Universidad Nororiental Privada "Gran Mariscal de Ayacucho"

Facultad: Ingeniería En Informática y Sistemas.

Materia: Programacion ||

Sección: 3D1

Ejercicio Tercer corte

Diners Bank

Profesor:

Thays Parra.

Bachilleres:

Jamal souki

C.I: 31.522.107

Keysha Montes de Oca.
C.I: 30.648.421.

Ciudad Guayana 10/12/2021

Planteamiento:

El Banco Diners Bank, desea automatizar todas sus operaciones bancarias. Por tal razón los ha contratado, con la finalidad de diseñar una estructura de datos en C++ que permita manejar de una manera fácil y eficiente todas sus operaciones considerando los siguientes requerimientos:

a) El banco maneja básicamente tres (03) tipos de cuenta (aunque su modelo deberá permitir el ingreso de muchos más): Cuentas de Ahorro, Corriente y de Activos Líquidos, además por cada una de ellas deberá llevarse la tasa de rendimiento correspondiente.
b) Por cada cliente se deberá conocer: Nro. de la Cuenta, Tipo de Cuenta, Nombre, Dirección, Teléfono, Saldo en la Cuenta, Cédula de Identidad, etc.
c) Deberá existir un registro de todas las transacciones realizadas por el banco: Código de la transacción, Fecha de la misma, Número de cuenta involucrada Monto en Bolívares y el número de la caja donde fue realizada.

Con el modelo diseñado, programar lo siguiente:

a) Obtener todos los clientes con Cuenta Corriente
b) Mostrar todas las transacciones que involucran a una cuenta de ahorros dada por el usuario desde una fecha hasta otra.

Mostrar las transacciones de Cobro y de depósito de una caja dada por el usuario.

Algoritmo General de la solución:

1-Declaramos las estructuras de datos: usuario y Actividad_De_cuenta.

2-Declaramos las variables de tipo entero Numero_Clientes, Numero_Cuentas y Numero_Transacciones: seteadas en 0 estas almacenan el numero de clientes, cuentas y transacciones.

3-Creamos la función void gotoxy(int x,int y) que se encarga de poner el cursor en el programa de forma en un lugar diferente

4-Definimos void menu(); al inicio para lograr llamarlo sin necesidad de que la función este al inicio del programa

5-Creamos la función int mostrar_saldos(int Numero_Cliente, int linea) : este usando if verifica que el número de cuenta de ahorro es mayor a 0 ósea si el usuario tiene si es muestra el saldo y el número de la cuenta, lo mismo se realiza con las otras 2 cuentas se crea un if que verifique eso y muestra en caso de ser mayor a 0, esa función va sumando la línea para que el saldo salga en línea diferente y no se vea mal y luego devuelve el número de la línea actual para que el programa siga usando.

6-Creamos la función void Transaccion(int actividad, int Numero_Clie):En esta en int numero_clie se guarda el numero del cliente y en int actividad depende de la opción del usuario que elija pueden ser dos deposito seria actividad == 1 o retiro que sería actividad == 2. Definimos float monto; guarda el monto que el usuario retirara o d epositara, luego atraves de un ciclo que pase por las estructura de

actividad de cuentas buscando una que su número de actividad sea 0 para así encontrar un espacio no usado y guardar el registro una vez encontrado se da la bienvenida al usuario se crea y se asigna valor de 5 a la variable línea (esta almacena la línea del mensaje del gotoxi) luego se llama a la función mostrar_saldos(Numero_clie) para mostrar los saldos de las cuentas que tiene el usuario y esta luego de mostrar devuelve el numero de línea donde quedo luego de eso el programa te pide que ingreses el monto, y suma 1 a la variable línea para que el mensaje se muestre una línea mas abajo, luego el programa le pide que ingrese el numero de caja, el dia, mes, año y va sumando una línea para que quede una línea debajo de otra, luego el programa te da a elegir cual cuenta hiciste la transacción dependiendo si tienes o no una cuenta, mientras el usuario ingresa los datos los va guardando en el struct y luego en switch dependiendo si la actividad es retiro o deposito y el tipo de cuenta si el usuario tiene o no, si es retiro o deposito el programa guarda el saldo actual y el monto a depositar/retirar si es correcto al final se le asigna un numero a la transacción y cliente . en caso de que haya un fallo el programa le pone al numero de transacción en 0 para que el programa no tome los datos registrados y se pueda usar ese espacio sobrescribiendo todo los datos de la transacción fallida.

7-Creamos la función void MostrarCaja(int Numero_Cliente): Definimos las variables de tipo entero línea que sería donde se almacena la línea que usara el gotoxi y la variable caja donde almacena el número de la caja que será visualizado todos las actividades que el cliente realizo en ella, le pedimos al cliente la caja donde realizo la transacción y la guardamos en la variable Caja luego imprimimos con gotoxi una parte para el Numero cuenta, monto, tipo de cuenta, Numero de transacción, el tipo de actividad si es retiro(-) o deposito(+) y la fecha luego con for que pasa por todas las actividades, este toma las que tienen un número de transacción asignado luego verifica si el cliente que hizo la transacción es el mismo y luego la caja si el mismo, el programa imprime en las líneas que se van sumando, los valores del Numero cuenta, monto, tipo de cuenta, número transacción, tipo transacción(+/-), Fecha luego de esto el programa se frena con system(“pause”) para que el usuario logre visualizar.

8-Creamos la función void MovimientoFHF(int Numero_Cliente): Creamos varias variables línea(Almacena la línea que usa el gotoxi) Luego DiaI,MesI,DiaI esta es la fecha desde donde va a buscar y DiaF, MesF,AnoF seria la fecha hasta donde el programa va a buscar, Imprimimos pidiéndole al usuario que ingrese la fecha desde donde va a buscar y ordenada mente mientras el usuario ingresa los datos se va imprimiendo y guardando en las variables mencionadas anteriormente luego se le pide al usuario la fecha hasta donde se va a buscar luego de eso imprime de forma ordenada lo que va a imprimir luego , luego el programa va buscando en las actividades en las estructura que no tenga numero de transaccion, luego verifica la fecha de la transacción usando los datos pedidos anteriormente de forma que si Act_Cuenta[c(Numero del reg)].Dia es mayor o igual al DiaI y si es menor que

DiaF de estas forma busca un día que este entre las fechas dadas y lo mismo ocurre con el mes y año, y que esta transaccion sea de la cuenta de ahorro y en caso de cumplir estas condiciónes el programa verifica que ese registro sea de ese cliente, luego el programa imprime los datos(Numero de cuenta, Monto, Tipo de cuenta, Numero de transacción, actividad, fecha) de las transacciones que cumplan con lo anterior mencionado por cada transacción una vez que termina el programa a la variable línea le suma 1 para así la otra aparezca debajo de esa línea.

9-Creamos la función void MenuCliente(int Numero_Cliente) Esta función es llamada en la función logear cuando ingresa la cedula de un cliente y la llama con el Numero del cliente, lo primero creamos una variable tipo entero select donde almacena la selección del cliente luego borramos la consola y mostramos el mensaje de bienvenida luego creamos y definimos la variable tipo entero linea como 5 luego llamamos a mostrar_saldos(Numero_cliente) para mostrar los saldos de las cuentas del cliente y luego se almacena en la variable linea el valor que devuelve el número de la línea donde quedo, luego se verifica si el usuario tiene una cuenta de ahorro verificando si el numero de la cuenta de ahorro es mayor a 0 en caso de que sea 0 el programa imprime" -- Opcion 1 No Disponible-- Movimientos de la cuenta ahorro desde una fecha. NO POSEE CUENTA DE AHORRO" en caso de tener el programa imprime 1. Movimientos de la cuenta ahorro desde una fecha hasta otra fecha.en caso de que la opción seleccionada sea la 1 que el usuario no tenga cuenta ahorro el programa regresa a este mismo menu luego imprime las otras opciones disponibles 2 registrar retiro, 3. Registrar Deposito, 4.ver deposito y retiro desde una caja determinada, en caso de seleccionar la opción 1: se verifica si el usuario tiene cuenta ahorro y si tiene se llama a la función MovimientoFHF(Usando el numero del cliente) y en caso de no tener lo devuelve al mismo menú. 2: Llama a la función Transaccion(2, Numero del cliente) es 2 el primer parámetro el cual indica que la transaccion es un retiro. 3: Llama a la función Transaccion(1, Numero del cliente) es 1 el primer parámetro el cual indica que la transaccion es un Deposito. 4: llama a la función MostrarCaja(con numero del cliente). 5: Devuelve la funcion menú(); enviando al menú principal.

10- Creamos la funcion void RegistrarCliente(): Creamos la variable tipo char SoN[1] luego cleamos un ciclo for el cual pasa por la estructura de datos de usuario buscando un espacio vacio osea que el numero del cliente sea 0 al encontrar le da la bienvenida al cliente le solicita el Nombre (c es el numero del cliente) cin >> user[c].Nombre;, luego solicita la cedula guardándola en user[c].Cedula, Luego solicita el teléfono y lo guarda en user[c].Telefono;, luego la dirección y la guarda en user[c]. Direccion;, luego le pregunta si desea tener una cuenta corriente el usuario responde esto lo guarda en la variable SoN luego s usando tolower se vuelve en minúscula "s" si es "s" el programa le suma 1 a la variable Numero_Cuentas y ese valor se lo da a la cuenta user[c].Num_CuentaC

y le pide el saldo al usuario guardándolo en user[c].SaldoC, luego sucede lo mismo pregunta al usuario si va a querer una cuenta de ahorro le pregunta el saldo el cual se guarda en user[c].SaldoA y le da un número a la cuenta guardado en user[c].Num_CuentaA y luego lo mismo con la cuenta de activos guarda el saldo en user[c].SaldoAL y el numero user[c].Num_CuentaAL y va sumando las líneas en la variable linea para que valla imprimiendo una línea debajo de otra y al final la variable Numero_cliente se le suma 1 y se le asigna ese numero al cliente al asignarla a la variable user[c].Num_Cliente.

11-Creamos la funcion void LogearCliente(): Creamos las variables de tipos char cedula[20] y SoN[1] Limpiamos la pantalla del programa con system("cls"); Mostramos el mensaje de bienvenida y le pedimos la cedula al usuario

gotoxy(17,6);cout<<"Ingrese su cedula:";

gotoxy(37,6); cin >> cedula; guardamos en la variable cedula

Luego usando un for pasamos por las cedulas de los usuarios hasta encontrar una que consida con la dada esta comparación es echa con strcmp en caso de encontrar se envía MenuCliente(c(Numero del usuario)); y si el numero del contador llega al numero máximo de usuarios significa que no encontró se le pregunta al usuario si quiere reintentar o salir guardando la elección en SoN y luego si es diferente a “s” el programa vuelve a enviarlo a logarCliente() para que ponga denuevo su cedula, si es “s” este envía al menú principal menu();

12-Creamos la funcion void MostrarCuentasCorrientes(): creamos la variable linea de tipo enterio y le damos valor 5, luego limpiamos la pantalla del programa y mostramos Clientes con cuenta corriente atraves de un ciclo for el cual pasa por los usuarios y comprueba que if(user[c].Num_CuentaC > 0) si tiene cuenta corriente muestra sus datos y luego de imprimir los datos le suma 1 a la variable linea para que aparesca abajo y luego al final del ciclo se le suma 1 tambien a la variable linea para que el system(“pause”); aparesca 2 lineas debajo .

13-Creamos la funcion void menu():Creamos una variable de tipo entero select la cual guarda la selección luego borramos la pantalla del programa y le damos el mensaje de bienvenida al usuario luego le pedimos que seleccione una opción:

1. Registrar Cliente:Llama a la funcion Registrarcliente() para registrar el mismo y luego a la funcion menú para que llame a este al terminar la funcion de registrar

2. Entrar como Cliente: Llama a la funcion LogearCliente() para entrar como cliente con la cedula luego llama a la funcion menú(); para devolver al menu al terminar

3. Mostrar lista de clientes con cuenta corriente: Llama ala funcion MostrarCuentasCorrientes(); para mostrar las cuentas corriente de los usuarios registrados

4. Salír: Limpia la pantalla del programa y imprime el mensaje de cerrando luego de 2000ms el programa se cierra

14-int main: es llamado al inicial el programa y este llama al menú principal menú();

Código de fuente comentado:

```
#include <cstdlib>
#include <iostream>
#include <conio.h>
#include <windows.h>

using namespace std;

//Constantes y variables

#define MAX_Usuarios        100000
#define MAX_Actividad       500000

int Numero_Clientes  = 0;
int Numero_Cuentas  = 0;
int Numero_Transacciones  = 0;

//Funcion gotoxy y llamado a la funcion menu();
void gotoxy(int x,int y){
    HANDLE hcon;
    hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwPos;
    dwPos.X = x;
    dwPos.Y= y;
    SetConsoleCursorPosition(hcon,dwPos);
}
void menu();
```

```
//estructuras inicio
struct usuario{
    char Nombre[24];
    int Num_Cliente;
    int Num_CuentaC;
    int Num_CuentaA;
    int Num_CuentaAL;
    float SaldoC;
    float SaldoA;
    float SaldoAL;
    char Telefono;
    char Cedula[20];
    char Direccion[50];
};
usuario user[MAX_Usuarios];

struct Actividad_De_cuenta{
    int Numero_Cliente;
    int Num_Cuenta;
    int TipoCuenta;//1 = corriente, 2 = ahorro, 3 activos liquidos
    int Num_Transaccion;
    int Caja_usada;
    int Actividad;//1= deposito, 2 = Retiro
    float Monto;
    float Saldo;
    int Dia;
    int mes;
    int ano;
};
Actividad_De_cuenta  Act_Cuenta[MAX_Actividad];
//Estructuras fin
```

```
//muestra los saldos de las cuentas de los usuarios
int mostrar_saldos(int Numero_Cliente, int linea)
{
    if(user[Numero_Cliente].Num_CuentaC >0){
    gotoxy(15,linea); cout << "Saldo Actual cuenta Corriente: "<< user[Numero_Cliente].SaldoC;
    linea ++;
    gotoxy(15,linea); cout << "Numero cuenta Corriente: "<< user[Numero_Cliente].Num_CuentaC;
    linea ++;
    }
    if(user[Numero_Cliente].Num_CuentaA >0){
    gotoxy(15,linea); cout << "Saldo Actual cuenta Ahorro: "<< user[Numero_Cliente].SaldoA;
    linea ++;
    gotoxy(15,linea); cout << "Numero cuenta Ahorro: "<< user[Numero_Cliente].Num_CuentaA;
    linea ++;
    }
    if(user[Numero_Cliente].Num_CuentaAL>0){
    gotoxy(15,linea); cout << "Saldo Actual cuenta Activos Liquidos: "<<
user[Numero_Cliente].SaldoAL;
    linea ++;
    gotoxy(15,linea); cout << "Numero cuenta Ahorro: "<< user[Numero_Cliente].Num_CuentaAL;
    linea ++;
    }
    linea ++;
    return linea;
}

//TRansacciones deposita y retiro
void Transaccion(int actividad, int Numero_Clie)
{
    float monto;
```

```
for(int c = 0; c < MAX_Actividad; c++)
{
    if(Act_Cuenta[c].Num_Cuenta == 0)
    {
        system("cls");
        gotoxy(25,2); cout << "Diners Bank";
        gotoxy(18,3); cout << "Bienvenid@: "<< user[Numero_Clie].Nombre;

        int linea = 5;

        linea = mostrar_saldos(Numero_Clie, linea);

        switch(actividad)
        {
            case 1://depositar
            {
                gotoxy(8,linea);cout<<"Ingrese el monto a Depositar Bs:";
                break;
            }
            case 2://retirar
            {
                gotoxy(8,linea);cout<<"Ingrese el monto a Retirar Bs:";
                break;
            }
        }
        cin >> monto;
        linea ++;

        gotoxy(8,linea);cout<<"Ingrese el numero de la caja:";
        cin >> Act_Cuenta[c].Caja_usada;
        linea ++;
```

```
gotoxy(8,linea);cout<<"Ingrese  el dia de la transaccion:";
cin >> Act_Cuenta[c].Dia;
linea ++;

gotoxy(8,linea);cout<<"Ingrese  el mes de la transaccion:";
cin >> Act_Cuenta[c].mes;
linea ++;

gotoxy(8,linea);cout<<"Ingrese  el año de la transaccion:";
cin >> Act_Cuenta[c].ano;
linea ++;

gotoxy(8,linea);cout<<"Ingrese  cual cuenta realizo la transaccion:";
if(user[Numero_Clie].Num_CuentaC  >0)cout<<" 1.Corriente  |";
if(user[Numero_Clie].Num_CuentaA  >0)cout<<" 2.Ahorro | ";
if(user[Numero_Clie].Num_CuentaAL  >0)cout<<" 3.Activos liquidos: ";
cin >> Act_Cuenta[c].TipoCuenta;
linea ++;

switch(actividad)
{
   case 1://depositar
   {
      switch(Act_Cuenta[c].TipoCuenta)
      {
         case 1:
         {
         user[Numero_Clie].SaldoC  = user[Numero_Clie].SaldoC  + monto;
         Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaC;

         Numero_Transacciones++;
```

```
                Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;


                Act_Cuenta[c].Monto = monto;
                Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoC;
                gotoxy(8,linea);cout <<"Saldo actual cuenta
Corriente:"<<user[Numero_Clie].SaldoC;
                linea++;
                break;
                }
                case 2:
                {
                user[Numero_Clie].SaldoA  = user[Numero_Clie].SaldoA  + monto;
                Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaA;


                Numero_Transacciones++;
                Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;


                Act_Cuenta[c].Monto = monto;
                Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoA;
                gotoxy(8,linea);cout <<"Saldo actual cuenta Ahorro:"<<user[Numero_Clie].SaldoA;
                linea++;
                break;
                }
                case 3:
                {
                user[Numero_Clie].SaldoAL  = user[Numero_Clie].SaldoAL  + monto;
                Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaAL;


                Numero_Transacciones++;
                Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;


                Act_Cuenta[c].Monto = monto;
```

```
                    Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoAL;
                    gotoxy(8,linea);cout <<"Saldo actual Activos
Liquidos:"<<user[Numero_Clie].SaldoAL;
                    linea++;
                    break;
                    }
                    default:
                    {
                        cout<<"\n\nLa opcion que usted ha seleccionado no esta disponible"<<endl;
                        cout<<"Transaccion ERRONEA."<<endl;
                        Act_Cuenta[c].Num_Cuenta  = 0;
                        system("pause");
                        menu();
                    }
                }

                linea ++;
                gotoxy(8,linea);cout<<"Transaccion  correcta. :D";

                linea ++;

                Act_Cuenta[c].Numero_Cliente  = Numero_Clie;

                gotoxy(8,linea);cout<<"Numero  de transaccion: #"<<Act_Cuenta[c].Num_Transaccion;
                gotoxy(8,linea);cout<<endl;
                system("pause");
                break;
            }
            case 2://retiro
            {
                switch(Act_Cuenta[c].TipoCuenta)
                {
```

```
                case 1:
                {
                   if(monto > user[Numero_Clie].SaldoC)
                   {
                      cout<<"\n\nError usted no tiene esa cantidad de dinero a retirar"<<endl;
                      cout<<"Transaccion ERRONEA."<<endl;
                      Act_Cuenta[c].Num_Cuenta = 0;
                      system("pause");
                      menu();
                   }
                   user[Numero_Clie].SaldoC  = user[Numero_Clie].SaldoC  - monto;
                   Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaC;

                   Numero_Transacciones++;
                   Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;

                   Act_Cuenta[c].Monto = monto;
                   Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoC;
                   gotoxy(8,linea);cout <<"Saldo actual cuenta
Corriente:"<<user[Numero_Clie].SaldoC;
                   linea ++;
                   break;
                }
                case 2:
                {
                   if(monto > user[Numero_Clie].SaldoA)
                   {
                      cout<<"\n\nError usted no tiene esa cantidad de dinero a retirar"<<endl;
                      cout<<"Transaccion ERRONEA."<<endl;
                      Act_Cuenta[c].Num_Cuenta = 0;
                      system("pause");
                      menu();
```

```
                    }
                    user[Numero_Clie].SaldoA  = user[Numero_Clie].SaldoA  - monto;
                    Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaA;

                    Numero_Transacciones++;
                    Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;

                    Act_Cuenta[c].Monto = monto;
                    Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoA;
                    gotoxy(8,linea);cout <<"Saldo actual cuenta
Ahorro:"<<user[Numero_Clie].SaldoA;
                    linea ++;
                    break;
                }
                case 3:
                {
                    if(monto > user[Numero_Clie].SaldoAL)
                    {
                        cout<<"\n\nError usted no tiene esa cantidad de dinero a retirar"<<endl;
                        cout<<"Transaccion ERRONEA."<<endl;
                        Act_Cuenta[c].Num_Cuenta = 0;
                        system("pause");
                        menu();
                    }
                    user[Numero_Clie].SaldoAL  = user[Numero_Clie].SaldoAL  - monto;
                    Act_Cuenta[c].Num_Cuenta  = user[Numero_Clie].Num_CuentaAL;

                    Numero_Transacciones++;
                    Act_Cuenta[c].Num_Transaccion  = Numero_Transacciones;

                    Act_Cuenta[c].Monto = monto;
                    Act_Cuenta[c].Saldo = user[Numero_Clie].SaldoAL;
```

```
                    gotoxy(8,linea);cout <<"Saldo actual Activos
Liquidos:"<<user[Numero_Clie].SaldoAL;
                    linea ++;
                    break;
                }
                default:
                {
                    cout<<"\n\nLa opcion que usted ha seleccionado no esta disponible"<<endl;
                    cout<<"Transaccion ERRONEA."<<endl;
                    Act_Cuenta[c].Num_Cuenta  = 0;
                    system("pause");
                    menu();
                }
            }

            linea ++;
            gotoxy(8,linea);cout<<"Transaccion  correcta. :D";

            linea ++;

            gotoxy(8,linea);cout<<"Numero  de transaccion: #"<<Act_Cuenta[c].Num_Transaccion;
            gotoxy(8,linea);cout<<endl;
            system("pause");
            break;
          }
        }
      break;
      }
   }
}
```

```
//Muestra las cuentas corriente existentes
void MostrarCaja(int Numero_Cliente)
{
    int linea= 7, Caja;
    system("cls");
    gotoxy(10,3);cout<<"Seleccione la caja la cual desea ver sus transacciones realizadas:";
    cin >>Caja;
    gotoxy(10,4);cout<<"Numero";
    gotoxy(10,5);cout<<"Cuenta:";
    gotoxy(24,4);cout<<"Monto:";
    gotoxy(32,4);cout<<"Tipo de";
    gotoxy(32,5);cout<<"cuenta:";
    gotoxy(50,4);cout<<"Numero";
    gotoxy(50,5);cout<<"Transaccion:";
    gotoxy(66,4);cout<<"+/-";//+ o -
    gotoxy(80,4);cout<<"Fecha";

    for(int c = 0; c < MAX_Actividad; c++)
            {
            if(Act_Cuenta[c].Num_Transaccion > 0)
            {
            if(Act_Cuenta[c].Numero_Cliente == Numero_Cliente)
                {
            if(Act_Cuenta[c].Caja_usada == Caja)
            {
                    gotoxy(10,linea);cout<<Act_Cuenta[c].Num_Cuenta;
                    gotoxy(24,linea);cout<<Act_Cuenta[c].Monto;

                    if(Act_Cuenta[c].TipoCuenta == 1)
                    {
                        gotoxy(32,linea);cout<<"Corriente";
                    }
```

```
                else if(Act_Cuenta[c].TipoCuenta  == 2)
                {
                    gotoxy(32,linea);cout<<"Ahorro";
                }
                else if(Act_Cuenta[c].TipoCuenta  == 3)
                {
                    gotoxy(32,linea);cout<<"Activos  Liquidos";
                }

                gotoxy(50,linea);cout<<Act_Cuenta[c].Num_Transaccion;

                if(Act_Cuenta[c].Actividad  == 1)
                {
                    gotoxy(66,linea);cout<<"+";
                }
                else
                {
                    gotoxy(66,linea);cout<<"-";
                }
                gotoxy(82,linea);cout<<Act_Cuenta[c].Dia<<"/"  <<Act_Cuenta[c].mes<<"/"
<<Act_Cuenta[c].ano;
                linea++;
              }
           }
        }
    }
    linea++;
    gotoxy(17,linea);system("pause");
}
//detecta los movimientos  de una fecha hasta otra
void  MovimientoFHF(int   Numero_Cliente)
{
```

```
    int linea= 14, DiaI,MesI,AnoI,  DiaF,  MesF,AnoF;

    system("cls");

    gotoxy(10,3);cout<<"Ingrese  la fecha inicial desde donde vera sus transacciones de la cuenta de ahorro:";

    gotoxy(10,5);cout<<"Dia";

    gotoxy(10,6);cin >>DiaI;

    gotoxy(12,5);cout<<"/";

    gotoxy(13,5);cout<<"Mes";

    gotoxy(13,6);cin >>MesI;

    gotoxy(15,5);cout<<"/";

    gotoxy(17,5);cout<<"Año";

    gotoxy(17,6);cin >>AnoI;


    gotoxy(10,7);cout<<"Ingrese  la fecha Final hasta donde vera sus transacciones de la cuenta de ahorro:";

    gotoxy(10,9);cout<<"Dia";

    gotoxy(10,10);cin >>DiaF;

    gotoxy(12,9);cout<<"/";

    gotoxy(13,9);cout<<"Mes";

    gotoxy(13,10);cin >>MesF;

    gotoxy(15,9);cout<<"/";

    gotoxy(17,9);cout<<"Año";

    gotoxy(17,10);cin >>AnoF;



    gotoxy(10,11);cout<<"Numero";

    gotoxy(10,12);cout<<"Cuenta:";

    gotoxy(24,11);cout<<"Monto:";

    gotoxy(32,11);cout<<"Tipo  de";

    gotoxy(32,12);cout<<"cuenta:";

    gotoxy(50,11);cout<<"Numero";

    gotoxy(50,12);cout<<"Transaccion:";

    gotoxy(66,11);cout<<"+/-";//+ o -
```

```
   gotoxy(80,11);cout<<"Fecha";


   for(int  c = 0; c < MAX_Actividad;  c++)
         {
         if(Act_Cuenta[c].Num_Transaccion  > 0)
         {
         if(Act_Cuenta[c].Dia  >= DiaI  && Act_Cuenta[c].Dia <= DiaF && Act_Cuenta[c].mes >=  MesI
&& Act_Cuenta[c].mes <= MesF && Act_Cuenta[c].ano >= AnoI  && Act_Cuenta[c].ano  <= AnoF)
         {
            if(Act_Cuenta[c].Numero_Cliente  == Numero_Cliente)
            {
               if(Act_Cuenta[c].TipoCuenta  == 2)//Ahorro
            {
                  if(user[Numero_Cliente].Num_CuentaA  >0)
                  {
                    gotoxy(10,linea);cout<<Act_Cuenta[c].Num_Cuenta;
                    gotoxy(24,linea);cout<<Act_Cuenta[c].Monto;


                    gotoxy(32,linea);cout<<"Ahorro";


                    gotoxy(50,linea);cout<<Act_Cuenta[c].Num_Transaccion;


                    if(Act_Cuenta[c].Actividad  == 1)
                    {
                       gotoxy(66,linea);cout<<"+";
                    }
                    else
                    {
                       gotoxy(66,linea);cout<<"-";
                    }
                    gotoxy(82,linea);cout<<Act_Cuenta[c].Dia<<"/"  <<Act_Cuenta[c].mes<<"/"
<<Act_Cuenta[c].ano;
                    linea++;
```

```
                    }
                }
            }
        }
    }
    linea++;
    gotoxy(17,linea);system("pause");
}
//Menu del cliente
void MenuCliente(int Numero_Cliente)
{
    int select;

    system("cls");

    gotoxy(25,2); cout << "Diners Bank";
    gotoxy(18,4); cout << "Bienvenid@: "<< user[Numero_Cliente].Nombre;

    int linea = 5;
    linea = mostrar_saldos(Numero_Cliente, linea);

    if(user[Numero_Cliente].Num_CuentaA >0)
    {
    gotoxy(8,linea);cout<<"1. Movimientos de la cuenta ahorro desde una fecha hasta otra fecha.";
    linea ++;
    }
    else
    {
    gotoxy(8,linea);cout<<"-- Opcion 1 No Disponible--Movimientos de la cuenta ahorro desde una
fecha. NO POSEE CUENTA DE AHORRO";
    linea ++;
```

```
    }
    gotoxy(8,linea);cout<<"2. Registrar Retiro de Dinero";linea ++;
          gotoxy(8,linea);cout<<"3. Registrar Deposito de Dinero";linea ++;
          gotoxy(8,linea);cout<<"4. Ver depositos y retiros realizados desde una caja
determinada";linea ++;
          gotoxy(8,linea);cout<<"5. Salir";linea ++;
    gotoxy(8,linea); cout << "Seleccione: [ ]";
          gotoxy(21,linea); cin >> select;//Guarda en la variable select la opcion seleccionada por el
usuario


    switch(select)
    {
       case 1:
       {
          if(user[Numero_Cliente].Num_CuentaA >0)
          MovimientoFHF(Numero_Cliente);
          else
          MenuCliente(Numero_Cliente);
       break;
       }
       case 2:
       {
          Transaccion(2, Numero_Cliente);//retiro
       break;
       }
       case 3:
       {
          Transaccion(1, Numero_Cliente);//depositar
       break;
       }
       case 4:
       {
          MostrarCaja(Numero_Cliente);
```

```
        break;
        }
        case 5:
        {
            menu();
        break;
        }
    }
}

//Sistema de registro
void RegistrarCliente()
{
    char SoN[1];
    system("cls");
    for(int c = 0; c < MAX_Usuarios; c++)
    {
        if(user[c].Num_Cliente == 0)//Slot Vacio detecta el array del estruct sin uso
        {
            gotoxy(25,2); cout << "Diners Bank";
            gotoxy(26,4); cout << "Registro";
            gotoxy(10,6); cout<<"Ingrese su Nombre_Apellido: ";
             cin >> user[c].Nombre;
            gotoxy(10,8); cout<<"Ingrese su cedula: ";
             cin >> user[c].Cedula;
            gotoxy(10,10); cout<<"Ingrese su Telefono: ";
             cin >> user[c].Telefono;
            gotoxy(10,12); cout<<"Ingrese su Direccion: ";
             cin >> user[c].Direccion;
             cin >> user[c].Direccion;
            gotoxy(10,14); cout<<"Desea tener una cuenta corriente s/n: ";
             cin >> SoN;
```

```
 SoN[0] = tolower(SoN[0]);
 int linea = 15;
 if(strcmp(SoN,  "s") == 0)
 {
 Numero_Cuentas++;
 user[c].Num_CuentaC  = Numero_Cuentas;

 gotoxy(10,linea);  cout<<"Ingrese  el saldo que tendra en la Cuenta  Corriente  en Bs: ";
 cin >> user[c].SaldoC;
 linea += 2;
}

gotoxy(10,linea);  cout<<"Desea tener una cuenta Ahorro  s/n: ";
 cin >> SoN;
 SoN[0] = tolower(SoN[0]);
 linea ++;
 if(strcmp(SoN,  "s") == 0)
 {
 Numero_Cuentas++;
 user[c].Num_CuentaA  = Numero_Cuentas;

 gotoxy(10,linea);  cout<<"Ingrese  el saldo que tendra en la Cuenta  Ahorro  en Bs: ";
 cin >> user[c].SaldoA;
 linea += 2;
}

gotoxy(10,linea);  cout<<"Desea tener una Cuenta de Activos Liquidos s/n: ";
 cin >> SoN;
 SoN[0] = tolower(SoN[0]);
 linea ++;
 if(strcmp(SoN,  "s") == 0)
 {
```

```
            Numero_Cuentas++;
            user[c].Num_CuentaAL  = Numero_Cuentas;

            gotoxy(10,linea); cout<<"Ingrese el saldo que tendra en la Cuenta de Activos Liquidos en
Bs: ";
            cin >> user[c].SaldoAL;
            linea += 2;
           }

            Numero_Clientes++;
            user[c].Num_Cliente  = Numero_Clientes;

           gotoxy(10,linea);cout<<"Registro  completado correctamente.";
            linea += 2;

           gotoxy(10,linea);system("pause");

            break;
       }
   }
}
//Sistema de ingreso para entrar como cliente
void LogearCliente()
{
    char cedula[20];
    char SoN[1];

    system("cls");

    gotoxy(25,2); cout << "Diners Bank";
    gotoxy(20,4); cout << "Ingresar:";
    gotoxy(17,6);cout<<"Ingrese  su cedula:";
```

```
            gotoxy(37,6);  cin >> cedula;
            for(int  c = 0; c < MAX_Usuarios;  c++)
            {
            if(strcmp(cedula,  user[c].Cedula)  == 0)
            {
            MenuCliente(c);
            break;
        }
        else if(c == MAX_Usuarios-1)
        {
            gotoxy(17,10);cout<<"ERROR:  Cedula  no encontrada.";
            gotoxy(10,11);  cout<<"Desea salir s/n: ";
            cin >> SoN;

            SoN[0] = tolower(SoN[0]);
            if(strcmp(SoN,  "s") == 0)
            {
            menu();
            }
            gotoxy(17,12);system("pause");
            LogearCliente();
            break;
        }
    }
}

//Muestra las cuentas corriente  existentes
void  MostrarCuentasCorrientes()
{
    int linea= 5;
    system("cls");
    gotoxy(17,3);cout<<"Clientes  con cuenta corriente";
```

```
    gotoxy(17,4);cout<<"Nombre:";
    gotoxy(27,4);cout<<"Cedula:";
    gotoxy(40,4);cout<<"Numero De cuenta:";

    for(int c = 0; c < MAX_Usuarios; c++)
            {
            if(user[c].Num_CuentaC > 0)
            {
            gotoxy(17,linea);cout<<user[c].Nombre;
            gotoxy(27,linea);cout<<user[c].Cedula;
            gotoxy(40,linea);cout<<user[c].Num_CuentaC;
            linea++;
        }
    }
    linea++;
    gotoxy(17,linea);system("pause");
}

//Menu principal

void menu()
{
    int select;

    system("cls");

    gotoxy(25,2); cout << "Diners Bank";
    gotoxy(20,4); cout << "Seleccione una opcion:";
    gotoxy(15,6);cout<<"1. Registrar Cliente";
    gotoxy(15,7);cout<<"2. Entrar como Cliente";
    gotoxy(15,8);cout<<"3. Mostrar lista de clientes con cuenta corriente";
    gotoxy(15,9);cout<<"4. Salír";
```

```
        gotoxy(22,11);  cout << "Seleccione:  [ ]";
        gotoxy(35,11);  cin >> select;

switch(select)
{
    case 1:
    {
        RegistrarCliente();
        menu();
        break;
    }
    case 2:
    {
        LogearCliente();
        menu();
        break;
    }
    case 3:
    {
        MostrarCuentasCorrientes();
        menu();
        break;
    }
    case 4:
    {
        system("cls");
        cout<<"Cerrando...";
        Sleep(2000);
        exit(0);
        break;
    }
    default:
```

```
        {
            cout<<"\n\nLa opcion que usted ha seleccionado no esta disponible"<<endl;
            cout<<"Por favor,  vuelva  a intentarlo"<<endl;
            system("pause");
            menu();
        }
    }
}

int main()
{
    menu();
    return 1;
}
```

Tabla de Variables usadas:

Variable usada:	Tipo:	Función:
Num_CuentaC;	**Int**	**Registra el número de cuenta corriente Automáticamente según se valla registrando.**
Num_Cliente	**Int**	**Registra el número de cliente automáticamente según se valla registrando.**
Num_CuentaA	**Int**	**Registra el número de cuenta de ahorros automáticamente según se valla registrando.**
Num_CuentaAL	**Int**	**Registra el número de activos líquidos automáticamente según se valla registrando.**
SaldoC	**Float**	**Almacena el monto de bs en la cuenta corriente.**

SaldoA	**Float**	**Almacena el monto de bs en los ahorros.**
SaldoAL	**Float**	**Almacena el monto de bs en los activos líquidos.**
Teléfono	**Char**	**Guarda el teléfono del usuario según se va registrando.**
Cedula	**Char**	**Guarda la cedula del usuario según se va registrando.**
Dirección	**Char**	**Guarda la dirección del usuario según se va registrando.**
Num_Transaccion	**Int**	**Guarda el de la transacción realizada por usuario.**
Monto	**float**	**Ingresa el valor para la transacción del usuario.**
Saldo	**float**	**Es la cantidad de dinero activa en la cuenta.**
Dia	**Int**	**Registra el día de la transacción realizada.**
Mes	**int;**	**Registra el mes de la transacción realizada.**
Ano	**Int**	**Registra el año de la transacción realizada.**
dial	**Int**	**Es el inicio del día para la búsqueda de las transacciones según se desee buscar.**
mesI	**Int**	**Es el inicio del mes para la búsqueda de las transacciones según se desee buscar.**
anoI	**Int**	**Es el inicio del año de búsqueda de las transacciones según se desee buscar.**
DiaF	**Int**	**Es el dia final de búsqueda de las transacciones según se desee buscar.**

mesF	**Int**	**Es el mes final de búsqueda de las transacciones según se desee buscar.**
anoF	**Int**	**Es el año final de búsqueda de las transacciones según se desee buscar.**
Caja usada	**Int**	**Registr la caja donde se hace la transacción.**
Actividad	**Int**	**Se encarga de alamcenar el tio de transacción dada por el usuario.**
Tipo de cuenta	**Int**	**Es la opción de cualquiera de las 3 cuentas a usar que tenga activa el usuario.**
Número de cuenta	**Int**	**Es el orden de las cuentaws según el banco las va habilitando.**
Nombre	**Char**	**Guarda el nombre del usuario de la cuenta según lo registrado.**
línea	**Int**	**Determina donde el programa muestra el mensaje.**
Numero_clientes	**Int**	**Lleva el orden de la cantidad de clientes que hay.**
Numero_cuentas	**Int**	**Lleva el orden de la cantidad de cuentas que hay.**
Numero_transacciones	**Int**	**Lleva el orden de la cantidad de transacciones que hay.**
SoN	**Char**	**Almacena la respuesta del usuario si es S/N**

Explicación de la estructura de datos:

int Numero_Clientes = 0;: Controla el número de clientes del banco

int Numero_Cuentas = 0; Controla el número de Cuentas del banco

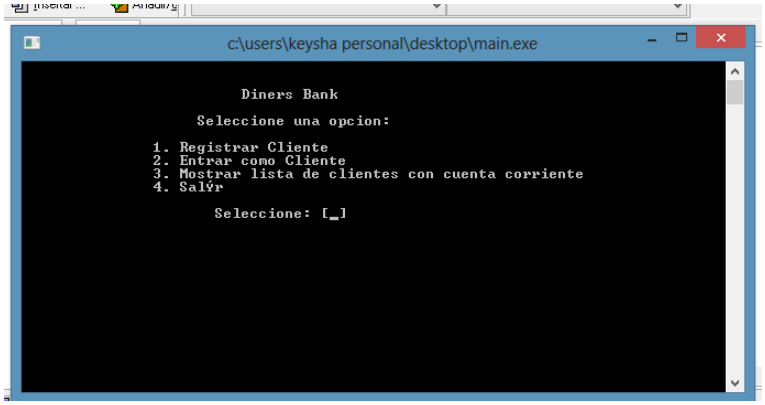
int Numero_Transacciones = 0; Controla el número de Transacciones del banco

En la estructura del usuario se utiliza char Nombre[24]; para almacenar el nombre, int Num_Cliente se utiliza para almacenar el número del cliente el cual es dado por Numero_Clientes y se le suma 1 a Numero_Clientes, luego se utiliza Num_CuentaC para guardar el numero de la cuenta Corriente el cual es dado por Numero_Cuentas y se le suma 1, se utiliza Num_CuentaA para guardar el el numero de la cuenta Ahorro el cual es dado por Numero_Cuentas y se le suma 1, se utiliza Num_CuentaAL para guardar el el numero de la cuenta de activos líquidos el cual es dado por Numero_Cuentas y se le suma 1, float SaldoC utilizado para guardar el saldo de la cuenta corriente luego de una transaccion o luego de crear la cuenta, float SaldoA utilizado para guardar el saldo de la cuenta ahorro luego de una transaccion o luego de crear la cuenta, float SaldoAL utilizado para guardar el saldo de la cuenta de activos liquidos luego de una transaccion o luego de crear la cuenta, Char Telefono; almacena el número de teléfono del cliente, char Cedula[20]; Almacena la cedula del Cliente el cual es dada al momento de registrarse, char Direccion[50] almacena la dirección_del_cliente.

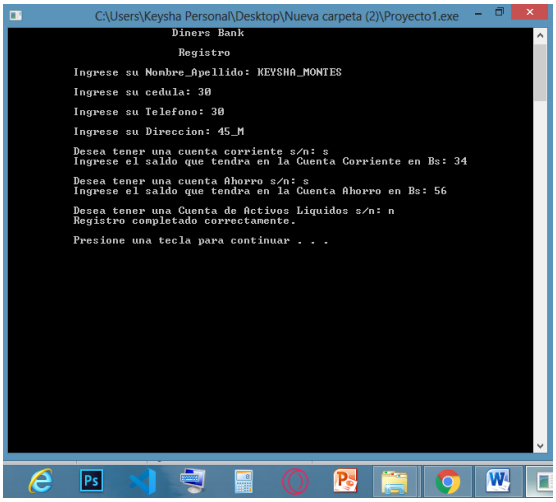
En la estructura del Actividad_De_cuenta se utiliza int Numero_Cliente; para almacenar el numero del cliente que hizo la transaccion en la funcion de transaccion, int Num_cuenta; se utiliza para guardar el numero de cuenta en la cual se hizo transaccion, int TipoCuenta; almacena el tipo de cuenta la cual hizo la transacción, int Num_Transaccion; Almacena el numero de la transaccion el cual es Numero_Transacciones y sele suma 1 a Numero_Transacciones, int Caja_usada; esta es dada por el usuario al momento de hacer la transaccion, int Actividad; si el valor es 1 seria depostio 2 seria retiro el numero lo determina el usuario al momento de hacer la transaccion, float Monto; El monto usado en la transaccion, float saldo; el saldo de la cuenta al momento de haber echo esta transaccion, int dia almacena el dia de la transaccion, int mes; almacena el mes de la transaccion, int ano almacena el año de la transaccion todos estos datos son suministrados por la función Transaccion(int actividad, int Numero_Clie) ;excepto int actividad es si es 1= deposito, 2 = Retiro y el parámetro Numero_cliente es el numero del cliente el cual desea realizar la transaccion, estos últimos datos son determinados en el menú del usuario y dependiendo del usuario.

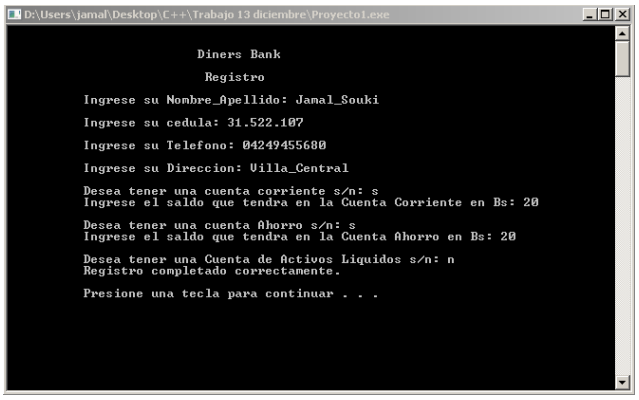
Capturas de pantalla:

Menu inicial

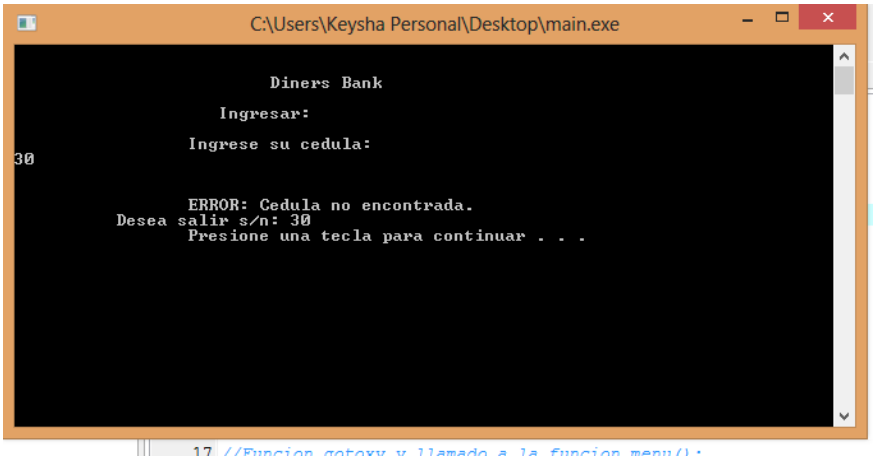


Registro de datos

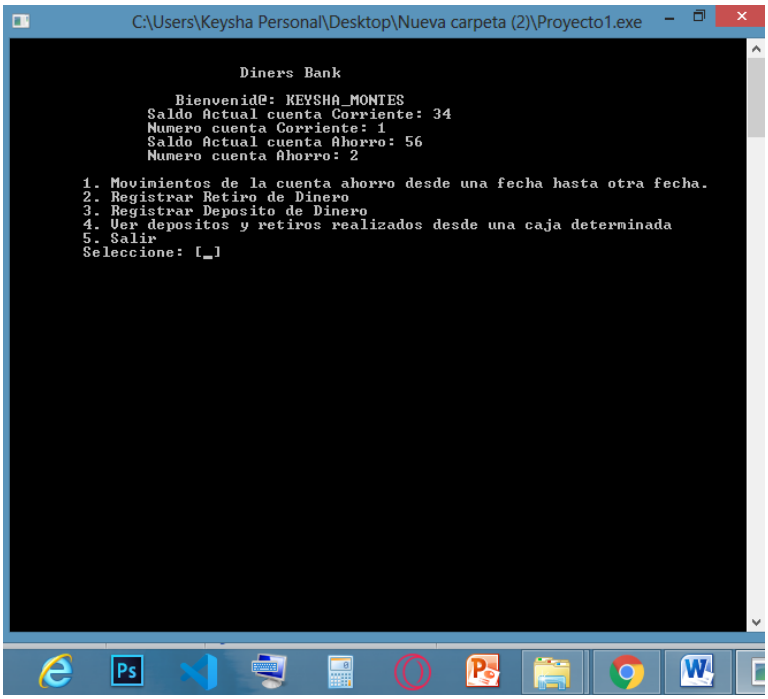




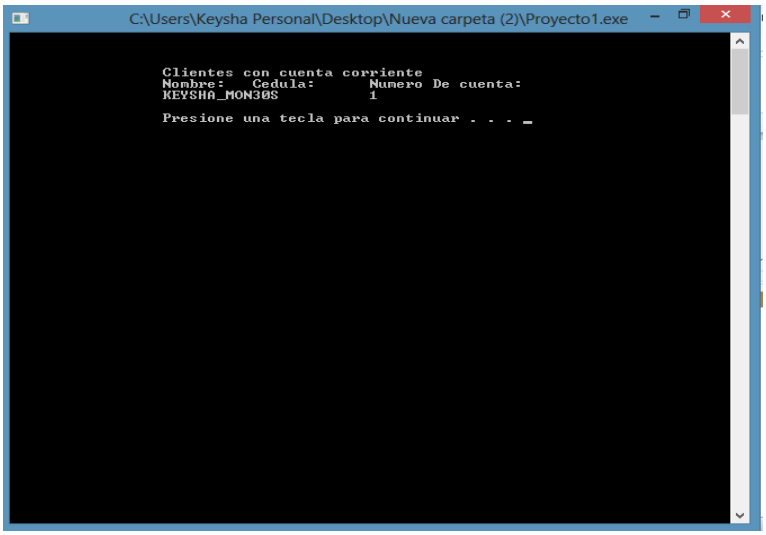
Ingreso erróneo

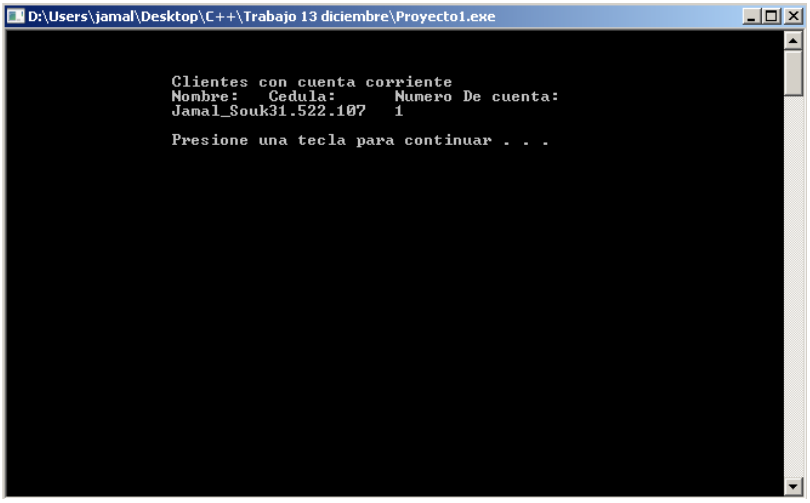


Ingreso como cliente:



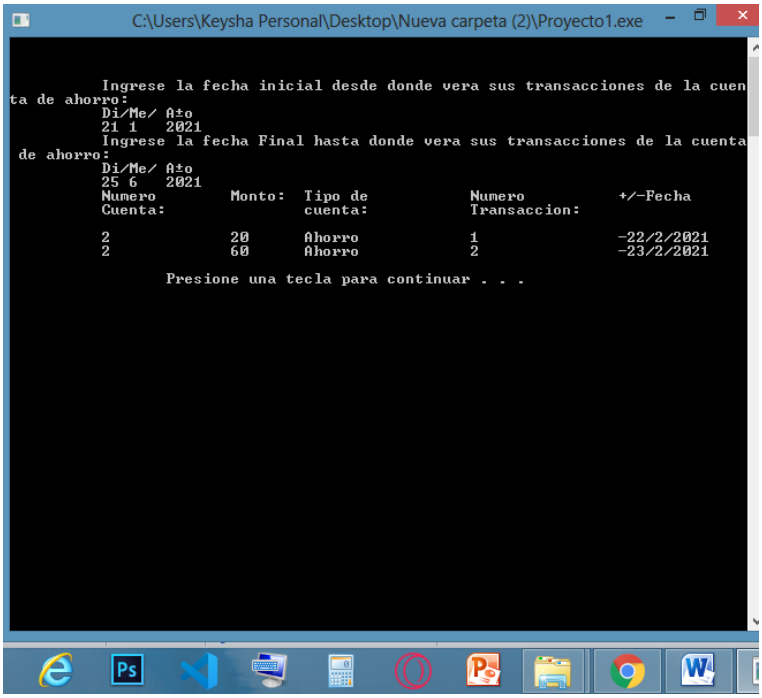
Listado cuentas corriente



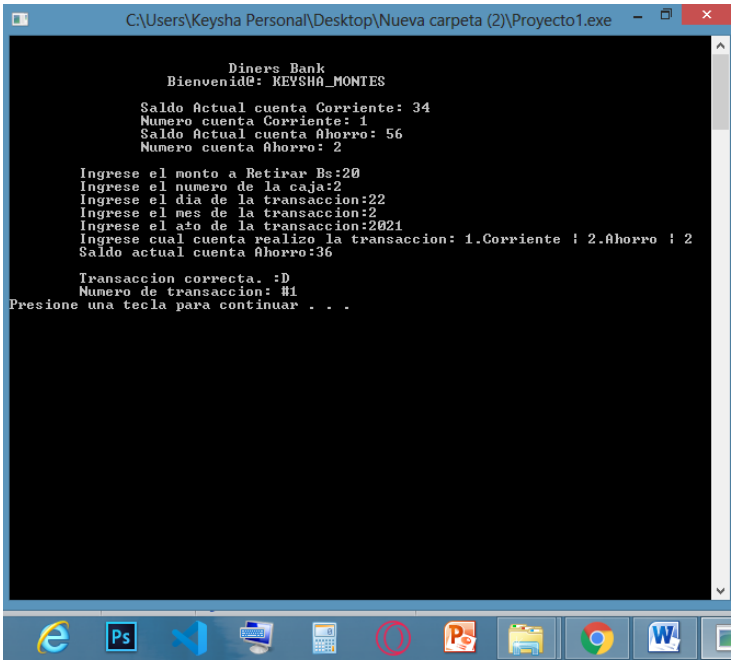


Nota: Esta imagen fue tomada luego de que el programa se cerrara por ende es la cuenta 1 la de Jamal_Souki

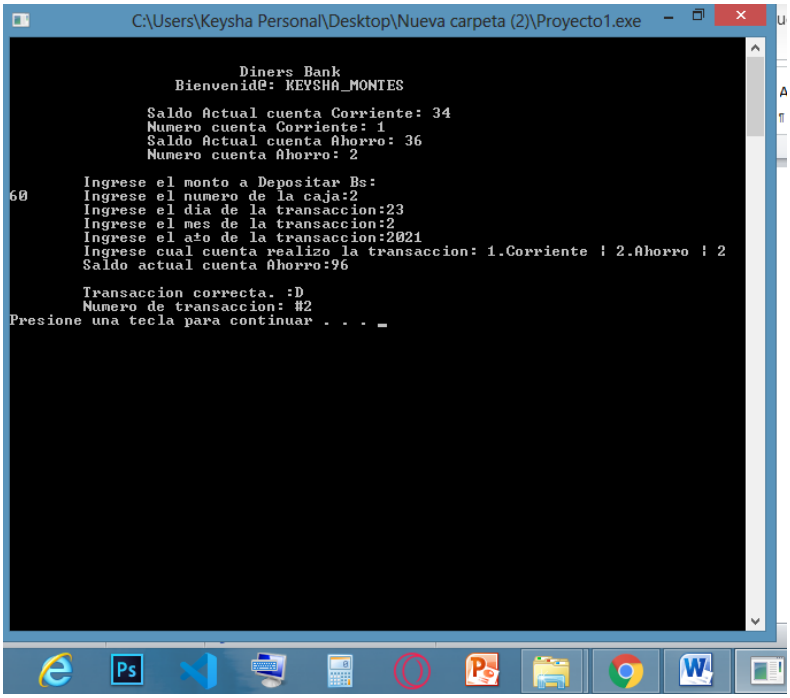
Movimientos cuenta ahorro de una fecha a otra



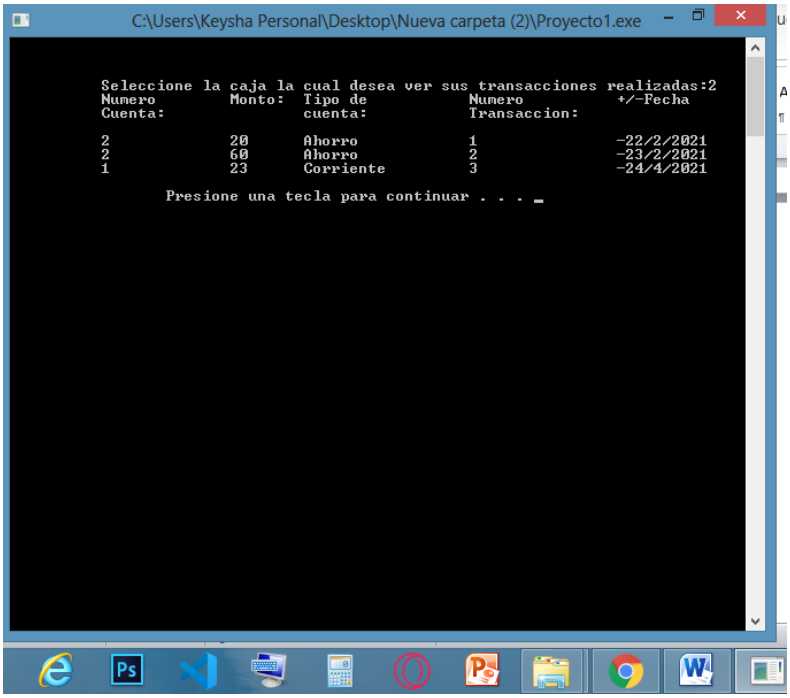
Registro de retiro



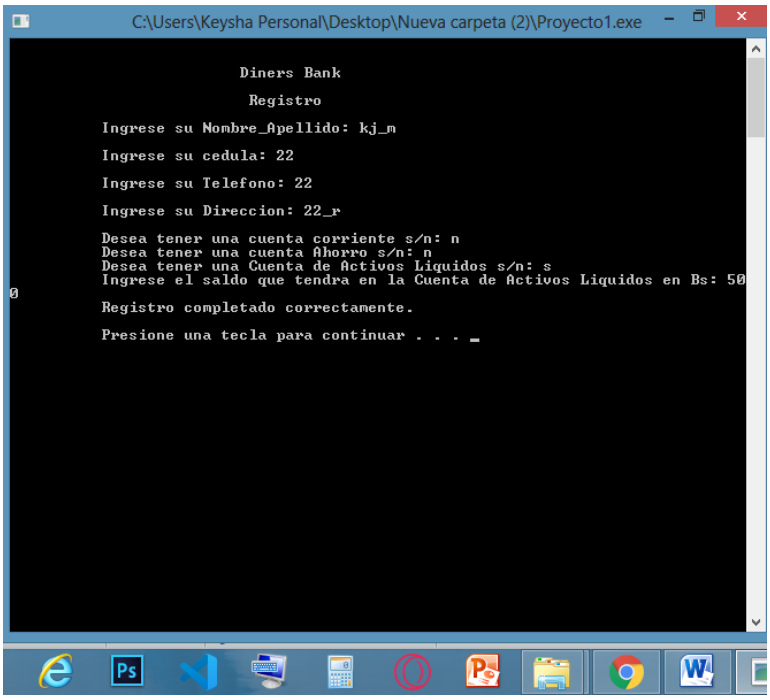
Registrar deposito

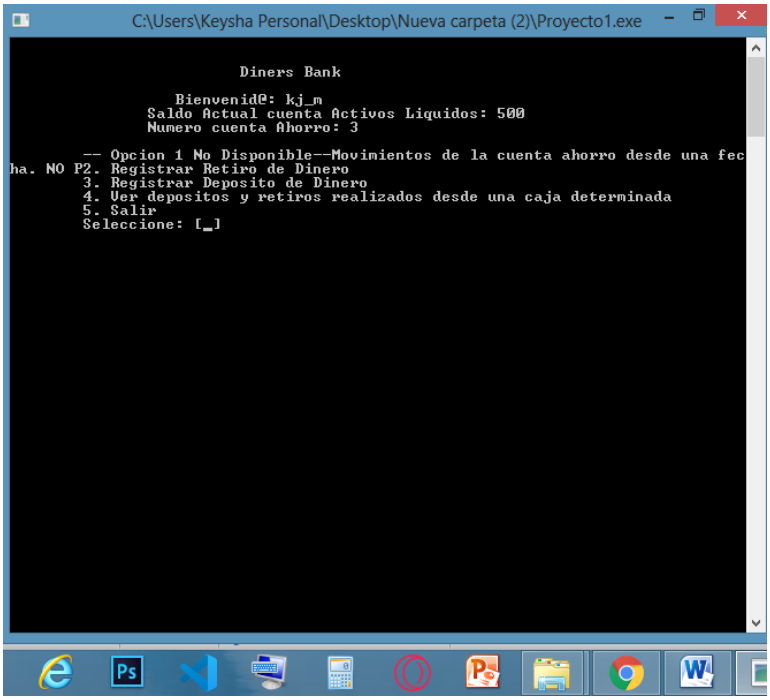


Ver depósitos y retiros de una caja determinada



Error de registro de fechas al no poseer cuenta ahorro





Transaccion de activos liquidos