



Christopher Ceballos Jimenez  
Mtra. Jannette Araceli Castellanos Barajas.  
Materia: Seminario de Estructuras II.

Actividad: Índices Invertidos.

## Código:

### Clases

```
template < typename E >
void mostrar( ListaLigada< E > &L );

class IndiceSecundario
{
public:
    char nombre[ 35 ] ;
    int enlace;
};

class Ligada
{
public:
    int id;
    int enlace;
};

ListaLigada< string > arregloIndices;
ListaLigada< int > nrrs;
ListaLigada< int > listaLigada;
```

```

class Factura
{
private:
    char id[ 10 ];
    char nombre[ 35 ];
    char direccion[ 40 ];
    char RFC[ 15 ];

    int buscarEnIndice( const string & );
    int existeId( const int );
    int buscarEnLigado( const int );
    Factura eliminarEnArchDatos( const int );
    void eliminarEnNrre( const int );
    void eliminarEnIndiceLigado( const int, const int );
    void eliminarEnIndiceFacturas( const string & );
    Factura modificarEnArchDatos( const int, const Factura & );
    void modificarEnNrre( const int, const int );
    void modificarEnIndiceFacturas( const string & nombreAnterior, const string & nuevoNombre, const int idViejo, const int nuevoId );
    void modificarEnIndiceLigado( const string &, const int );

public:
    Factura( void ) {}
    Factura( const Factura & );
    void setId( const string & );
    void setNombre( const string & );
    void setDireccion( const string & );
    void setRFC( const string & );
    Factura pedirRegistro( void );
    void agregar( const Factura & );

```

```

    void agregar( const Factura & );
    void mostrarTodo( void );
    void inicializar( void );
    void buscarPorNombre( const string & );
    void guardar( void );
    bool buscarId( const int, Factura & );
    bool eliminar( const int, Factura & );
    void modificar( const int, const Factura & );
    friend ostream &operator<<( ostream &, const Factura & );
    void mostrarLista( void );
};

```

```

void Factura::mostrarLista( void )

```

```

{
    mostrar( listaLigada );
    mostrar( nrre );
    mostrar( arregloIndices );
}

```

```

Factura Factura::pedirRegistro( void )

```

```

{
    string cadena;
    Factura FacturaARetornar;

    cout << "\n\t\t\t\t\t *-*-*CAPTURA DE DATOS DEL USUARIO*-*-*";

    cout << "\n\n\t\t Ingrese el ID del cliente: ";
    fflush( stdin );

```

```

        getline( cin, cadena );
    }
    FacturaARetornar.setId( cadena );

    cout << "\n\t\t Ingrese el nombre del cliente: ";
    fflush( stdin );
    getline( cin, cadena );
    FacturaARetornar.setNombre( cadena );

    cout << "\n\t\t Ingrese la direccion de facturacion: ";
    fflush( stdin );
    getline( cin, cadena );
    FacturaARetornar.setDireccion( cadena );

    cout << "\n\t\t Ingrese el RFC: ";
    fflush( stdin );
    getline( cin, cadena );
    FacturaARetornar.setRFC( cadena );

    return FacturaARetornar;
}

Factura::Factura( const Factura &origen )
{
    strcpy( id, origen.id );
    strcpy( nombre, origen.nombre );
    strcpy( direccion, origen.direccion );
    strcpy( RFC, origen.RFC );
}

```

```
void Factura::setId( const string &valorId )
```

```
{  
    int longitud = valorId.size();  
    longitud = ( longitud < 10 ? longitud : 9 );  
    valorId.copy( id, longitud );  
    id[ longitud ] = '\\0';  
}
```

```
void Factura::setNombre( const string &valorNombre )
```

```
{  
    int longitud = valorNombre.size();  
    longitud = ( longitud < 35 ? longitud : 34 );  
    valorNombre.copy( nombre, longitud );  
    nombre[ longitud ] = '\\0';  
}
```

```
void Factura::setDireccion( const string &valorDireccion )
```

```
{  
    int longitud = valorDireccion.size();  
    longitud = ( longitud < 40 ? longitud : 39 );  
    valorDireccion.copy( direccion, longitud );  
    direccion[ longitud ] = '\\0';  
}
```

```
void Factura::setRFC( const string &valorRFC )
```

```
{  
    int longitud = valorRFC.size();  
    longitud = ( longitud < 15 ? longitud : 14 );  
    valorRFC.copy( RFC, longitud );  
    RFC[ longitud ] = '\\0';  
}
```

```
ostream &operator<<( ostream &salida, const Factura &Factura )
```

```
{  
    salida << "\\n\\t\\t ID:          " << Factura.id << endl  
        << "\\t\\t Nombre:      " << Factura.nombre << endl  
        << "\\t\\t Direccion: " << Factura.direccion << endl  
        << "\\t\\t RFC:          " << Factura.RFC << endl;  
  
    return salida;  
}
```

```
void Factura::mostrarTodo( void )
```

```
{  
    Factura factura;  
    ifstream archivo( "facturas.txt", ios::in );  
    if( !archivo )  
        cout << "No existe el archivo" << endl;  
    else  
    {  
        cout << endl;  
        while( !archivo.eof() )  
        {
```

```

    {
        archivo.read( ( char * ) &factura, sizeof( Factura ) );
        if( !archivo.eof() )
            cout << factura << endl << endl;
    }
    archivo.close();
}

int Factura::existeId( const int idABuscar )
{
    nrrs.irAlInicio();
    for( int i = 0; i < nrrs.obtenerTamanio(); i++ )
        if( nrrs.obtenerElemento() == idABuscar )
            return nrrs.obtenerEnlace();
        else
            nrrs.siguiente();
    return -1;
}

int Factura::buscarEnIndice( const string &nombreABuscar )
{
    arregloIndices.irAlInicio();
    for( int i = 0; i < arregloIndices.obtenerTamanio(); i++ )
        if( arregloIndices.obtenerElemento() == nombreABuscar )
            return arregloIndices.posicionActual();
        else
            arregloIndices.siguiente();
    return -1;
}

```

```

int Factura::buscarEnLigado( const int idABuscar )
{
    listaLigada.irAlInicio();
    for( int i = 0; i < listaLigada.obtenerTamano(); i++ )
        if( listaLigada.obtenerElemento() == idABuscar )
            return listaLigada.posicionActual();
        else
            listaLigada.siguiente();
    return -1;
}

Factura Factura::eliminarEnArchDatos( const int idAEliminar )
{
    ifstream datos( "facturas.txt", ios::in );
    ofstream temporal( "temporal.txt", ios::out );
    Factura registro;
    Factura registroARetornar;

    while( !datos.eof() )
    {
        datos.read( ( char * ) &registro, sizeof( Factura ) );
        if( atoi( registro.id ) != idAEliminar )
            temporal.write( ( char * ) &registro, sizeof( Factura ) );
        else
            registroARetornar = registro;
    }
    datos.close();
    temporal.close();
    remove( "facturas.txt" );
}

```



```

    return registroARetornar;
}

void Factura::eliminarEnNrRs( const int idAEliminar )
{
    int posicion = existeId( idAEliminar );
    nrRs.irAPosicion( posicion );
    //cout << "posicion = " << posicion << endl;
    nrRs.eliminar();
    for( int i = posicion; i < nrRs.obtenerTamano(); i++ )
    {
        nrRs.modificar( nrRs.obtenerElemento(), nrRs.obtenerEnlace() - 1 );
        nrRs.siguiente();
    }
    //mostrar( nrRs );
}

void Factura::eliminarEnIndiceFacturas( const string &nombreAEliminar )
{
    int posicion = buscarEnIndice( nombreAEliminar );
    arregloIndices.irAPosicion( posicion );
    listaLigada.irAPosicion( arregloIndices.obtenerEnlace() );
    if( listaLigada.obtenerEnlace() == -1 )
        arregloIndices.eliminar();
}

void Factura::eliminarEnIndiceLigado( const int idAEliminar, const int posDeRegistro )
{
    int posicion = buscarEnLigado( idAEliminar );
    listaLigada.irAPosicion( posicion );
    int aux = listaLigada.posicionActual();
    int posNueva = listaLigada.obtenerEnlace();
    arregloIndices.irAPosicion( posDeRegistro );
    if( listaLigada.obtenerEnlace() == -1 )
        listaLigada.modificar( -1, -1 );
    else
    {
        if( arregloIndices.obtenerEnlace() == listaLigada.posicionActual() )
        {
            arregloIndices.modificar( arregloIndices.obtenerElemento(), listaLigada.obtenerEnlace() );
            listaLigada.modificar( -1, -1 );
        }
        else
        {
            listaLigada.irAPosicion( arregloIndices.obtenerEnlace() );
            while( listaLigada.obtenerEnlace() != aux )
                listaLigada.irAPosicion( listaLigada.obtenerEnlace() );
            listaLigada.modificar( listaLigada.obtenerElemento(), posNueva );
            listaLigada.irAPosicion( posicion );
            listaLigada.modificar( -1, -1 );
        }
    }
}

```

```

void Factura::inicializar( void )
{
    IndiceSecundario indiSecundario;
    Ligada liga;
    string cadenaNombre;
    ifstream indice( "indiceFacturas.txt", ios::in );
    ifstream indiceLigado( "ligadoFacturas.txt", ios::in );
    ifstream indiceNrr( "indiceNrr.txt", ios::in );

    if( !indice || !indiceLigado || !indiceNrr ) /// si los archivos no existen los crea
    {
        ofstream indice( "indiceFacturas.txt", ios::out );
        ofstream indiceLigado( "ligadoFacturas.txt", ios::out );
        ofstream indiceNrr( "indiceNrr.txt", ios::out );
    }
    else
    {
        /// archivo de indices para el nombre
        arregloIndices.irAlInicio();
        while( !indice.eof() )
        {
            indice.read( ( char * ) &indiSecundario, sizeof( IndiceSecundario ) );
            if( !indice.eof() )
            {
                cadenaNombre = indiSecundario.nombre;
                arregloIndices.ponAlFinal( cadenaNombre, indiSecundario.enlace );
            }
            cadenaNombre.clear();
        }
    }
}

```

```

    /// archivo de indice ligado para nombre
    listaLigada.irAlInicio();
    while( !indiceLigado.eof() )
    {
        indiceLigado.read( ( char * ) &liga, sizeof( Ligada ) );
        if( !indiceLigado.eof() )
            listaLigada.ponAlFinal( liga.id, liga.enlace );
    }
    /// archivo de indice de nrr's para los datos de empleados
    nrrs.irAlInicio();
    while( !indiceNrr.eof() )
    {
        indiceNrr.read( ( char * ) &liga, sizeof( Ligada ) );
        if( !indiceNrr.eof() )
            nrrs.ponAlFinal( liga.id, liga.enlace );
    }

    indice.close();
    indiceLigado.close();
    indiceNrr.close();
}

void Factura::guardar( void )
{
    ofstream indice( "indiceFacturas.txt", ios::out );
    ofstream indiceLigado( "ligadoFacturas.txt", ios::out );
    ofstream indiceNrr( "indiceNrr.txt", ios::out );

    IndiceSecundario contenedor;
    arregloIndices.irAlInicio();
    for( int i = 0; i < arregloIndices.obtenerTamano(); i++ )
    {
        for( int j = 0; j < sizeof( contenedor.nombre ); contenedor.nombre[ j++ ] = '\0' );
        strcpy( contenedor.nombre, arregloIndices.obtenerElemento().c_str() );
        contenedor.enlace = arregloIndices.obtenerEnlace();
        indice.write( ( char * ) &contenedor, sizeof( IndiceSecundario ) );
        arregloIndices.siguiente();
    }

    Ligada liga;
    listaLigada.irAlInicio();
    for( int i = 0; i < listaLigada.obtenerTamano(); i++ )
    {
        liga.id = listaLigada.obtenerElemento();
        liga.enlace = listaLigada.obtenerEnlace();
        indiceLigado.write( ( char * ) &liga, sizeof( Ligada ) );
        listaLigada.siguiente();
    }
}

```

```

nrrs.irAlInicio();
for( int i = 0; i < nrrs.obtenerTamanio(); i++ )
{
    liga.id = nrrs.obtenerElemento();
    liga.enlace = nrrs.obtenerEnlace();
    indiceNrr.write( ( char * ) &liga, sizeof( Ligada ) );
    nrrs.siguiente();
}

indice.close();
indiceLigado.close();
indiceNrr.close();
}

```

Métodos:

### Agregar

```

void Factura::agregar( const Factura &nuevoValor )
{
    string cadenaNombre;

    ofstream archivo( "facturas.txt", ios::app );
    if( existeId( atoi( nuevoValor.id ) ) == -1 )
    {
        archivo.write( ( char * ) &nuevoValor, sizeof( Factura ) );
        nrrs.ponAlFinal( atoi( nuevoValor.id ), nrrs.obtenerTamanio() );

        cadenaNombre = nuevoValor.nombre;
        int posEncontrado = buscarEnIndice( cadenaNombre );
        if( posEncontrado != -1 ) /// si ya existe el nombre
        {
            arregloIndices.irAPosicion( posEncontrado );
            listaLigada.ponAlFinal( atoi( nuevoValor.id ), arregloIndices.obtenerEnlace() );
            arregloIndices.modificar( arregloIndices.obtenerElemento(), listaLigada.obtenerTamanio() - 1 );
        }
        else /// si no existe el nombre
        {
            listaLigada.ponAlFinal( atoi( nuevoValor.id ), -1 );
            arregloIndices.ponAlFinal( cadenaNombre, listaLigada.obtenerTamanio() - 1 );
        }
        mostrar( listaLigada );
        mostrar( nrrs );
        mostrar( arregloIndices );
    }
    else
        cout << "No se agrego, el ID ya existe" << endl;
    archivo.close();
}

```

## Buscar

```
bool Factura::buscarId( const int idABuscar, Factura &resultado )
{
    long int posByte;
    ifstream datos( "facturas.txt", ios::in );

    if( !datos )
        cout << "No existe el archivo" << endl;
    else
    {
        nrrs.irAlInicio();
        for( int i = 0; i < nrrs.obtenerTamano(); i++ )
        {
            if( nrrs.obtenerElemento() == idABuscar )
            {
                posByte = nrrs.obtenerEnlace() * sizeof( Factura );
                datos.seekg( posByte, ios::beg );
                datos.read( ( char * ) &resultado, sizeof( Factura ) );
                datos.close();
                return true;
            }
            nrrs.siguiente();
        }
        datos.close();
        return false;
    }
}

void Factura::buscarPorNombre( const string & nombreABuscar )
{
    ifstream datos( "facturas.txt", ios::in );
    int indiceRegistro;
    long int posByte;
    Factura temporal;
    bool yaTermino = false;

    if( !datos )
        cout << "No existe el archivo" << endl;
    else
    {
        int posEncontrado = buscarEnIndice( nombreABuscar );
        if( !( posEncontrado == -1 ) )
        {
            cout << endl;
            arregloIndices.irAPosicion( posEncontrado );
            listaLigada.irAPosicion( arregloIndices.obtenerEnlace() );
            do
            {
                indiceRegistro = existeId( listaLigada.obtenerElemento() );
                posByte = indiceRegistro * sizeof( Factura );
                datos.seekg( posByte, ios::beg );
                datos.read( ( char * ) &temporal, sizeof( temporal ) );
                cout << temporal << endl << endl;
                if( listaLigada.obtenerEnlace() == -1 )
                    yaTermino = true;
            }
            else
                listaLigada.irAPosicion( listaLigada.obtenerEnlace() );
        }
    }
}
```

```

        else
            listaLigada.irAPosicion( listaLigada.obtenerEnlace() );
        }
        while( yaTermino == false );
    }
    else
        cout << "No se encontraron resultados" << endl;
    }
    datos.close();
}

```

## Modificar

```

void Factura::modificar( const int idAModificar, const Factura &reemplazo )
{
    string cadenaNombre, cad;
    Factura empleadoModificado;
    if( existeId( idAModificar ) != -1 && existeId( atoi( reemplazo.id ) ) == -1 ) // si existe el id
    {
        empleadoModificado = modificarEnArchDatos( idAModificar, reemplazo );
        modificarEnNrres( idAModificar, atoi( reemplazo.id ) );
        cadenaNombre = reemplazo.nombre;
        cad = empleadoModificado.nombre;
        modificarEnIndiceFacturas( cad, cadenaNombre, atoi( empleadoModificado.id ), atoi( reemplazo.id ) );
        cadenaNombre = empleadoModificado.nombre;
        if( !( strcmp( empleadoModificado.nombre, reemplazo.nombre ) == 0 ) )
            modificarEnIndiceLigado( cadenaNombre, atoi( empleadoModificado.id ) );
    }
    else
        cout << "El registro no existe o el id ya existe" << endl;

    mostrar( listaLigada );
    mostrar( nrres );
    mostrar( arregloIndices );
}

void Factura::modificarEnIndiceLigado( const string &nombreAModificar, const int idViejo )
{
    int posicion = buscarEnIndice( nombreAModificar );
    int a, b, c;
    bool bandera = false;
    arregloIndices.irAPosicion( posicion );
    listaLigada.irAPosicion( arregloIndices.obtenerEnlace() );
    a = c = listaLigada.posicionActual();
    if( listaLigada.obtenerElemento() == idViejo && listaLigada.obtenerEnlace() == -1 )
    {
        arregloIndices.eliminar();
        listaLigada.modificar( -1, -1 );
    }
    else
    {
        while( listaLigada.obtenerElemento() != idViejo )
        {
            a = listaLigada.posicionActual();
            listaLigada.irAPosicion( listaLigada.obtenerEnlace() );
            c = listaLigada.posicionActual();
            bandera = true;
        }
        b = listaLigada.obtenerEnlace();
        if( bandera == true )
            arregloIndices.modificar( arregloIndices.obtenerElemento(), a );
        else
            arregloIndices.modificar( arregloIndices.obtenerElemento(), b );

        listaLigada.irAPosicion( a );
    }
}

```

```

        listaLigada.irAPosicion( c );
        listaLigada.modificar( -1, -1 );
    }
}

void Factura::modificarEnIndiceFacturas( const string &nombreAnterior, const string &nuevoNombre, const int idViejo, const int nuevoId )
{
    int posicion = buscarEnIndice( nuevoNombre );
    if( posicion == -1 ) /// si nuevoNombre NO existe en el archivo
    {
        listaLigada.ponAlFinal( nuevoId, -1 );
        arregloIndices.ponAlFinal( nuevoNombre, listaLigada.obtenerTamanio() - 1 );
    }
    else /// si nuevoNombre YA existe
    {
        if( nombreAnterior == nuevoNombre )
        {
            arregloIndices.irAPosicion( posicion );
            listaLigada.irAPosicion( arregloIndices.obtenerEnlace() );
            while( listaLigada.obtenerElemento() != idViejo )
                listaLigada.irAPosicion( listaLigada.obtenerEnlace() );
            listaLigada.modificar( nuevoId, listaLigada.obtenerEnlace() );
        }
        else
        {
            arregloIndices.irAPosicion( posicion );
            listaLigada.ponAlFinal( nuevoId, arregloIndices.obtenerEnlace() );
            arregloIndices.modificar( arregloIndices.obtenerElemento(), listaLigada.obtenerTamanio() - 1 );
        }
    }
}

void Factura::modificarEnNrRs( const int idAModificar, const int nuevoID )
{
    int posicion = existeId( idAModificar );
    nrRs.irAPosicion( posicion );
    nrRs.modificar( nuevoID, nrRs.obtenerEnlace() );
}

Factura Factura::modificarEnArchDatos( const int idAModificar, const Factura &registroReemplazo )
{
    Factura registroARetornar;
    ifstream datos( "facturas.txt", ios::in | ios::out );
    if( datos )
    {
        int posDeRegistro = existeId( idAModificar );
        int posByte = posDeRegistro * sizeof( Factura );
        datos.seekg( posByte, ios::beg );
        datos.read( ( char * ) &registroARetornar, sizeof( Factura ) );
        datos.seekg( (long)datos.tellg() - sizeof( Factura ), ios::beg );
        datos.write( ( char * ) &registroReemplazo, sizeof( Factura ) );
    }
    else
        cout << "No existe el archivo" << endl;
    datos.close();
    return registroARetornar;
}

```

Eliminar



Menú

```

int main( void )
{
    Factura factura, nuevo, facturaBuscar, facturaEliminado;
    string respuesta;
    char opcion;
    factura.inicializar();
    do
    {
        system("cls");
        cout << "\n\n\t\t\t\t\t *-*-*FACTURACION*-*-*" << endl;
        cout << "\n\t\t\t\t\t [1]- Agregar datos de facturacion." << endl;
        cout << "\t\t\t\t\t [2]- Mostrar datos para facturacion." << endl;
        cout << "\t\t\t\t\t [3]- Buscar datos por nombre" << endl;
        cout << "\t\t\t\t\t [4]- Modificar datos." << endl;
        cout << "\t\t\t\t\t [5]- Eliminar datos de facturacion." << endl;
        cout << "\t\t\t\t\t [6]- Mostrar la lista ligada." << endl;
        cout << "\t\t\t\t\t [7]- Salir" << endl;
        cout << "\t\t\t\t\t Elija una opcion: ";
        cin >> opcion;
        switch( opcion )
        {
            case '1': /// agregar
                system("cls");
                nuevo = factura.pedirRegistro();
                factura.agregar( nuevo );
                system("pause>>cls");
                break;

            case '2': /// mostrar
                system("cls");
                factura.mostrarTodo();
                system("pause>>cls");
                break;

            case '3': /// buscar por nombre
                system("cls");
                cout << "\n\n\t\t\t\t\t Ingresa el nombre a buscar: ";
                fflush( stdin );
                getline( cin, respuesta );
                factura.buscarPorNombre( respuesta );
                system("pause>>cls");
                break;

            case '4': /// modificar
                system("cls");
                cout << "\n\n\t\t\t\t\t Ingrese el id del cliente a modificar: ";
                fflush( stdin );
                cin >> respuesta;
                if( factura.buscarId( atoi( respuesta.c_str() ), facturaBuscar ) )
                {
                    nuevo = factura.pedirRegistro();
                    factura.modificar( atoi( respuesta.c_str() ), nuevo );
                }
                else
                {
                    cout << "\n\n\t\t\t\t\t No existe el cliente" << endl;
                    system("pause>>cls");
                }
                break;
        }
    }
}

```

```

case '5': /// eliminar
    system("cls");
    cout << "\n\n\t\t Ingrese el RFC del cliente a eliminar: ";
    fflush( stdin );
    cin >> respuesta;
    if( factura.buscarId( atoi( respuesta.c_str() ), facturaBuscar ) )
    {
        if( factura.eliminar( atoi( respuesta.c_str() ), facturaEliminado ) )
        {
            cout << facturaEliminado << endl;
            cout << "\n\n\t\t CLIENTE ELIMINADO CON EXITO" << endl;
        }
        else
            cout << "\n\n\t\t NO se elimino al cliente" << endl;
    }
    else
        cout << "\n\n\t\t No existe el cliente" << endl;
    system("pause>>cls");
    break;

case '6':
    system("cls");
    factura.mostrarLista();
    system("pause>>cls");
    break;
}

}

while( opcion != '7' );

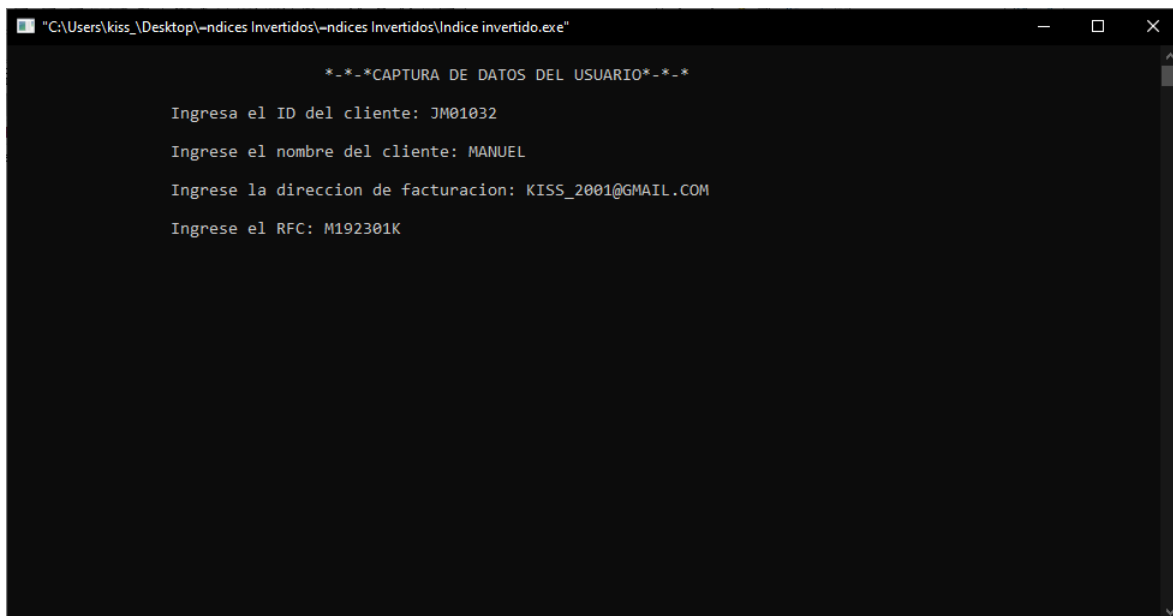
```

```
factura.guardar();  
system("cls");  
cout << "\n\n\t\t Saliendo del sistema....";  
system("pause>>cls");
```

```
return 0;
```

```
template < typename E >  
void mostrar( ListaLigada< E > &L )  
{  
    int posOriginal = L.posicionActual();  
    L.irAlInicio();  
  
    cout << "[";  
    for( int i = 0; i < L.obtenerTamanio(); i++ )  
    {  
        cout << L.obtenerElemento() << "-> " << L.obtenerEnlace();  
        if( !( i == ( L.obtenerTamanio() - 1 ) ) )  
        {  
            cout << ", ";  
            L.siguiente();  
        }  
    }  
    cout << "]" << endl;
```

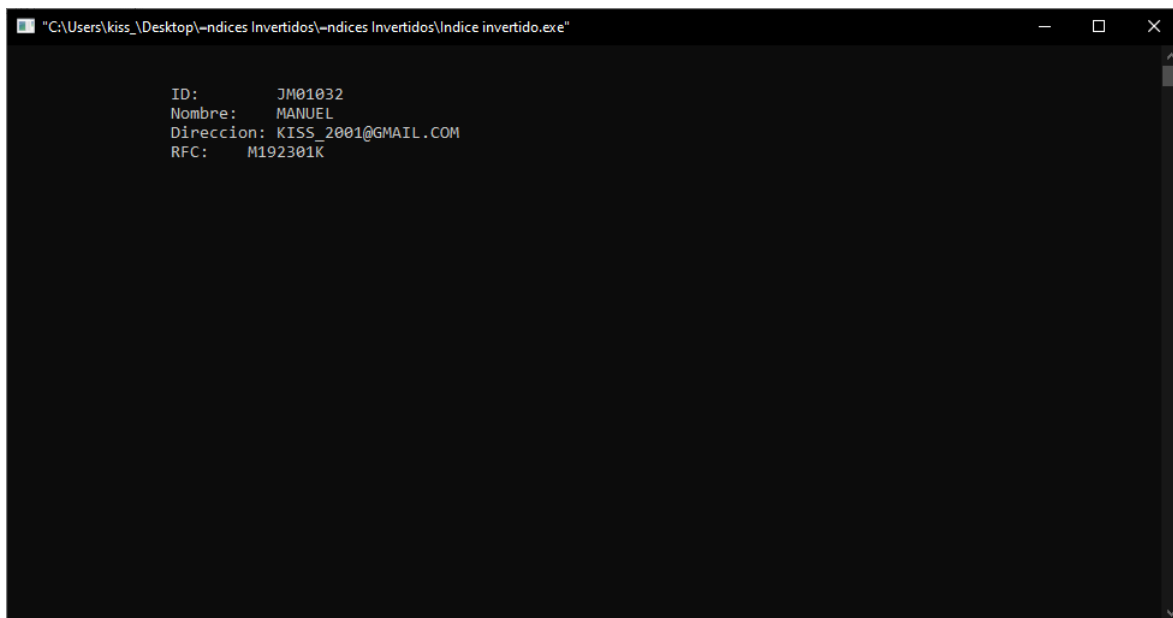
## Ejecución



```
"C:\Users\kiss\Desktop\~ndices Invertidos\~ndices Invertidos\Indice invertido.exe"

*-*~*CAPTURA DE DATOS DEL USUARIO*-*~*

Ingresa el ID del cliente: JM01032
Ingresa el nombre del cliente: MANUEL
Ingresa la direccion de facturacion: KISS_2001@GMAIL.COM
Ingresa el RFC: M192301K
```



```
"C:\Users\kiss\Desktop\~ndices Invertidos\~ndices Invertidos\Indice invertido.exe"

ID: JM01032
Nombre: MANUEL
Direccion: KISS_2001@GMAIL.COM
RFC: M192301K
```

