_actual, int dim_nueva){
        Usuario **nuevo = 0;
   nuevo = new Usuario*[dim_nueva];
   if (nuevo == 0){
      cerr << "Error. No hay memoria suficiente. Se abortará la ejecución" << endl;
      exit(-1);
   }

   //1.1) Discusión acerca de hasta donde copio en el escenario de disminución
   if (util_v > dim_nueva) //recorto
      util_v = dim_nueva;


   //2º Copio las componentes del vector antiguo en el nuevo
   for(int i =0; i < util_v; i++)
      nuevo[i] = u[i];
   //3º Libero la memoria del antiguo vector
   delete [] u;
   //4º Devuelvo el puntero del nuevo vector, para que se guarde en la variable del módulo llamante
y cambio la dimensión del vector actual
   dim_v_actual = dim_nueva;
   return nuevo;
}
/*
* @info vamos a introducir un usuario
*@pre le pasamos la tabla que exista en el programa. y usuaremos el resize ya previamente creado
*@param le vamos a sumar una posicion a la dimension de totaltuplas(util)
*@param hago un settotalfotos para que la util sea igual de grande que la dimension del vector
*@post le abremos añadido una posicion mas al vector de usuarios tabla
*/
void anadirusuarios(TablaUsuarios *tablausuario,string login,string nombre,string apellido,string
perfilusuario, int dimensionfoto, int totalfotos){

        tablausuario -> tabla = resize (tablausuario -> tabla, tablausuario -> totaltuplas,tablausuario
-> totaltuplas,tablausuario -> totaltuplas+1);
        Usuario *u = crearusuario(login, nombre, apellido,perfilusuario, dimensionfoto, totalfotos);
        tablausuario->tabla[tablausuario->totaltuplas-1] = u;

}

/**
 * @brief Cambia la dimensión del vector a una nueva dim_nueva
 *  1) Creo un vector nuevo con la nueva dimension donde vamos a guardar la tabla de usuarios de
doble puntero
 *  2) Copio el contenido del vector que me pasan, manualmente ya que va a ser una tabla
predefinida con fotos
 *      @return Devuelvo la nueva tabla con todos los usuarios creados he introducidos.
 */
TablaUsuarios* InicializarTabla() {
        TablaUsuarios *tu = new TablaUsuarios;
        settotaltuplas(tu, 7); //totaltuplas es la util que estamos usando de usuarios hasta el momento
```

```
	Usuario **doble_puntero = new Usuario*[7];
	Foto *f = 0;

	f = iniciar_foto();

	if(tu==0){
		cerr<< "No hay memoria suficiente";
		exit(-1);
	}
	tu->tabla = doble_puntero; //asignamos a puntero a puntero para que sea dinamico.

	tu->tabla[0] = crearusuario ("zatu","antonio","guerrero", "administrador", 0, 0);

	tu->tabla[1] = crearusuario ("x_pep","pepe","vegano", "limitado", 3, 3);
	tu->tabla[1]->v_foto[0] = *f;
	introducir_una_foto(f, "escritorio/home", "raw" ,12311);
	tu->tabla[1]->v_foto[1] = *f;
	introducir_una_foto(f, "escritorio/home", "png" ,123);
	tu->tabla[1]->v_foto[2] = *f;
	introducir_una_foto(f, "disco1/home", "raw" ,198032);
	tu->tabla[2] = crearusuario ("b_pablolondra","pablo","londra", "administrador", 1, 1);
	tu->tabla[2]->v_foto[0] = *f;
	introducir_una_foto(f, "escritorio/home", "png" ,111000);
	tu->tabla[3] = crearusuario ("c_pedrito","Pedro","infantes", "limitado", 1 ,1);
	tu->tabla[3]->v_foto[0] = *f;
	introducir_una_foto(f, "disco1/home/escritorio", "jpg" ,19999);
	tu->tabla[4] = crearusuario ("zorrillo","manolo","mamut", "limitado", 1, 1);
	tu->tabla[4]->v_foto[0] = *f;
	introducir_una_foto(f, "home", "png" ,19880);
	tu->tabla[5] = crearusuario ("xixo","raul","arenas", "administrador", 1, 1);
	tu->tabla[5]->v_foto[0] = *f;
	introducir_una_foto(f, "carpetaraul", "png" ,52390);
	tu->tabla[6] = crearusuario ("guille","guillermo","fiestas", "limitado", 1, 1);
	tu->tabla[6]->v_foto[0] = *f;
	introducir_una_foto(f, "escritorio", "jpg" , 78990);

	return tu;
}
/**
 * @brief modulo que imprime una foto
 * @pre le pasamos la foto que queramos imprimir
 * @infor llamamos a los getruta, gettipo, gettamaño que contienen los datos de la foto
 * @post no devuelvo nada ya que solo voy a imprimir una foto.
 */
void imprimir_una_foto(Foto *f){
	cout << " La ruta de la foto es. " << getruta(f) << endl;
	cout << " El tipo de la foto es. " << gettipo(f) << endl;
	cout << " El tamaño de la foto es. " << gettamanio(f) << endl;
}
/**
 * @brief modulo que imprime todas las  fotos que contiene un usuario
 * @pre le pasamos el usuario que queramos imprimir
```

```
 * @infor llamamos imprimir una foto y le pasamos la foto de ese usuario a la funcion contienen los datos de la foto
 * @post no devuelvo nada ya que solo voy a imprimir las fotos.
 */
void imprimir_las_fotos(Usuario *u){
	for( int i = 0; i < getdim_v_foto(u); i++){
		cout << " Imprimiendo las fotos del usuario con el nombre " << getnombre(u) << endl;
		cout << "
********************************************************************** " << endl;
		cout << " ***		Siguiente foto.			****" << endl;
		imprimir_una_foto(&u -> v_foto[i]);
		cout << " ***		Siguiente foto.			****" << endl;
		cout << "
********************************************************************** " << endl;
	}
}
/**
 * @brief modulo que imprime un usuario
 * @pre le pasamos la usuario que queramos imprimir
 * @info llamamos a los getlogin, getnombre, getapellido, getperfilusuario, getdim_v_foto y gettotalfotosuario que contienen los datos del usuario
 * @post no devuelvo nada ya que solo voy a imprimir un usuario.
 */
void imprimirusuario(Usuario *u){
	cout << BLACK_COLOR << " Login: " << getlogin (u) << endl;
   cout << " Nombre: " << getnombre(u) << endl;
   cout << " Apellido: " << getapellido(u)  << endl;
   cout << " Perfil del usuario: " << getaperfilusuario(u) << endl;
   cout << " Dimenion de la v_foto(util) " << getdim_v_foto(u) << endl;
   cout << " Total de fotos de usuario(tamaño del vector de fotos del usuario): " << gettotalfotosusuario(u) << RESTORE_DEFAULT_COLOR << "\t" << endl;
}
/**
 * @brief modulo que imprime todos los datos que contiene un usuario
 * @pre le tabla de usuarios que queramos imprimir
 * @infor llamamos imprimir un usuario y le pasamos el usuario de ese usuario a la funcion contienen los datos del usuario
 * @post no devuelvo nada ya que solo voy a imprimir las usuario.
 */
void imprimirtabla(TablaUsuarios *tabla_u){
	for (int i = 0; i < gettotaltuplas(tabla_u); i++){
		cout << " Imprimiendo el usuario: " << i<< endl;
		imprimirusuario(tabla_u ->tabla[i]);
		cout << "
************************************************************************** " << endl;
		cout << WHITE_COLOR << " Siguiente usuario: " << RESTORE_DEFAULT_COLOR << "\t" << endl;
	}
}
/**
```

```
 * @brief Modulo que borra un usuario
 * @info intercambia el usuario a la ultima posicion y disminuyo el vector en una posicion
 * @pre pasadle la tabla de usuarios y el login del usuario que quiero borrar
 * @post al acabar el modulo abremos disminuido el vector en una posicion
 */
void borrarusuarios(TablaUsuarios *u, string login){
	Usuario *aux = 0;
	bool encontrado = false;
	for (int i = 0; i < gettotaltuplas(u); i++){
		if (login == getlogin(u->tabla[i])){
			cout << " usuario " << getnombre(u-> tabla[i]) << endl;
			aux = u->tabla[i];
			u-> tabla[i] = u -> tabla [u -> totaltuplas-1];
			u -> tabla [u -> totaltuplas-1] = aux;
			u -> tabla = resize ( u -> tabla, u->totaltuplas,u->totaltuplas,u->totaltuplas-1);
			encontrado = true;
		}
	}
	if(encontrado != true){
		cout << WHITE_COLOR << " Usuario no encontrado: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
	}
	else
		cout << RED_COLOR << " El usuario sea eliminado correctamente: " <<
RESTORE_DEFAULT_COLOR <<"\t" << endl;

	delete aux;
}
/**
 * @brief Modulo que borra una foto
 * @info intercambia la foto a la ultima posicion y disminuyo el vector en una posicion usuando el
resize
 * @pre le paso el usuario que contiene un vector de fotos que es donde se encuentra las fotos que
quiero borrar y la ruta
 * @post al acabar el modulo abremos disminuido el vector en una posicion
 */
void borrar_foto(Usuario *u, string ruta){
	Foto *aux = 0, *f = 0;
	aux = new Foto;
	f  = new Foto;
	bool encontrado = false;
	for (int i = 0; i < getdim_v_foto(u); i++){
		*f = u->v_foto[i];
		if (ruta == getruta(f)){
			*aux = u->v_foto[i];
			u -> v_foto[i] = u ->v_foto[u->dim_v_foto-1];
			u -> v_foto[u->dim_v_foto-1] = *aux;
			u -> v_foto = resizeFotos(u->v_foto, getdim_v_foto(u), u->dim_v_foto, u-
>dim_v_foto-1);
			encontrado = true;
			settotalfotosusuario(u, getdim_v_foto(u));
		}
```

```
	}
	if(encontrado != true){
		cout << GREEN_COLOR << " Foto no encontrada. "<<
RESTORE_DEFAULT_COLOR << "\t" << endl;
	}
	else
		cout << RED_COLOR << " La foto esta eliminada correctamente: " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;

}
/**
 * @brief Modulo que busca un usuario en la tabla de usuario(tabla o vector donde se encuentran
todos los usuario)
 * @info cuando encuentre el usuario llamare a otra funcion para imprimir todos los datos de ese
usuario
 * @pre pasadle la tabla de usuarios y el login del usuario que quiero buscar
 */
void buscarusuario(TablaUsuarios *u, string login){
	bool encontrado = false;
	for (int i = 0; i < gettotaltuplas(u) && encontrado != true; i++){
		if (login == getlogin(u->tabla[i])){
			cout << " bieen!!! lo hemos encontrado te imprimo todos sus datos: " << endl;
			imprimirusuario(u->tabla[i]);
		}
	}
}
/**
 * @brief Modulo que busca un usuario en la tabla de usuario(tabla o vector donde se encuentran
todos los usuario)
 * @info cuando encuentre el usuario llamare a otra funcion para imprimir todas las fotos del
usuario
 * @pre pasadle la tabla de usuarios y el login del usuario que quiero buscar
 */
void buscarusuario_imprimir_foto(TablaUsuarios *u, string login){
	bool encontrado = false;
	for (int i = 0; i < gettotaltuplas(u) && encontrado != true; i++){
		if (login == getlogin(u->tabla[i])){
			cout << " bieen!!! lo hemos encontrado te imprimo todas sus fotos: " << endl;
			imprimir_las_fotos(u->tabla[i]);
		}
	}
}
/**
 * @brief Modulo que busca un usuario en la tabla de usuario(tabla o vector donde se encuentran
todos los usuario)
 * @info cuando encuentre el usuario devolvere un booleano diciendo que ese usuario si existe
 * @pre pasadle la tabla de usuarios y el login del usuario que quiero buscar
 * @post devuelvo un bool diciendo que es true si existe el usuario o false si no lo encontrado
 */
bool comprobar_si_existe(TablaUsuarios *u, string login){
	bool esta = false;
	for (int i = 0; i < gettotaltuplas(u) && esta != true; i++){
```

```
                if (login == getlogin(u->tabla[i])){
                        esta = true;
                }
        }
        return esta;
}
/**
 * @brief Modulo que busca un usuario en la tabla de usuario(tabla o vector donde se encuentran todos los usuario)
 * @info cuando encuentre el usuario devolvere el usuario, le pido que me introduzca el usuario en este modulo para buscar si existe
 * @pre pasadle la tabla de usuarios
 * @post devuelvo el usuario que estaba buscando
 */
Usuario* comprobar_si_existe_foto(TablaUsuarios *u){
        bool encontrado = false;
        string usuario;
        Usuario *u_retur;

        do{
                cout << YELLOW_COLOR << "introduce el nombre del usuario nuevo:" << RESTORE_DEFAULT_COLOR <<"\t" << endl;
                cin >> usuario;
                for (int i = 0; i < gettotaltuplas(u) && encontrado != true; i++){
                        if (usuario == getlogin(u->tabla[i])){
                                encontrado = true;
                                u_retur = u->tabla[i];
                        }
                }
        }while (encontrado != true);

        return u_retur;
}
/**
 * @brief Modulo en el que le voy a pedir al usuario que introduzca todos los datos para crear un usuario nuevo
 * @info pedir al usuario todos los datos necesarios y comprobar si el usuario existe
 * @pre pasadle la tabla de usuarios ya creada
 * @param nombre, apellido, usuario, tipo. login util_foto, dimension de la foto, que son todos los datos que le vamos a introducir al usuario
 * @post despues de introducir todos los datos por pantalla llamare a una funcion donde creare un usuario con sus set pertinentes
 */
void datos_crear_usuario(TablaUsuarios *tablausuario, string nombre,string apellido,string usuario,string tipo,string login, int util_foto, int dim_fotos){
        bool encontrado = false;

        cout << CYAN_COLOR << " Vamos añadir un usuario nuevo: "<< RESTORE_DEFAULT_COLOR << "\t" << endl;
        do{
                cout << YELLOW_COLOR << "introduce nombre el usuario" << RESTORE_DEFAULT_COLOR <<"\t" << endl;
```

```
                cin >> usuario;
                encontrado = comprobar_si_existe(tablausuario, usuario);
        }while (encontrado != false);

        cout << YELLOW_COLOR << "introduce el nombre: "<< RESTORE_DEFAULT_COLOR
<<"\t" << endl;
        cin >> nombre;
        cout << YELLOW_COLOR << "introduce el apellido: "<< RESTORE_DEFAULT_COLOR
<<"\t" << endl;
        cin >> apellido;
        cout << YELLOW_COLOR << "introduce el tipo usuario (particular, tienda): "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
        cin >> tipo;
        //hago el resize y aumento en uno la posicion del vector
        anadirusuarios(tablausuario, usuario, nombre, apellido, tipo, util_foto, dim_fotos);
}
/**
 * @brief Modulo en el que le voy a pedir al usuario que introduzca todos los datos para crear un
foto nueva
 * @info pedir al usuario todos los datos necesarios y comprobar si el usuario existe
 * @pre pasadle la tabla de usuarios ya creada
 * @post despues de introducir todos los datos por pantalla llamare a una funcion donde creare un
usuario con sus set pertinentes y aumentare el vector en una posicion
 */
void datos_crear_foto(TablaUsuarios *tu){
        bool encontrado = false;
        string ruta, tipo, usuario;
        int tam;
        Usuario *u;
        Foto *foto_nueva = 0;
        foto_nueva = new Foto;

        cout << CYAN_COLOR << " Vamos añadir un usuario: "<<
RESTORE_DEFAULT_COLOR << "\t" << endl;
        //compruebo si existe un usuario
        u = comprobar_si_existe_foto(tu);

        cout << " introduce la ruta: " << endl;
        cin >> ruta;
        cout << " intruduce el tipo (png, raw, jpg): " << endl;
        cin >>tipo;
        cout << " intruduce el tamaño: " << endl;
        cin >>tam;
        //llamo a los set
        introducir_una_foto(foto_nueva, ruta, tipo, tam);
        //aumento en 1 la posicion del vector
        introducirfoto(u,usuario, ruta, tipo,tam, foto_nueva);
}
/**
 * @brief Modulo borro la tabla de usuarios
 * @info recorrer todo el vector y eliminar primero las fotos despues cada usuario y por ultimo la
tabla
```

```
 * @pre pasadle la tabla de usuarios ya creada que vamos a borrar
 * @post despues de este modulo abremos borrado todo, icluyendo todas las fotos, todos los
usuarios y todos los punteros
 */
void borrar_tabla(TablaUsuarios *ta){
        for (int i = 0; i < gettotaltuplas(ta); i++){
                delete [] ta->tabla[i]->v_foto; //borra el vector fotos de la tabla de usuarios
                delete ta->tabla[i]; // borramos todos los usuarios.
        }
        ta->tabla = 0;
        delete ta->tabla;
        ta = 0;
        delete ta;
}
/**
 * @brief Modulo que ordena la tabla de usuarios
 * @info modulo que ordena los usuarios por login
 * @pre pasadle la tabla de usuarios ya creada que vamos a ordenar
 * @post devolver la tabla de usuarios con todos los usuarios ya ordenados
 */
void burbuja(TablaUsuarios *ta) {
        int i = 0, j = 0;
        Usuario *us;
   for(i = 0; i < gettotaltuplas(ta); i++){
      for(j = i+1; j < gettotaltuplas(ta); j++){
         if((getlogin(ta->tabla[i]) > getlogin(ta->tabla[j])) || (getlogin(ta->tabla[i]) == getlogin(ta-
>tabla[j]))){
            us = ta->tabla[i];
            ta->tabla[i] = ta->tabla[j];
            ta->tabla[j] = us;
         }
      }
   }
}

/**
 * @brief Modulo que ordena la tabla de usuarios
 * @info modulo que ordena los usuarios por el total de fotos
 * @pre pasadle la tabla de usuarios ya creada que vamos a ordenar
 * @post devolver la tabla de usuarios con todos los usuarios ya ordenados por el totoal de fotos
 */
void burbuja_total_fotos(TablaUsuarios *ta) {
        int i = 0, j = 0;
        Usuario *us;
   for(i = 0; i < gettotaltuplas(ta); i++){
      for(j = i+1; j < gettotaltuplas(ta); j++){
         if((getdim_v_foto(ta->tabla[i]) > getdim_v_foto(ta->tabla[j])) || getdim_v_foto(ta->tabla[i])
== getdim_v_foto(ta->tabla[j])){
            us = ta->tabla[i];
            ta->tabla[i] = ta->tabla[j];
            ta->tabla[j] = us;
         }
```

```
      }
   }
}
/**
* @brief menu que sale por pantalla en el que llamaremos a diferentes funciones en funcion de lo que nos introduzca el usuario
*@ 1)  pulsa 1 Crear la tabla de Usuarios.
*@ 2)  pulsa 2 Eliminar la tabla Usuarios.
*@ 3)  pulsa 3 Imprimir la tabla Usuarios.
*@ 4)  pulsa 4 Insertar un Usuario en tabla Usuarios.
*@ 5)  pulsa 5 Eliminar un Usuario en tabla Usuarios.
*@ 6)  pulsa 6 Buscar un Usuario por Atributo Login
*@ 7)  pulsa 7 Ordenar tabla de Usuarios por Atributo
*@ 8)  pulsa 8 Añadir una Fotografía a un Usuario.
*@ 9)  pulsa 9 Eliminar una Fotografía de un Usuario "
*@ 10)          pulsa 10 Imprimir todas las Fotos de un Usuario.
*@ 11)  pulsa 11 ordenando la tabla de usuarios por total de fotos
*@ 12)          pulsa 12 Salir " << endl;
*@ post al pulsar 11 borraremos todo
*@ info para hacer cualquier cosa primero tendremos que crear la tabla
*/
void menu_en_pantalla() {
   string nombre, apellido, usuario, tipo, login, ruta, foto, usuario_Crear;
   int menu, anio = 0, si = 0, util_foto = 0, dim_fotos = 0;
   int tamanio = 0;
   bool creada_tabla = false;
   TablaUsuarios *tablausuario = 0;
   Usuario *u = 0;
   u = new Usuario;

   do{ //filtro
          cout << " pulsa 1 Crear la tabla de Usuarios. " << endl;
               cout << " pulsa 2 Eliminar la tabla Usuarios. "<< endl;
               cout << " pulsa 3 Imprimir la tabla Usuarios. "<< endl;
               cout << " pulsa 4 Insertar un Usuario en tabla Usuarios. "<< endl;
               cout << " pulsa 5 Eliminar un Usuario en tabla Usuarios. "<< endl;
               cout << " pulsa 6 Buscar un Usuario por Atributo Login "<< endl;
               cout << " pulsa 7 Ordenar tabla de Usuarios por Atributo "<< endl;
               cout << " pulsa 8 Añadir una Fotografía a un Usuario. "<< endl;
               cout << " pulsa 9 Eliminar una Fotografía de un Usuario "<< endl;
               cout << " pulsa 10 Imprimir todas las Fotos de un Usuario." << endl;
               cout << " pulsa 11 ordenando la tabla de usuarios por total de fotos. " << endl;
               cout << " pulsa 12 Salir " << endl;
               cout << RED_COLOR <<" Si borras la tabla tienes que añadir una nueva[1] si no te tienes que salir del programa pulsando [12] "<< RESTORE_DEFAULT_COLOR << "\t" << endl;
               do{
               cin >> menu;
        }while(menu != 1 && menu != 12 && creada_tabla != true);

        switch (menu) {
             case 1:
               //creamos la tabla
```

```
                if(tablausuario != 0){

                        cout << RED_COLOR << "Error. La tabla ya ha sido creada, borrala
antes de volver a generarla " << RESTORE_DEFAULT_COLOR <<"\t"  << endl;
                        }
                        else
                        {
                                tablausuario = InicializarTabla();
                                cout << GREEN_COLOR << "Acabas de crear la tabla: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
                                creada_tabla = true;
                        }
        break;
        case 2:
          //borramos la tabla
                        if (creada_tabla == true){
                                borrar_tabla(tablausuario);
                                tablausuario = 0;
                                cout << GREEN_COLOR  << " Has borrado la tabla: " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                        }
                        else
                                cout << RED_COLOR << " la tabla ya esta borrada " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                                creada_tabla = false;
        break;
        case 3:
          // llamamos a imprimir los usuarios de la tabla
                        cout << RED_COLOR<< "Imprimiendo la tabla: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
                        imprimirtabla (tablausuario);

        break;
        case 4:
          //añadimos un usuario
                datos_crear_usuario(tablausuario, nombre, apellido, usuario, tipo, login,
util_foto, dim_fotos);
        break;

        case 5:
          //borramos un usuario
                cout << RED_COLOR << "¿Seguro que quieres borrar a tu amigo?: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
                cout << CYAN_COLOR << "Introduce el nombre del usuario que quieres
borrar "<< RESTORE_DEFAULT_COLOR <<"\t" << endl;
                cin >> login;
                borrarusuarios(tablausuario, login);
                break;
                case 6:
                        //buscamos un usuario
                        cout << BLUE_COLOR << "Vamos a buscar a tu amigo: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
```

```
                cout << BLUE_COLOR << "Introduce el login del usuario que
quieres buscar."<< RESTORE_DEFAULT_COLOR <<"\t" << endl;
                cin >> login;
                buscarusuario(tablausuario, login);
            break;
            case 7:
                //ordenamos la tabla de usuarios
                cout << YELLOW_COLOR << " ordenando la tabla de usuarios " <<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
                burbuja(tablausuario);
            break;
            case 8:
                //creamos una foto en un usuario
                datos_crear_foto(tablausuario);
            break;
            case 9:
                //eliminamos la foto de un usuario
                cout << CYAN_COLOR << "Vamos eliminar fotos del usuario: "<<
RESTORE_DEFAULT_COLOR <<"\t" << endl;
            cout << BLUE_COLOR << " introduce la ruta de la foto " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                cin >> ruta;
                u = comprobar_si_existe_foto(tablausuario);
                borrar_foto(u, ruta);
            break;
            case 10:
                //buscamos la foto de un usuario
                cout << GREEN_COLOR << "Vamos a buscar a la foto: "<<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                cout << YELLOW_COLOR << "Introduce el nombre del usuario que
quieres buscar." << RESTORE_DEFAULT_COLOR << "\t" << endl;
                cin >> login;

                buscarusuario_imprimir_foto(tablausuario, login);
            break;
            case 11:
                //ordenamos la tabla de usuarios por el total de fotos
                cout << CYAN_COLOR << " ordenando la tabla de usuarios por total
de fotos " << RESTORE_DEFAULT_COLOR << "\t"  << endl;
                burbuja_total_fotos(tablausuario);
            break;
            case 12:
                //salimos del programa y borramos la tabla usuario y foto como en el
numero 2
                cout << WHITE_COLOR << " Hasta luego espero mi 10 "<<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                if (creada_tabla == true){
                    borrar_tabla(tablausuario);
                    tablausuario = 0;
                    cout << GREEN_COLOR  << " Has borrado la tabla: " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                }
```

```
                                        else
                                                cout << RED_COLOR << " la tabla ya esta borrada " <<
RESTORE_DEFAULT_COLOR << "\t" << endl;
                                                creada_tabla = false;
                                break;
                        }
        }while(menu != 12);
}
int main(){
        menu_en_pantalla();
}
```