Fundamentos de Programación, Proyecto Final Otoño 2015

Sistema de Banco Virtual

Resumen

El presente documento es la especificación para el desarrollo de un sistema de servicios bancarios en línea para cajeros automáticos. El sistema permite administrar una red de sucursales de cajeros automáticos y los servicios que se pueden ofrecer a los clientes.

Las funciones principales que se pretende realice y administre el sistema son: los servicios de cajeros automáticos, manejo de transacciones bancarias, la alta de usuarios y diversos reportes, por mencionar los más importantes.

Descripción del Sistema

Este sistema de Banco Virtual permite la programación de servicios y administración de una red de cajeros en distintas zonas de una ciudad, la administración de los usuarios y de las transacciones que ellos efectúan. El sistema deberá dar acceso a dos tipos de usuarios:

- El ***administrador del sistema***, que es la persona que dará de alta los servicios bancarios en línea y a los usuarios que deseen hacer uso de los cajeros.
- El ***usuario de cajeros***, que serán las personas que harán uso de los servicios de sistema en cajeros automáticos.

El sistema permitirá al ***administrador*** controlar y asignar recursos a los distintos cajeros automáticos que se encuentran distribuidos en la ciudad (Sucursales Tlalpan, Santa Fe y Coyoacán).

Para poder hacer uso de los servicios de los cajeros por un ***usuario***, el ***administrador*** deberá dar de alta los usuarios.

Este sistema deberá ser desarrollado por equipos de tres personas; no hay cambios de equipo durante el desarrollo del proyecto. El sistema deberá desarrollarse y presentarse compilado en la computadora Antares.

Alcances y Limitaciones

La funcionalidad y estética de la interfaz serán evaluadas. El diseño de la interfaz de usuario y administrador queda a consideración de cada equipo. A su vez, el programa deberá tener el máximo de validación posible en función de lo que se ha visto en el curso.

El sistema debe permitir las siguientes facilidades al ***administrador del sistema***:

1.- Acceso como administrador vía password.
2.- Alta y baja de ***usuarios de los servicios*** bancarios.
3.- Transferencia de dinero a los distintos cajeros.
4.- Mostrar el estatus de todos los cajeros.
5.- Mostrar un reporte de cada **usuario** en pantalla.
6.- Ayuda al usuario ***administrador***.
7.- Salida del sistema.

Acceso como administrador vía password: al iniciar el programa en modo ***administrador,*** *el sistema* deberá pedir el nombre de usuario y contraseña para que le permita, entonces, ejecutar todas las funciones del sistema.

Alta y baja de usuarios de los servicios bancarios: en este módulo el sistema debe permitir crear un nuevo usuario del servicio con los siguientes datos:

- Nombre.
- Dirección.
- RFC.
- Password.
- Cuenta de correo electrónico.
- Número de cuenta (asignado por el banco).

- Número de tarjeta de débito (que es la cuenta de cheques).
- Número de tarjeta de crédito.
- Número de usuario. Deberá ser un número secuencial creado por el sistema.
- Saldo de la cuenta de cheques (de donde se dispone para la tarjeta de débito).
- Saldo de la tarjeta de crédito.

Transferencia de dinero a los distintos cajeros: en este módulo el sistema debe permitir que el banco pueda transferir un monto de dinero a un determinado cajero. De esta forma se puede ir acumulando recursos en cada cajero para que tengan dinero disponible.

Mostrar el estatus de todos los cajeros: en este módulo el sistema debe de generar un reporte en pantalla con el estatus de todas los cajeros que debe de mostrar: nombre de la sucursal del cajero y dinero disponible.

Mostrar un reporte de cada usuario en pantalla: en este módulo el sistema debe de generar un reporte en pantalla con los datos completos de usuario un usuario determinado por su número de cuenta.

Ayuda al usuario: en este módulo se le mostrará al administrador una guía de uso y funcionalidades del sistema.

Salida del sistema: en este módulo el usuario indicará que saldrá del sistema. Antes de salir el sistema actualizará los archivos del catálogo con la información actualizada y terminará la ejecución del programa.

El sistema debe permitir las siguientes facilidades al ***usuario del servicio***:

1.- Acceso como ***usuario del servicio*** a un cajero determinado de una sucursal.
2.- Disposición en efectivo de su cuenta de cheques con tarjeta de débito.
3.- Disposición en efectivo de su tarjeta de crédito.
4.- Mostrar su saldo de cuenta de cheques.
5.- Mostrar su saldo de tarjeta de crédito.
6.- Realizar el pago de su cuenta de cheques a la tarjeta de crédito

Acceso como usuario del servicio: un cliente podrá iniciar el programa en modo ***usuario*** especificando la sucursal. A partir de aquí se debe mostrar en pantalla las siguientes opciones. Este acceso simula el ingreso con tarjeta y NIP.

Disposición en efectivo de su cuenta de cheques con tarjeta de débito: el usuario podrá retirar un monto de su cuenta no mayor a su saldo y con un límite por día de $7,000.00 en los distintos cajeros. Si trata de exceder debe ser notificado por el sistema y evitar la transacción. Después de esta disposición, el sistema debe actualizar su saldo. Deberá enviar un comprobante de transacción vía correo electrónico.

Disposición en efectivo de su tarjeta de crédito: el usuario podrá retirar un monto de su cuenta no mayor a su saldo y con un límite por día de $10,000.00 en los distintos cajeros. Si trata de exceder debe ser notificado por el sistema y evitar la transacción. Se hará un cobro de comisión del 2% de lo dispuesto con afectación al saldo de la tarjeta de crédito. Deberá enviar un comprobante de transacción vía correo electrónico.

Mostrar su saldo de cuenta de cheques: en este módulo podrá el usuario verificar el saldo que tenga en su cuenta de cheques.

Mostrar su saldo en tarjeta de crédito: en este módulo podrá el usuario verificar el saldo que tenga en su cuenta de crédito.

Requisitos sistema

La información de los catálogos de ***usuarios*** y ***cajeros*** deberán estar almacenados en archivos y estos deberán ser transferidos, al iniciar el programa, a estructuras estáticas (arreglos) o dinámicas (estructuras auto-referenciadas con punteros) para el manejo de la información por el usuario y administrador durante su sesión en el sistema. Al finalizar la sesión del usuario o el administrador, el sistema deberá actualizar la información en archivos antes de salir de ejecución.

El programa ejecutable deberá llamarse *bancodigital.*

Primera forma de ejecución:

$ ./bancodigital

En este caso no se pasa ningún argumento de programa a través de la línea de comandos. Al iniciar, el programa deberá mostrar el nombre de los desarrolladores del sistema. En este modo se entra como administrador del sistema y se solicita el acceso con password. Cuando el usuario presiona una tecla, debe aparecer un menú que integre los módulos descritos en la sección anterior para el administrador. El programa deberá permanecer en el ciclo del menú y cada vez que se termine de utilizar un módulo se deberá regresar al menú específico.

Segunda forma de ejecución.

$ ./bancodigital –*sucursal* (-tlalpan ó –santafe ó -coyoacan)
$ ./bancodigital -c
$ ./bancodigital -usu

En este caso lo único que hace el programa es listar en pantalla la información general contenida en los diferentes archivos. Cada uno de los modificadores indicados corresponden a:

Modificador	**Comportamiento**
-*sucursal*	Se entra como modo usuario a una distinta sucursal
-c	Despliega los créditos de los desarrolladores del sistema.
-usu	Despliega el listado de todos los ***usuarios del servicio*** catalogados ordenados por nombre.

Bitácora de usuario: el sistema deberá llevar por cada sesión a "modo administrador", un registro de las acciones realizadas por usuario administrador durante la sesión. Esta bitácora deberá tener la fecha y hora de ingreso (*"time stamp"*) y las acciones que realizó con el catálogo: alta y baja de usuario y transferencia de dinero a cajeros.

Especificación y restricciones de la programación del proyecto

El proyecto final tiene como objetivo que los alumnos del curso apliquen de manera práctica todos los conocimientos que fueron transmitidos a lo largo del semestre, así como las "mejores prácticas" de la programación. Por lo anterior, se deberán considerar los siguientes elementos y que serán evaluados:

- Uso del lenguaje de programación con comentarios.
- Uso de funciones y compilación modular.
- Uso de estructuras dinámicas y estáticas.
- Uso de archivos.

Cada equipo usará las estructuras que considere más convenientes para el diseño y desarrollo del proyecto.

Algunas buenas prácticas de programación que se deben tomar en cuenta:

- Los equipos deberán trabajar en apego a las prácticas éticas de programación.
- Identificadores significativos (es decir, nombres de variables y de funciones deben referirse a la utilidad de los mismos).
- Alineación de código para garantizar la legibilidad del código fuente (debe ser la alineación que se genera de manera automática por medio de emacs).
- Documentación del código fuente, es decir, cada archivo de código fuente debe contener todos los comentarios que sean necesarios para garantizar que cualquier otro programador competente pueda comprender el código. En sí, se recomienda que los comentarios se usen para explicar aspectos no obvios del código, o que no se comprendan a primera vista.

Documentación que debe presentarse como entrega del proyecto

Se deberá entregar un trabajo escrito con la definición del problema, el diseño de pantallas, el diseño de la solución (diagramas de bloques y/o pseudocódigo, en tantos niveles como sea necesario), el plan de pruebas del sistema y un manual de usuario. Además se deberá anexar una tabla que liste y explique que parte del proyecto hizo cada integrante.

Nota: Se evaluará la redacción, estilo y ortografía del trabajo (la única excepción es la ortografía en el código fuente).

Fechas de Entrega

Análisis y Diseño: los equipos deben entregar el análisis y diseño del proyecto el día miércoles 25 de noviembre a las 7:00 horas.
Proyecto final: el miércoles 02 de diciembre a las 7:00 horas entregarán el proyecto codificado y compilado en una cuenta de Antares. Posteriormente a cada equipo se le asignará un turno de revisión. **Los equipos que no presenten el análisis y diseño no tendrán derecho a la revisión del proyecto.**

ANEXO: DOCUMENTACIÓN DEL PROYECTO

Documentar código es un elemento importante para dar mayor claridad a los algoritmos y a la forma como fue codificado un problema.

La documentación se hará por medio de un formato especial de los comentarios. En este caso se seguirá la siguiente convención:

Comentarios a nivel del archivo:
Estos comentarios proveen una descripción general del problema y elementos generales como son el autor y fecha de creación y modificación. Se deben colocar en la parte superior, antes de cualquier línea de código (arriba de las directivas #include)

```
/*
*  @file arreglos.c
*
*  @brief Este programa permite capturar
dos arreglos de tipo entero
*  para despues sumarlos en un tercer
arreglo, y obtener el promedio
*  de la suma de valores del tercer
arreglo, imprimiendo la salida en
*  tipo flotante.
*
*  @author Pablo Segovia
*  @date 12/03/2010
*/
```

@file debe declarar el nombre del archivo
@brief una descripción de lo que hace el programa. La descripción debe estar en un párrafo y terminar con "."
@author debe especificar el nombre del o los autores del código
@date es la última referencia con la fecha en que se creó el archivo

Comentarios a nivel de función:
Arriba de la implementación de cada función (no donde se declara el prototipo ni donde se invoca), se debe documentar cada función, de acuerdo al autor, parámetros, fecha y tipo de retorno. Por ejemplo:

```
/*
*  Esta función recibe un caracter en el
primer argumento y si
*  este es una letra minuscula lo
convierte a mayuscula y regresa
*  el nuevo valor en el segundo
argumento.
*  Regresa un 1 si pudo convertir a
mayuscula el caracter
*  y 0 si no pudo hacerlo.
*  @author Iggy Pop
*  @param chrData     El caracter a
convertir (minuscula a mayuscula)
*  @param *may        El caracter
convertido en mayuscula
*  @return int
*
*/
intcarMinAMay(char chrData, char *may)
{
........
}
```

@param Nombre_Variable Descripción Muestra los argumentos de la función
@return identifica el tipo del valor que regresa la función

Nota: Si hubiera algún algoritmo importante dentro de una función, se podrán incluir comentarios antes de la implementación de dicho comentario