UNIVERSIDAD DE GUADALAJARA

INGENIERIA EN COMPUTACIÓN

SEMINARIO DE SOLUCION DE PROBLEMAS DE ESTRUCTURA DE DATOS II

Dispersión

INTEGRANTES:

Christopher Ceballos Jimenez

Código:

Clase Pagos

```
class Pagos
{
private:
    char id[ 12 ];
    char nombre[ 35 ];
    char membresia[ 35 ];
    char precio[ 10 ];
    char fecha [ 20 ];

    int dispersion( const char llave[ 12 ] );
    long int buscarId( const string & );
public:
    Pagos( void );
    ///Promocion( const Promocion & );
    void setId( const string & );
    void setNombre( const string & );
    void setMembresia( const string & );
    void setPrecio( const string & );
    void setFecha ( const string & );
    friend ostream &operator<<( ostream &, const Pagos & );

    bool agregar( const Pagos & );
    void mostrar( void );
    bool buscar( const string &, Pagos & );
    bool eliminar( const string &, Pagos & );
    bool modificar( const string &, const Pagos & );
    void mostrarIndice( void );
    Pagos pedirRegistro( void );
    bool contiene( const string & );
```

```
    bool contiene( const string & );
    void genera( void );
};

/// constructor
Pagos::Pagos( void )
{
    for( int i = 0; i < sizeof( id ); id[ i++ ] = '\0' );
    for( int j = 0; j < sizeof( nombre ); nombre[ j++ ] = '\0' );
    for( int k = 0; k < sizeof( membresia ); membresia[ k++ ] = '\0' );
    for( int i = 0; i < sizeof( precio ); precio[ i++ ] = '\0' );
    for( int l = 0; l < sizeof( fecha); fecha[ l++ ] = '\0' );
}

void Pagos::genera( void )
{
    Pagos promo;
    ofstream archivo( "dispersion.txt", ios::out );
    if( !archivo )
        cout << "No se pudo crear el archivo" << endl;
    else
        for( int i = 0; i < NUMREGISTROS; i++ )
            archivo.write( ( char * ) &promo, sizeof( Pagos ) );
    archivo.close();
}
```

```
bool Pagos::contiene( const string &idABuscar )
{
    if( buscarId( idABuscar ) == -1 )
        return false;
    return true;
}

Pagos Pagos::pedirRegistro( void )
{
    Pagos registroARetornar;
    string cadena;

    cout << endl << "\t\t ID: ";
    fflush( stdin );
    getline( cin, cadena );
    while( contiene( cadena ) )
    {
        cout << "\n\t\t Ese ID ya existe: ";
        fflush( stdin );
        getline( cin, cadena );
    }
    registroARetornar.setId( cadena );

    cout << "\n\t\t Nombre Cliente: ";
    fflush( stdin );
    getline( cin, cadena );
    registroARetornar.setNombre( cadena );


    cout << "\n\t\t Membresia: ";
    fflush( stdin );
    getline( cin, cadena );
    registroARetornar.setMembresia( cadena );

    cout << "\n\t\t Precio: ";
    fflush( stdin );
    getline( cin, cadena );
    registroARetornar.setPrecio( cadena );

    cout << "\n\t\t Fecha de Pago: ";
    fflush( stdin );
    getline( cin, cadena );
    registroARetornar.setFecha ( cadena );

    return registroARetornar;
}
```

```
int Pagos::dispersion( const char llave[ 12 ] )
{
    /// llena el sobrante de la llave con espacios
    char llaveCopia[ 12 ];
    strcpy( llaveCopia, llave );
    if( strlen( llaveCopia ) < 12 )
        for( int i = strlen( llaveCopia ); i < 12; i++ )
            llaveCopia[ i ] = ' ';
    llaveCopia[ 12 ] = '\0';

    /// realiza el algoritmo
    int sum = 0;
    int j = 0;
    while( j < 12 )
    {
        sum = ( sum + 100 * llaveCopia[ j ] + llaveCopia[ j + 1 ] )  % 20000;
        j += 2;
    }
    return ( sum % 99 ); /// retorna la posición donde se guardará el registros.
}
```

Métodos:

Agregar

```
bool Pagos::agregar( const Pagos &nuevoPago )
{
    int posIndice;
    long int posByte;
    string idString = nuevoPago.id;

    if( contiene( idString ) )
        return false;

    posIndice = dispersion( nuevoPago.id );
    cout << "Se guardara en la posicion indice: " << posIndice << endl;
    fstream archivo( "dispersion.txt", ios::in | ios
    posByte = posIndice * sizeof( Pagos );
    archivo.seekg( posByte, ios::beg );
    archivo.write( ( char * ) &nuevoPago, sizeof( Pagos ) );

    archivo.close();
    return false;
}
```

int Pagos::agregar::posIndice

```cpp
void Pagos::setId( const string &valorId )
{
    int longitud = valorId.size();
    longitud = ( longitud < 12 ? longitud : 11 );
    valorId.copy( id, longitud );
    id[ longitud ] = '\0';
}

void Pagos::setNombre( const string &valorNombre )
{
    int longitud = valorNombre.size();
    longitud = ( longitud < 35 ? longitud : 34 );
    valorNombre.copy( nombre, longitud );
    nombre[ longitud ] = '\0';
}

void Pagos::setMembresia( const string &valorMembresia )
{
    int longitud = valorMembresia.size();
    longitud = ( longitud < 35 ? longitud : 34 );
    valorMembresia.copy( membresia, longitud );
    membresia[ longitud ] = '\0';
}

void Pagos::setPrecio( const string &valorPrecio )
{
    int longitud = valorPrecio.size();
    longitud = ( longitud < 10 ? longitud : 9 );
    valorPrecio.copy( precio, longitud );
    precio[ longitud ] = '\0';
}

void Pagos::setFecha( const string &valorFecha)
{
    int longitud = valorFecha.size();
    longitud = ( longitud < 11 ? longitud : 10);
    valorFecha.copy( fecha, longitud);
    fecha[ longitud ] = '\0';
}

ostream &operator<<( ostream &salida, const Pagos &Pagos )
{
    salida << "\n\t\t ID:         " << Pagos.id << endl
           << "\t\t Producto:   " << Pagos.nombre << endl
           << "\t\t Membresia: " << Pagos.membresia << endl
           << "\t\t Precio:     " << Pagos.precio << endl
           << "\t\t Fecha de Pago: " << Pagos.fecha << endl;

    return salida;
}
```

Imprimir

```
void Pagos::mostrar( void )
{
    Pagos pago;
    ifstream archivo( "dispersion.txt", ios::in );
    if( !archivo )
        cout << "No existe el archivo" << endl;
    else
        for( int i = 0; i < NUMREGISTROS; i++ )
        {
            archivo.read( ( char * ) &pago, sizeof( Pagos ) );
            if( !( pago.id[ 0 ] == '\0' ) )
                cout << endl << pago << endl;
        }
    archivo.close();
}
```

Buscar

```
bool Pagos::buscar( const string &idABuscar, Pagos &pagoEncontrado )
{
    int posIndice;
    long int posByte;

    if( !contiene( idABuscar ) )
        return false;

    ifstream archivo( "dispersion.txt", ios::in );
    if( !archivo )
    {
        cout << "El archivo no existe" << endl;
        archivo.close();
        return false;
    }

    posByte = buscarId( idABuscar );
    archivo.seekg( posByte, ios::beg );
    archivo.read( ( char * ) &pagoEncontrado, sizeof( Pagos ) );
    archivo.close();
    return true;
}
```

```
long int Pagos::buscarId( const string &idABuscar )
{
    Pagos promo;
    int posIndice;
    long int posByte;

    ifstream archivo( "dispersion.txt", ios::in );
    if( !archivo )
    {
        archivo.close();
        return -1;
    }
    else
    {
        posIndice = dispersion( idABuscar.c_str() );
        posByte = posIndice * sizeof( Pagos );
        archivo.seekg( posByte, ios::beg );
        archivo.read( ( char * ) &promo, sizeof( Pagos ) );
        if( strcmp( promo.id, idABuscar.c_str() ) == 0 )
        {
            archivo.close();
            return posByte;
        }
    }
    archivo.close();
    return -1;
}
```

Modificar

```
bool Pagos::modificar( const string &idAModificar, const Pagos &pagoNuevo )
{
    Pagos pago;
    long int posByteAnterior, posByteNuevo;
    int posRegAnterior;

    if( !contiene( idAModificar ) )
        return false;

    fstream archivo( "dispersion.txt", ios::in | ios::out );
    if( !archivo )
    {
        cout << "El archivo no exite" << endl;
        archivo.close();
        return false;
    }

    posRegAnterior = dispersion( pagoNuevo.id );
    posByteNuevo = posRegAnterior * sizeof( Pagos );

    posByteAnterior = buscarId( idAModificar );

    archivo.seekp( posByteAnterior, ios::beg );
    archivo.write( ( char * ) &pago, sizeof( Pagos) );
    archivo.seekp( posByteNuevo, ios::beg );
    archivo.write( ( char * ) &pagoNuevo, sizeof( Pagos ) );
    archivo.close();
    return true;
}
```

Eliminar

```
bool Pagos::eliminar( const string &idAEliminar, Pagos &destinoEliminado )
{
    Pagos pago;
    if( !contiene( idAEliminar ) )
        return false;

    fstream archivo( "dispersion.txt", ios::in | ios::out );
    if( !archivo )
    {
        cout << "El archivo no exite" << endl;
        archivo.close();
        return false;
    }

    long int posByte = buscarId( idAEliminar );
    archivo.seekg( posByte, ios::beg );
    archivo.read( ( char * ) &destinoEliminado, sizeof(  Pagos ) );
    archivo.seekg( posByte, ios::beg );
    archivo.write( ( char * ) &pago, sizeof(  Pagos ) );
    archivo.close();
    return true;
}
```

Menú

```
int main( void )
{
    Pagos pago, pagoBuscar, pagoModificar, pagoEliminar, pagoAgregar;
    string idABuscar, idAModificar, idAEliminar;
    char opcion[10], op;

    pago.genera();

    do
    {
        system( "cls" );
        cout<<"\n\n\t\t *-*-*Bienvenido al menu Pagos*-*-*\n"
            << "\t\t [1]- Agregar"    << endl
            << "\t\t [2]- Mostrar"    << endl
            << "\t\t [3]- Buscar"     << endl
            << "\t\t [4]- Modificar"  << endl
            << "\t\t [5]- Eliminar"   << endl
            << "\t\t [6]- Salir"      << endl
            << "\t\t Seleccione la opcion: ";
        cin >> opcion;

        if(strcmp(opcion, "1") == 0)
        {
            system("cls");
            pagoAgregar = pago.pedirRegistro();
            if( pago.agregar( pagoAgregar ) )
                cout << "Cliente agregado con exito" << endl;
            else
                cout << "No se agrego la promocion" << endl;
```

```
            system("pause>>cls");
        }

        if(strcmp(opcion, "2") == 0)
        {
            system( "cls" );
            pago.mostrar();
            system("pause>>cls");
        }

        if(strcmp(opcion, "3") == 0)
        {
            system("cls");
            cout << "\n\n\t Ingrese el ID del pago a buscar: ";
            fflush( stdin );
            getline( cin, idABuscar );
            if( pago.buscar( idABuscar, pagoBuscar ) )
                cout << endl << pagoBuscar << endl;
            else
                cout << "\n\t No existe el registro" << endl;
                system("pause>>cls");
        }

        if(strcmp(opcion, "4") == 0)
        {
            system("cls");
            cout << "\n\n\t Ingrese el ID del pago a modificar: ";
            fflush( stdin );
            getline( cin, idAModificar );
           if( pago.buscar( idAModificar, pagoBuscar ) )
           {
               cout << endl << pagoBuscar << endl;
               cout << "\n\t ¿Desea modificarlo?" << endl;
               cout << "\t [1] Si" << endl;
               cout << "\t [2] No" << endl;
               cin >> op;

               if( op == '1' )
               {
                   pagoModificar = pago.pedirRegistro();
                   if( pago.modificar( idAModificar, pagoModificar ) )
                       cout << endl << "\n\t Promocion modificada con exito" << endl;
                   else
                       cout << endl << "\n\t No se pudo modificar" << endl;
               }
           }
           else
               cout << "\n\t No existe el registro" << endl;
               system("pause>>cls");
        }

        if(strcmp(opcion, "5") == 0)
        {
            system("cls");
            cout << "\n\n\ Ingrese el ID del pago a eliminar: ";
            fflush( stdin );
            getline( cin, idAEliminar );
            if( pago.buscar( idAEliminar, pagoBuscar ) )
```

```
            if( pago.buscar( idAEliminar, pagoBuscar ) )
            {
                cout << endl << pagoBuscar << endl;
                cout << "\n\t Desea eliminarlo?" << endl;
                cout << "\t [1] Si" << endl;
                cout << "\t [2] No" << endl;
                cin >> op;

                if( op == '1' )
                    if( pago.eliminar( idAEliminar, pagoEliminar ) )
                        cout << endl << "\n\t" << pagoEliminar << endl << " SE ELIMINO CORRECTAMENTE" << endl;
                    else
                        cout << "\n\t NO SE PUDO ELIMINAR" << endl;
            }
            else
                cout << "\n\t No existe el registro" << endl;
                system("pause>>cls");
        }
        if(strcmp(opcion, "6") == 0)
        {
            break;
        }

    }
    while(opcion != "6");
    system("cls");
    cout << "\n\n\t\t Saliendo del Sistema.....";
    system("pause>>cls");
}
```

Ejecución

Menú:

```
      *-*-*Bienvenido al menu Pagos*-*-*
      [1]- Agregar
      [2]- Mostrar
      [3]- Buscar
      [4]- Modificar
      [5]- Eliminar
      [6]- Salir
      Seleccione la opcion:
```

Opción 1:

```
                ID: 134679

                Nombre Cliente: diego

                Membresia: premium

                Precio: 56

                Fecha de Pago: 24
Se guardara en la posicion indice: 13
No se agrego la promocion
```

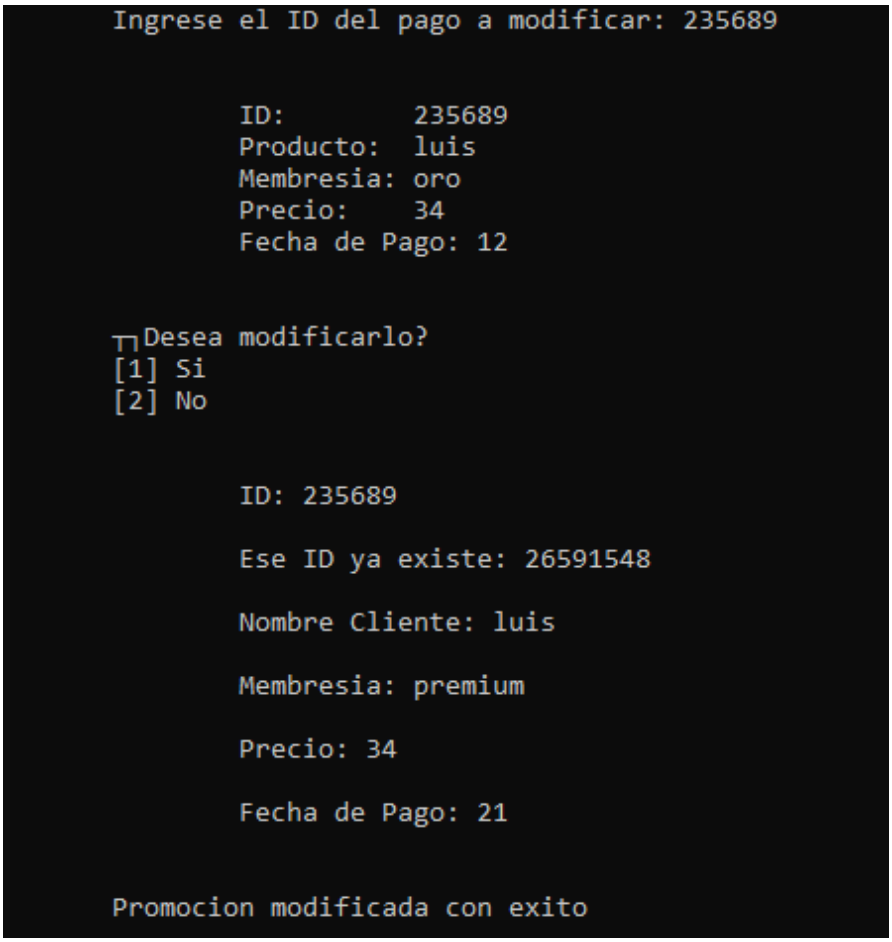
Opción 2:

```
ID:         124578
Producto:   jorge
Membresia:  basica
Precio:     12
Fecha de Pago: 23


ID:         134679
Producto:   diego
Membresia:  premium
Precio:     56
Fecha de Pago: 24


ID:         235689
Producto:   luis
Membresia:  oro
Precio:     34
Fecha de Pago: 12
```

Opción 3:

```
Ingrese el ID del pago a buscar: 235689

        ID:         235689
        Producto:   luis
        Membresia:  oro
        Precio:     34
        Fecha de Pago: 12
```

Opción 4:


Ingrese el ID del pago a modificar: 235689
ID: 235689
Producto: luis
Membresia: oro
Precio: 34
Fecha de Pago: 12
Desea modificarlo?
[1] Si
[2] No
ID: 235689
Ese ID ya existe: 26591548
Nombre Cliente: luis
Membresia: premium
Precio: 34
Fecha de Pago: 21
Promocion modificada con exito

Opción 5:

```
Ingrese el ID del pago a eliminar: 26591548


                ID:         26591548
                Producto:   luis
                Membresia:  premium
                Precio:     34
                Fecha de Pago: 21


        Desea eliminarlo?
        [1] Si
        [2] No
1



                ID:         26591548
                Producto:   luis
                Membresia:  premium
                Precio:     34
                Fecha de Pago: 21

SE ELIMINO CORRECTAMENTE
```

Opción 6:

```
        Saliendo del Sistema.....
```