



Escuela de  
Ingeniería y Arquitectura  
**Universidad** Zaragoza

---

# PRÁCTICA 1 - COBOL

GRUPO 3 - Sistemas Legados

---

David Zandundo Fuster (780500)  
Álvaro Andrada Gimeno (795326)  
Rubén Albacete Seren (736650)

<b>Paso 1 - Compilación y corrección de errores iniciales</b>	<b>3</b>
<b>Paso 2 - Modificación de las opciones del programa</b>	<b>6</b>
<b>Paso 3 - Mejoras del programa final</b>	<b>18</b>

## Paso 1 - Compilación y corrección de errores iniciales

Como primera tarea, hay que buscar un compilador capaz de crear un ejecutable para Windows. Para ello se buscó en diferentes sitios web, pero se encontró un IDE para desarrollar aplicaciones escritas solamente en COBOL llamado [OpenCobolIDE](#), que utiliza un compilador llamado [GnuCOBOL](#) que, por lo que pudimos ver en su documentación, se utiliza en multitud de sistemas operativos.

Se vio que este IDE tenía una interfaz bastante fácil de comprender y usar, además podía compilar todos los ficheros de nuestro programa y crear un solo ejecutable en Windows.

Una vez que se pudo ejecutar el programa, se vio que el texto que se presenta por pantalla no estaba encuadrado y algunos ni siquiera se veían. La etiqueta DISPLAY es la encargada de sacar por pantalla texto y tenía dos números entre paréntesis que eran la posición de la fila y columna. Se modificó eso añadiendo las etiquetas FILE y COL (por ejemplo DISPLAY "Enter - Aceptar" LINE 24 COL 3.), se cambió en todos los ficheros y ahora sí que se podían ver los textos de la pantalla.

Se vio también que cuando intentaba acceder al fichero de "movimientos.ubd" crasheaba el programa, eso era porque no existía ese fichero. Se creó uno nuevo con contenido vacío, ya que todavía no se había hecho ningún movimiento.

Se apreció que cada vez que se quería abrir un fichero cualquiera salía la pantalla de error que se había diseñado para el programa. Eso se debía a que cuando se comprobaba la variable 'FSM', que indicaba el estatus del fichero, este no era el correcto:

OPEN INPUT F-MOVIMIENTOS. IF FSM <> 30 GO TO PSYS-ERR	Este era el código que había inicialmente. Ese valor de '30' corresponde con un <a href="#">código de error permanente</a> , es decir que cuando no había un error con el fichero entraba dentro del IF.
OPEN INPUT F-MOVIMIENTOS. IF FSM <> 00 GO TO PSYS-ERR	Con el código de la izquierda, si en la variable se almacena un valor distinto a 00, entonces había ocurrido algún error y entraba en el IF. Si el fichero se abre correctamente, en la variable FSM se almacenaba un 00, y no se entraba dentro del IF.

Se vio que la operación de "Cambiar pin" no estaba implementada, para ello se creó un fichero llamado "BANK9.cbl", a partir del fichero de "BANK2.cbl" que consultaba el saldo. Lo que hace esta funcionalidad es presentar por pantalla el pin correspondiente con el número de la tarjeta, a su vez se pide al usuario escribir el nuevo pin para ese número de cuenta, una vez escrito, el usuario pulsa la tecla ENTER y se le presenta una pantalla con el nuevo pin que ha introducido, nuevamente el usuario pulsa la tecla ENTER y vuelve al menú principal.

A continuación, se detallan los extractos de código más importantes para esta nueva funcionalidad:

<pre>READ TARJETAS INTO WS-Tarjeta AT END CLOSE TARJETAS NOT AT END   PERFORM     IF TNUM-ID = TNUM       MOVE TPIN-ID TO TPIN-USER     END-PERFORM   END-READ CLOSE TARJETAS.</pre>	<p>Este código se encuentra al principio, lo que hace es leer del fichero "tarjetas.udb" la información de la tarjeta correspondiente con el número de cuenta del usuario (TNUM) y almacena el pin de la tarjeta (TPIN-USER) para luego presentarlo por la pantalla.</p>
<pre>OPEN I-O TARJETAS. IF FST NOT = 00   GO TO PSYS-ERR. MOVE TNUM TO TNUM-E DELETE TARJETAS  MOVE TNUM TO TNUM-E MOVE NEW-PIN TO TPIN-E WRITE TAJETAREG INVALID KEY GO   PSYS-ERR. CLOSE TARJETAS.</pre>	<p>Este código se encuentra al final, lo que hace es que, una vez que ya tenemos el nuevo pin que quiere el usuario, borra en el fichero de "tarjetas.udb" el registro donde se encuentra la información de la tarjeta asociada (TNUM), y por último, escribe el número de cuenta (TNUM) y su nuevo pin (NEW-PIN) en el fichero, cambiando así el pin de la cuenta.</p>

Así es como queda la nueva funcionalidad en nuestro sistema:

The screenshot shows the UnizarBank ATM screen with the following text:

Cajero Automatico UnizarBank

03-10-2022 19:15

Cambia el pin de tu tarjeta

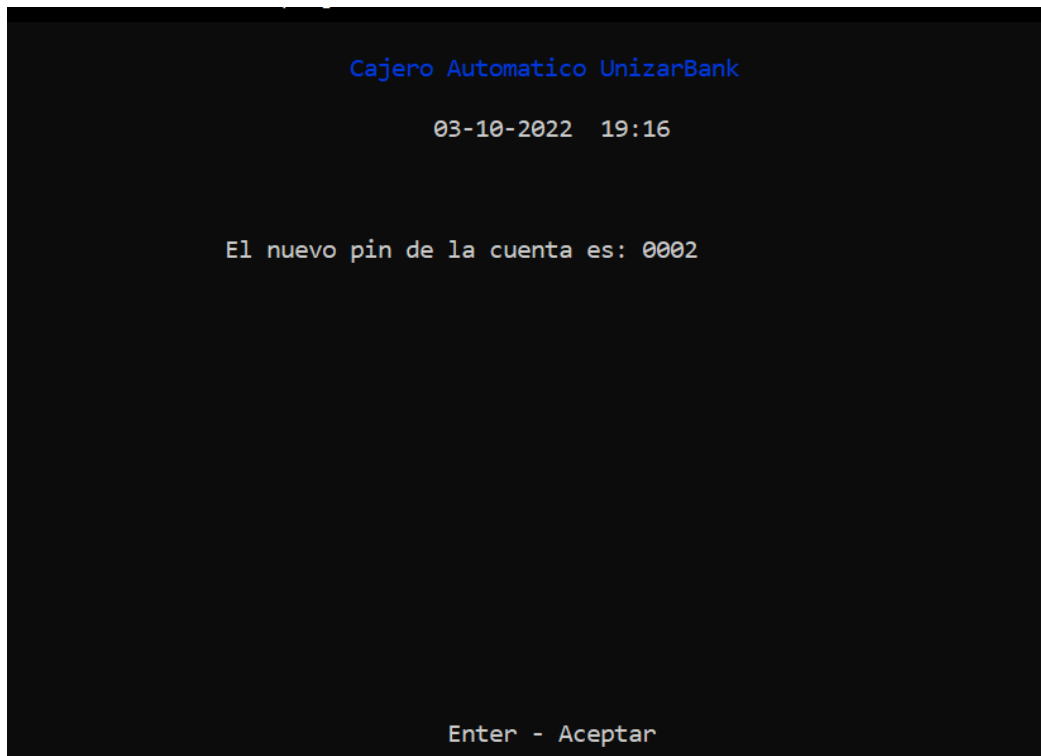
El pin de tu cuenta 0000000000000001 es 0001

Introduzca un nuevo pin: 0002

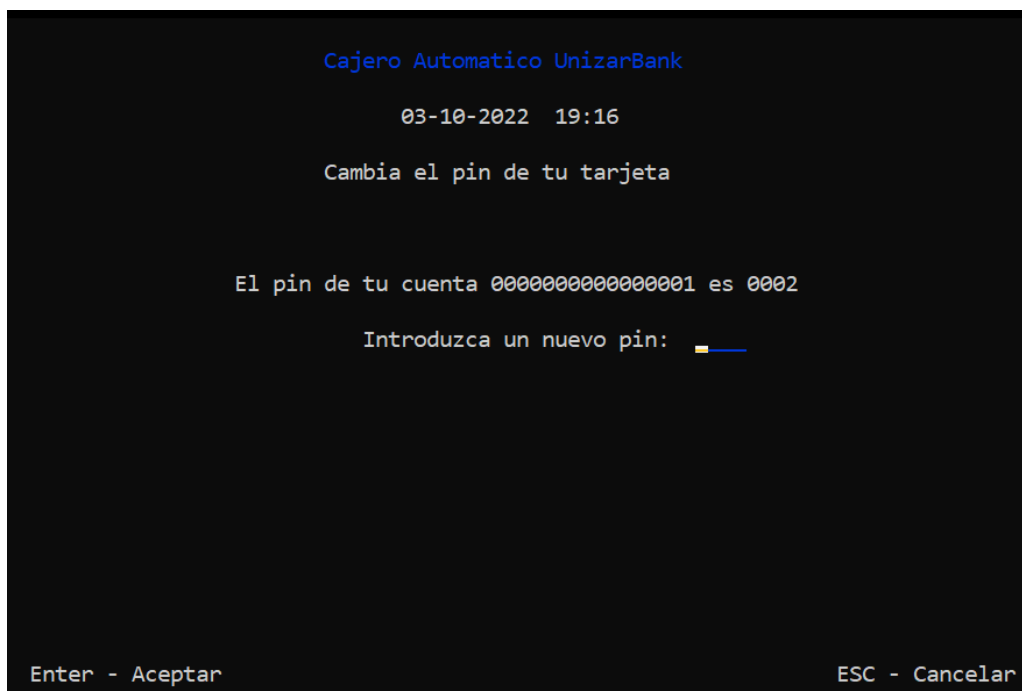
Enter - Aceptar

ESC - Cancelar

Primero el sistema te muestra el pin actual de la cuenta, para que luego el usuario introduzca el nuevo pin. Una vez que se introduce el pin y se pulsa la tecla ENTER aparece la siguiente pantalla:



Mostrando el nuevo pin de la cuenta, si se pulsa a la tecla ENTER se vuelve al menú principal. Si volvemos a seleccionar la opción de 'Cambiar pin', como se puede observar, se muestra el nuevo pin de la cuenta, pudiendo cambiar de nuevo.



## Paso 2 - Modificación de las opciones del programa

Para cambiar de posición las opciones del menú solo hace falta cambiar de sitio las llamadas a las funciones, dentro del IF que se encuentra en el fichero "BANK1.cbl".

Lo siguiente se ha intentado pero no se ha conseguido que funcione (se ha usado una alternativa peor):

Para la mejora de la opción "Ordenar transferencia" (la cual consistía en añadir la posibilidad de que una transferencia se ejecute puntualmente en una fecha dada, o de forma mensual en el día indicado), se decidió añadir en la pantalla del formulario de ordenar transferencia (en BANK6.cbl) espacios para escribir una fecha (para escribir la fecha en la que quieres que se ejecute una transferencia puntual) y otro espacio en el que escribir el día del mes en el que quieres programar una transferencia mensual.

```
01 FILTRO-CUENTA.  
    05 FILLER BLANK WHEN ZERO AUTO UNDERLINE  
        LINE 12 COL 54 PIC 9(16) USING CUENTA-DESTINO.  
    05 FILLER AUTO UNDERLINE  
        LINE 14 COL 54 PIC X(15) USING NOMBRE-DESTINO.  
    05 FILLER BLANK ZERO AUTO UNDERLINE  
        *> SIGN IS LEADING SEPARATE  
        LINE 16 COL 54 PIC 9(7) USING EURENT-USUARIO.  
    05 FILLER AUTO UNDERLINE  
        LINE 16 COL 63 PIC 9(2) USING EURDEC-USUARIO.  
    *> NUEVO  
    05 FILLER AUTO UNDERLINE  
        LINE 19 COL 58 PIC 9(2) USING DIA-USUARIO.  
    05 FILLER AUTO UNDERLINE  
        LINE 19 COL 61 PIC 9(2) USING MES-USUARIO.  
    05 FILLER AUTO UNDERLINE  
        LINE 19 COL 64 PIC 9(4) USING ANO-USUARIO.  
    05 FILLER UNDERLINE  
        LINE 20 COL 70 PIC 9(2) USING DIA-MENSUAL-USUARIO.
```

Lo que se ha decidido hacer con eso es comprobar al darle a ENTER en esa pantalla si la fecha o el día mensual han sido rellenados. Sólo se puede rellenar uno de los dos a la vez para una sola transferencia. Se ha hecho con CHECK-TRANSFERENCIA (y se ha modificado LECTURA-SALDO-DST para que lleve a CHECK-TRANSFERENCIA al terminar):

```
LECTURA-SALDO-DST.  
    READ F-MOVIMIENTOS NEXT RECORD AT END  
    GO TO CHECK-TRANSFERENCIA.  
    IF MOV-TARJETA = CUENTA-DESTINO THEN  
        IF LAST-USER-DST-MOV-NUM < MOV-NUM THEN
```

```

        MOVE MOV-NUM TO LAST-USER-DST-MOV-NUM
    END-IF
END-IF.

GO TO LECTURA-SALDO-DST.

CHECK-TRANSFERENCIA.
    IF ((DIA-USUARIO <> 0 AND MES-USUARIO <> 0 AND
        DIA-USUARIO <> 0) OR (DIA-MENSUAL-USUARIO <> 00)) THEN
        OPEN I-O TRANSFERENCIAS
        GO TO LECTURA-TRANSFERENCIAS
    ELSE
        GO TO GUARDAR-TRF
    END-IF.

```

Si se ha comprobado eso entonces se ejecuta LECTURA-TRANSFERENCIAS y REGISTRAR-TRANSF-PENDIENTE, que en lugar de guardar directamente la transferencia en la lista de movimientos (como en el caso de que se realice una transferencia inmediata, la cual aún se puede realizar no rellenando ninguno de los campos nuevos), se guardan los datos de la transferencia en un archivo diferente que sólo registra las transferencias pendientes (tanto puntuales como mensuales).

```

LECTURA-TRANSFERENCIAS.
    READ TRANSFERENCIAS NEXT RECORD
    AT END GO TO REGISTRAR-TRANSF-PENDIENTE.
    IF LAST-MOV-NUM < TRANSF-NUM THEN
        MOVE TRANSF-NUM TO LAST-MOV-NUM
    END-IF.
    GO TO LECTURA-TRANSFERENCIAS.

REGISTRAR-TRANSF-PENDIENTE.
    CLOSE TRANSFERENCIAS.
    OPEN I-O TRANSFERENCIAS.
    IF FSTM <> 00 THEN
        PERFORM IMPRIMIR-CABECERA THRU IMPRIMIR-CABECERA
        DISPLAY FSTM LINE 9 COL 25
        WITH FOREGROUND-COLOR IS BLACK
        BACKGROUND-COLOR IS RED
        DISPLAY "Error abrir transferencias" LINE 11 COL 32
        WITH FOREGROUND-COLOR IS BLACK
        BACKGROUND-COLOR IS RED
        DISPLAY "Enter - Aceptar" LINE 24 COL 33
        GO TO EXIT-ENTER
    END-IF.

    CLOSE F-MOVIMIENTOS.
    MOVE LAST-USER-DST-MOV-NUM TO MOV-NUM.
    PERFORM MOVIMIENTOS-OPEN THRU MOVIMIENTOS-OPEN.

```

```

READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR.

MOVE FUNCTION CURRENT-DATE TO CAMPOS-FECHA.
ADD 1 TO LAST-MOV-NUM.
MOVE LAST-MOV-NUM TO TRANSF-NUM.
MOVE TNUM TO TARJETA-ORIGEN.
MOVE CUENTA-DESTINO TO TARJETA-DESTINO.
MOVE EURENT-USUARIO TO TRANSF-IMPORTE-ENT.
MOVE EURDEC-USUARIO TO TRANSF-IMPORTE-DEC.
MOVE 0 TO ULTIMA-MENSUALIDAD.
MOVE ANO TO ULTIMO-ANO.
MOVE DIA TO DIA-ORDEN.
IF DIA-MENSUAL-USUARIO = 00 THEN
    MOVE ANO-USUARIO TO TRANSF-ANO
    MOVE MES-USUARIO TO TRANSF-MES
    MOVE DIA-USUARIO TO TRANSF-DIA
ELSE
    MOVE 0 TO TRANSF-ANO
    MOVE 0 TO TRANSF-MES
    MOVE DIA-MENSUAL-USUARIO TO TRANSF-DIA
END-IF.

WRITE TRANSF-REG.
IF FSTM <> 00 THEN
    PERFORM IMPRIMIR-CABECERA THRU IMPRIMIR-CABECERA
    DISPLAY FSTM LINE 9 COL 25
        WITH FOREGROUND-COLOR IS BLACK
        BACKGROUND-COLOR IS RED
    DISPLAY "Error escribir en transferencias" LINE 11 COL 32
        WITH FOREGROUND-COLOR IS BLACK
        BACKGROUND-COLOR IS RED
    DISPLAY "Enter - Aceptar" LINE 24 COL 33
    DISPLAY TRANSF-NUM LINE 25 COL 33
    GO TO EXIT-ENTER
END-IF.
CLOSE TRANSFERENCIAS.

```

También se ha modificado BANK1.cbl para que, al iniciar el programa, se lean las transferencias pendientes y se realicen todas las que se tendrían que haber realizado desde la última vez que se ejecutó el programa. Esto lo realiza LEER-TRANSF.

```

LEER-TRANSF.
READ TRANSFERENCIAS NEXT RECORD AT END GO TO P2.
*>DISPLAY TRANSF-DIA LINE 28 COL 20.
*>DISPLAY DIA-ORDEN LINE 28 COL 30.
*>DISPLAY TRANSF-MES LINE 28 COL 34.
*>DISPLAY ULTIMA-MENSUALIDAD LINE 29 COL 28.
IF (TRANSF-MES <> 00) THEN
    IF ((ANO > TRANSF-ANO OR

```



```

        (ANO = TRANSF-ANO AND MES > TRANSF-MES) OR
        (ANO = TRANSF-ANO AND MES > TRANSF-MES
        AND DIA > TRANSF-DIA))) THEN
            MOVE 1 TO TIPO-TRANSF
            GO TO PCONSULTA-SALDO
ELSE
    COMPUTE MES-VAR = ULTIMA-MENSUALIDAD + 1
    DISPLAY TARJETA-ORIGEN LINE 30 COL 28.
    IF((DIA >= TRANSF-DIA AND DIA-ORDEN <= TRANSF-DIA
    AND ULTIMA-MENSUALIDAD = 00)
    OR (MES > ULTIMA-MENSUALIDAD AND
    (DIA > TRANSF-DIA OR MES > MES-VAR) AND
    ULTIMA-MENSUALIDAD <> 00)) THEN
        MOVE 2 TO TIPO-TRANSF
        GO TO PCONSULTA-SALDO
END-IF.
GO TO LEER-TRANSF.

```

Cuando se encuentra una transferencia que hay que realizar, se calculan los saldos de las cuentas origen y destino (PCONSULTA-SALDO, LECTURA-MOV y LAST-MOV-FOUND).

```

PCONSULTA-SALDO.
    OPEN INPUT F-MOVIMIENTOS.
    IF FSM <> 00
        GO TO PSYS-ERR.

    MOVE 0 TO LAST-MOV-NUM.
    MOVE 0 TO LAST-USER-ORD-MOV-NUM.
    MOVE 0 TO LAST-USER-DST-MOV-NUM.

LECTURA-MOV.
    READ F-MOVIMIENTOS NEXT RECORD AT END GO LAST-MOV-FOUND.

    IF MOV-TARJETA = TARJETA-ORIGEN THEN
        IF LAST-USER-ORD-MOV-NUM < MOV-NUM THEN
            MOVE MOV-NUM TO LAST-USER-ORD-MOV-NUM
        END-IF
    END-IF.
    IF MOV-TARJETA = TARJETA-DESTINO THEN
        DISPLAY "FOUND" LINE 28 COL 28

        IF LAST-USER-DST-MOV-NUM < MOV-NUM THEN
            MOVE MOV-NUM TO LAST-USER-DST-MOV-NUM
        END-IF
    END-IF.
    IF LAST-MOV-NUM < MOV-NUM
        MOVE MOV-NUM TO LAST-MOV-NUM.
    GO LECTURA-MOV.

```

```

LAST-MOV-FOUND.
  CLOSE F-MOVIMIENTOS.

  IF TIPO-TRANSF = 1 THEN
    GO TO GUARDAR-TRANSF-1
  ELSE
    GO TO GUARDAR-TRANSF-2
  END-IF.

```

Después, se realizan las transferencias (registrándose como movimientos). De esto se encargan GUARDAR-TRANSF-1 (para transferencias puntuales) y GUARDAR-TRANSF-2 (para transferencias mensuales).

En el caso de las transferencias puntuales, simplemente se realizan las que tengan una fecha anterior a la actual. De eso se encarga GUARDAR-TRANSF-1, el cual guarda la transferencia en el archivo de movimientos de forma similar a como se hace al ordenar una transferencia para que se ejecute al momento.

```

GUARDAR-TRANSF-1.

  MOVE LAST-USER-ORD-MOV-NUM TO MOV-NUM.
  OPEN I-O F-MOVIMIENTOS.
  IF FSM <> 00
    GO TO PSYS-ERR.
  READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR.

  COMPUTE CENT-SALDO-ORD-USER = (MOV-SALDOPOS-ENT * 100)
    + MOV-SALDOPOS-DEC.
  COMPUTE CENT-IMPOR-USER = (TRANSF-IMPORTE-ENT * 100)
    + TRANSF-IMPORTE-DEC
  SUBTRACT CENT-IMPOR-USER FROM CENT-SALDO-ORD-USER.
  COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-ORD-USER / 100).
  MOVE FUNCTION MOD(CENT-SALDO-ORD-USER, 100)
    TO MOV-SALDOPOS-DEC.

  ADD 1 TO LAST-MOV-NUM.

  MOVE LAST-MOV-NUM TO MOV-NUM.
  MOVE TARJETA-ORIGEN TO MOV-TARJETA.
  MOVE TRANSF-DIA TO MOV-DIA.
  MOVE TRANSF-MES TO MOV-MES.
  MOVE TRANSF-ANO TO MOV-ANO.
  MOVE 0 TO MOV-HOR.
  MOVE 0 TO MOV-MIN.
  MOVE 0 TO MOV-SEG.
  MOVE "Transferimos." TO MOV-CONCEPTO.

```

```

MULTIPLY -1 BY TRANSF-IMPORTE-ENT.
MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT.
MULTIPLY -1 BY TRANSF-IMPORTE-ENT.
MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC.

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR.

CLOSE F-MOVIMIENTOS.
MOVE LAST-USER-DST-MOV-NUM TO MOV-NUM.
OPEN I-O F-MOVIMIENTOS.
IF FSM <> 00
    GO TO PSYS-ERR.
READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR.
COMPUTE CENT-SALDO-DST-USER = (MOV-SALDOPOS-ENT * 100)
    + MOV-SALDOPOS-DEC.
ADD CENT-IMPOR-USER TO CENT-SALDO-DST-USER.
COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-DST-USER / 100).
MOVE FUNCTION MOD(CENT-SALDO-DST-USER, 100)
    TO MOV-SALDOPOS-DEC.

ADD 1 TO LAST-MOV-NUM.

MOVE LAST-MOV-NUM TO MOV-NUM.
MOVE TARJETA-DESTINO TO MOV-TARJETA.
MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT.
MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC.
MOVE TRANSF-DIA TO MOV-DIA.
MOVE TRANSF-MES TO MOV-MES.
MOVE TRANSF-ANO TO MOV-ANO.
MOVE 0 TO MOV-HOR.
MOVE 0 TO MOV-MIN.
MOVE 0 TO MOV-SEG.
MOVE "Nos transfieren." TO MOV-CONCEPTO.

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR.
CLOSE F-MOVIMIENTOS.
GO TO LEER-TRANSF.

```

En el caso de las mensuales, en el fichero de transferencias se guarda el último mes y año en el que se realizó la transferencia (en ULTIMA-MENSUALIDAD y ULTIMO-ANO), por lo que con eso se pueden realizar todas las restantes desde ese mes en ese año hasta la actualidad. Esto lo realiza GUARDAR-TRANSF-2. Si se realiza en el mismo mes en el que se programó la transferencia entonces se guarda la transferencia de la misma forma que con las transferencias puntuales, solo que modificando también ULTIMA-MENSUALIDAD y ULTIMO-ANO en la transferencia del archivo de transferencias. Si han pasado uno o varios meses, entonces se realiza un bucle que guarda todas las transferencias en los movimientos (esto lo hace BUCLE).

GUARDAR-TRANSF-2.

IF (ULTIMA-MENSUALIDAD = 00) THEN

MOVE LAST-USER-ORD-MOV-NUM TO MOV-NUM

OPEN I-O F-MOVIMIENTOS

\*>IF FSM <> 00

\*>GO TO PSYS-ERR.

READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR

COMPUTE CENT-SALDO-ORD-USER = (MOV-SALDOPOS-ENT \* 100)  
+ MOV-SALDOPOS-DEC

COMPUTE CENT-IMPOR-USER = (TRANSF-IMPORTE-ENT \* 100)  
+ TRANSF-IMPORTE-DEC

SUBTRACT CENT-IMPOR-USER FROM CENT-SALDO-ORD-USER

COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-ORD-USER / 100)  
MOVE FUNCTION MOD(CENT-SALDO-ORD-USER, 100)  
TO MOV-SALDOPOS-DEC

ADD 1 TO LAST-MOV-NUM

MOVE LAST-MOV-NUM TO MOV-NUM

MOVE TARJETA-ORIGEN TO MOV-TARJETA

MOVE TRANSF-DIA TO MOV-DIA

MOVE TRANSF-MES TO MOV-MES

MOVE TRANSF-ANO TO MOV-ANO

MOVE 0 TO MOV-HOR

MOVE 0 TO MOV-MIN

MOVE 0 TO MOV-SEG

MOVE "Transferimos." TO MOV-CONCEPTO

MULTIPLY -1 BY TRANSF-IMPORTE-ENT

MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT

MULTIPLY -1 BY TRANSF-IMPORTE-ENT

MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR.

CLOSE F-MOVIMIENTOS.

MOVE LAST-USER-DST-MOV-NUM TO MOV-NUM

OPEN I-O F-MOVIMIENTOS.

\*>IF FSM <> 00

\*>GO TO PSYS-ERR.

READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR

COMPUTE CENT-SALDO-DST-USER = (MOV-SALDOPOS-ENT \* 100)  
+ MOV-SALDOPOS-DEC

ADD CENT-IMPOR-USER TO CENT-SALDO-DST-USER

COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-DST-USER / 100)

MOVE FUNCTION MOD(CENT-SALDO-DST-USER, 100)  
TO MOV-SALDOPOS-DEC.

ADD 1 TO LAST-MOV-NUM  
MOVE LAST-MOV-NUM TO MOV-NUM  
MOVE TARJETA-DESTINO TO MOV-TARJETA

MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT  
MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC  
MOVE TRANSF-DIA TO MOV-DIA  
MOVE TRANSF-MES TO MOV-MES  
MOVE TRANSF-ANO TO MOV-ANO  
MOVE 0 TO MOV-HOR  
MOVE 0 TO MOV-MIN  
MOVE 0 TO MOV-SEG  
MOVE "Nos transfieren." TO MOV-CONCEPTO

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR  
CLOSE F-MOVIMIENTOS

IF ((TRANSF-MES > ULTIMA-MENSUALIDAD  
AND TRANSF-ANO = ULTIMO-ANO)  
OR TRANSF-ANO > ULTIMO-ANO)  
MOVE TRANSF-MES TO ULTIMA-MENSUALIDAD

MOVE FUNCTION MAX(TRANSF-ANO ULTIMO-ANO) TO ULTIMO-ANO

REWRITE TRANSF-REG

GO TO LEER-TRANSF

ELSE

COMPUTE NUM-MENSUALIDADES = ANO - ULTIMO-ANO  
MULTIPLY 12 BY NUM-MENSUALIDADES  
IF (MES > ULTIMA-MENSUALIDAD)  
ADD MES TO NUM-MENSUALIDADES  
SUBTRACT ULTIMA-MENSUALIDAD FROM NUM-MENSUALIDADES  
ELSE  
ADD ULTIMA-MENSUALIDAD TO NUM-MENSUALIDADES  
SUBTRACT MES FROM NUM-MENSUALIDADES

IF (DIA < TRANSF-DIA) THEN  
SUBTRACT 1 FROM NUM-MENSUALIDADES

MOVE ULTIMA-MENSUALIDAD TO BUCLE-MES  
ADD 1 TO BUCLE-MES  
MOVE ULTIMO-ANO TO BUCLE-ANO  
ADD 1 TO BUCLE-ANO

MOVE LAST-USER-ORD-MOV-NUM TO MOV-NUM  
OPEN I-O F-MOVIMIENTOS

```

IF FSM <> 00
  GO TO PSYS-ERR
READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR

COMPUTE CENT-SALDO-ORD-USER = (MOV-SALDOPOS-ENT * 100)
  + MOV-SALDOPOS-DEC
COMPUTE CENT-IMPOR-USER = (TRANSF-IMPORTE-ENT * 100)
  + TRANSF-IMPORTE-DEC
*>SUBTRACT CENT-IMPOR-USER FROM CENT-SALDO-ORD-USER.
*>COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-ORD-USER / 100)
*>MOVE FUNCTION MOD(CENT-SALDO-ORD-USER, 100)
  *>TO MOV-SALDOPOS-DEC
CLOSE F-MOVIMIENTOS

MOVE LAST-USER-DST-MOV-NUM TO MOV-NUM
OPEN I-O F-MOVIMIENTOS
IF FSM <> 00
  GO TO PSYS-ERR
READ F-MOVIMIENTOS INVALID KEY GO PSYS-ERR
COMPUTE CENT-SALDO-DST-USER = (MOV-SALDOPOS-ENT * 100)
  + MOV-SALDOPOS-DEC
CLOSE F-MOVIMIENTOS
*>ADD CENT-IMPOR-USER TO CENT-SALDO-DST-USER

```

END-IF.

BUCLE.

```

OPEN I-O F-MOVIMIENTOS

ADD 1 TO LAST-MOV-NUM

SUBTRACT CENT-IMPOR-USER FROM CENT-SALDO-ORD-USER.
COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-ORD-USER / 100).
MOVE FUNCTION MOD(CENT-SALDO-ORD-USER, 100)
  TO MOV-SALDOPOS-DEC.

MOVE LAST-MOV-NUM TO MOV-NUM
MOVE TARJETA-ORIGEN TO MOV-TARJETA
MOVE TRANSF-DIA TO MOV-DIA
MOVE BUCLE-MES TO MOV-MES
MOVE BUCLE-ANO TO MOV-ANO
MOVE 0 TO MOV-HOR
MOVE 0 TO MOV-MIN
MOVE 0 TO MOV-SEG
MOVE "Transferimos." TO MOV-CONCEPTO

MULTIPLY -1 BY TRANSF-IMPORTE-ENT
MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT
MULTIPLY -1 BY TRANSF-IMPORTE-ENT
MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC

```

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR

ADD 1 TO LAST-MOV-NUM  
ADD CENT-IMPOR-USER TO CENT-SALDO-DST-USER  
COMPUTE MOV-SALDOPOS-ENT = (CENT-SALDO-DST-USER / 100).  
MOVE FUNCTION MOD(CENT-SALDO-DST-USER, 100)  
TO MOV-SALDOPOS-DEC.

MOVE LAST-MOV-NUM TO MOV-NUM  
MOVE TARJETA-DESTINO TO MOV-TARJETA  
MOVE TRANSF-IMPORTE-ENT TO MOV-IMPORTE-ENT  
MOVE TRANSF-IMPORTE-DEC TO MOV-IMPORTE-DEC  
MOVE TRANSF-DIA TO MOV-DIA  
MOVE BUCLE-MES TO MOV-MES  
MOVE BUCLE-ANO TO MOV-ANO  
MOVE 0 TO MOV-HOR  
MOVE 0 TO MOV-MIN  
MOVE 0 TO MOV-SEG  
MOVE "Nos transfieren." TO MOV-CONCEPTO

WRITE MOVIMIENTO-REG INVALID KEY GO TO PSYS-ERR

ADD 1 TO BUCLE-MES  
IF (BUCLE-MES < 12)  
MOVE FUNCTION MOD(BUCLE-MES,12) TO BUCLE-MES  
IF (BUCLE-MES = 1)  
ADD 1 TO BUCLE-ANO

SUBTRACT 1 FROM NUM-MENSUALIDADES

IF (NUM-MENSUALIDADES = 0)  
IF ((TRANSF-MES > ULTIMA-MENSUALIDAD  
AND BUCLE-ANO = ULTIMO-ANO)  
OR BUCLE-ANO > ULTIMO-ANO) THEN  
MOVE BUCLE-MES TO ULTIMA-MENSUALIDAD

IF (BUCLE-ANO > ULTIMO-ANO) THEN  
MOVE BUCLE-ANO TO ULTIMO-ANO

REWRITE TRANSF-REG  
GO TO LEER-TRANSF  
ELSE  
GO TO BUCLE  
END-IF.

Para todas estas modificaciones de BANK1.cbl para el procesado de transferencias programadas puntuales y mensuales al programa, se han introducido en WORKING-STORAGE SECTION:

```

77 LAST-MOV-NUM          PIC 9(35).
77 TIPO-TRANSF           PIC 9(1).
77 MES-VAR               PIC 9(2).

77 LAST-USER-ORD-MOV-NUM PIC 9(35).
77 LAST-USER-DST-MOV-NUM PIC 9(35).
77 CENT-SALDO-ORD-USER   PIC S9(9).
77 CENT-SALDO-DST-USER   PIC S9(9).
77 CENT-IMPOR-USER       PIC S9(9).

77 NUM-MENSUALIDADES     PIC 9(4).
77 BUCLE-MES             PIC 9(4).
77 BUCLE-ANO             PIC 9(4).

```

Además, tanto en BANK1.cbl como en BANK6.cbl se ha introducido lo siguiente en FILE-SECTION para acceder al archivo que almacena las transferencias programadas:

```

FD TRANSFERENCIAS
  LABEL RECORD STANDARD
  VALUE OF FILE-ID IS "transf.txt".
01 TRANSF-REG.
  02 TRANSF-NUM          PIC 9(35).
  02 TARJETA-ORIGEN      PIC 9(16).
  02 TARJETA-DESTINO     PIC 9(16).
  02 TRANSF-IMPORTE-ENT  PIC S9(7).
  02 TRANSF-IMPORTE-DEC  PIC 9(2).
  02 TRANSF-DIA          PIC 9(2).
  02 DIA-ORDEN           PIC 9(2).
  02 TRANSF-MES          PIC 9(2).
  02 TRANSF-ANO          PIC 9(4).
  02 ULTIMA-MENSUALIDAD  PIC 9(2).
  02 ULTIMO-ANO          PIC 9(4).

```

Y lo siguiente en FILE-CONTROL:

```

SELECT TRANSFERENCIAS ASSIGN TO DISK
  ORGANIZATION IS INDEXED
  ACCESS MODE IS DYNAMIC
  RECORD KEY IS TRANSF-NUM
  FILE STATUS IS FSTM.

```

Con todas estas modificaciones para las transferencias programadas, el programa da error al intentar reescribir una de las entradas del archivo de transferencias programadas (aparecía error de clave repetida, aunque sólo se estaba reescribiendo y no se modificaba la clave). Por eso, los ficheros BANK1.cbl y BANK6.cbl con estas funciones se han dejado en una carpeta aparte (Pr1-transacciones-futuras) y se han usado otras versiones que lo



que hacen es: BANK6.cbl escribe directamente en los movimientos una transferencia puntual, mientras que para una transferencia mensual lo que se hace es escribir 12 transferencias en movimientos que se realizan en el día elegido de cada uno de los 12 meses siguientes a cuando se ha programado la transferencia. BANK1.cbl no hace nada extra (no lee transferencias programadas).

## Paso 3 - Mejoras del programa final

Para la mejora de la interfaz de la opción 4 "Ingreso efectivo", se ha modificado el fichero "BANK5.cbl". Primero se ha creado un nuevo display para pedir al usuario que introduzca el número de billetes en concreto:

```
01 ENTRADA-USUARIO.  
  05 FILLER BLANK ZERO AUTO UNDERLINE  
    LINE 13 COL 40 PIC 9(2) USING BILL10-USUARIO.  
  05 FILLER BLANK ZERO AUTO UNDERLINE  
    LINE 14 COL 40 PIC 9(2) USING BILL20-USUARIO.  
  05 FILLER BLANK ZERO UNDERLINE  
    LINE 15 COL 40 PIC 9(2) USING BILL50-USUARIO.
```

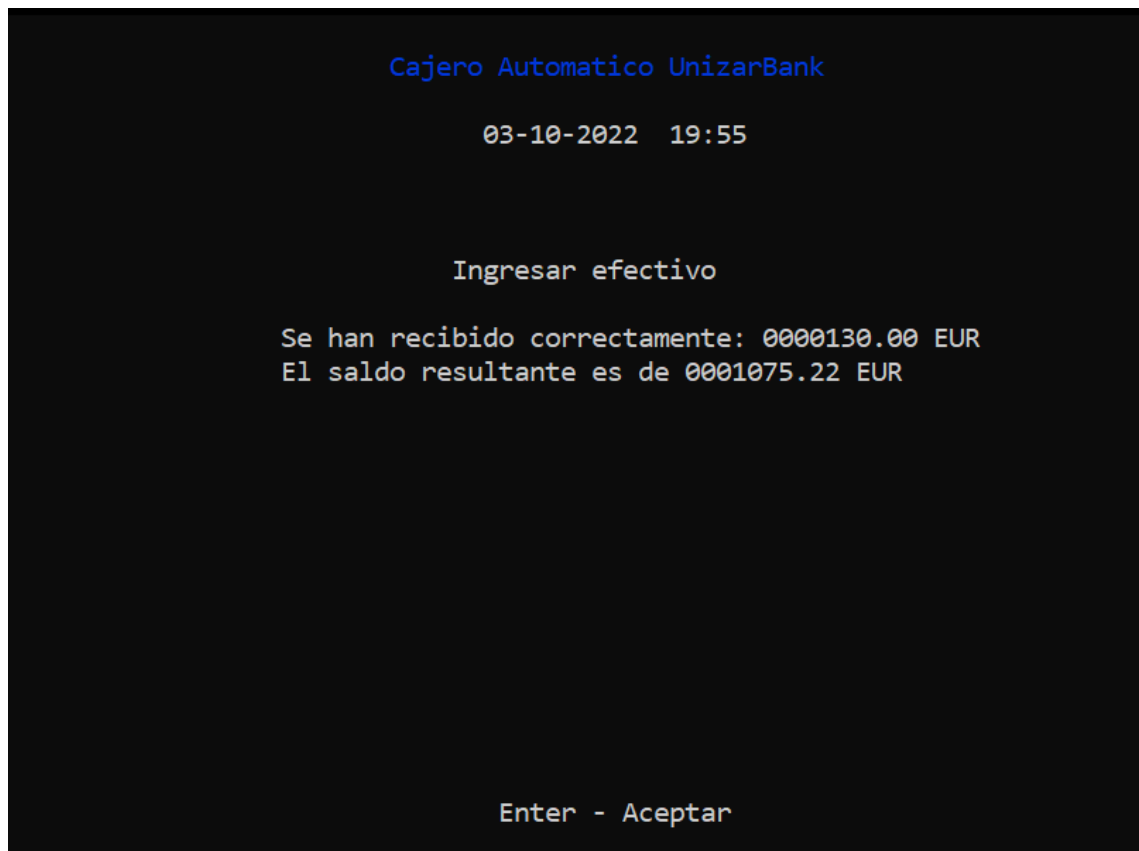
Una vez que el usuario introduzca los datos, se procesa la información que como se hacía anteriormente pero ahora, que no hay no hay céntimos, hay que multiplicar con el valor del billete introducido:

```
COMPUTE EURENT-USUARIO = (BILL10-USUARIO*10 +  
                          BILL20-USUARIO*20 +  
                          BILL50-USUARIO*50).  
MOVE 00 TO EURDEC-USUARIO.  
COMPUTE CENT-IMPOR-USER = (EURENT-USUARIO * 100)  
                          + EURDEC-USUARIO.  
ADD CENT-IMPOR-USER TO CENT-ACUMULADOR.
```

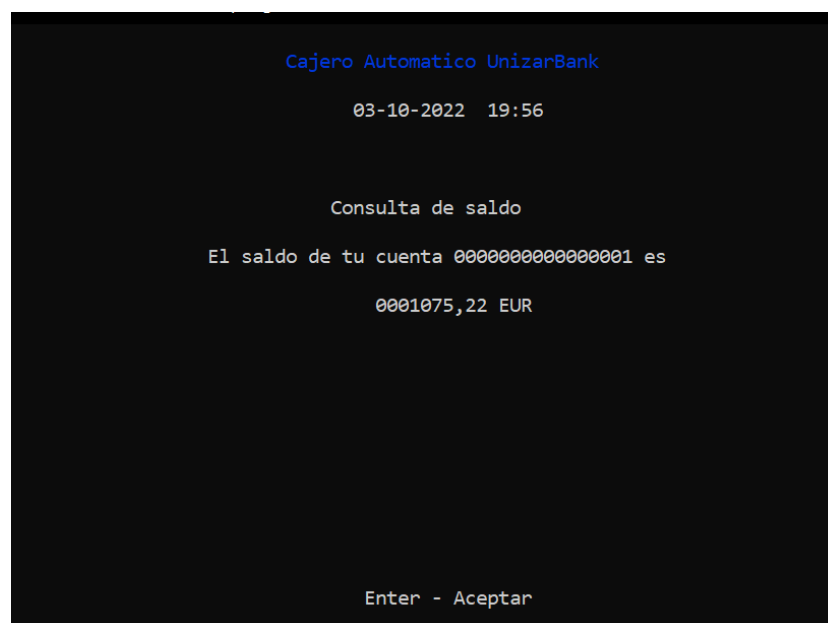
Así es el resultado de ejecutar la operación:

```
Cajero Automatico UnizarBank  
  
03-10-2022  19:54  
  
Ingresar efectivo  
  
Saldo Actual:  0000945.22 EUR  
Por favor, introduzca los billetes:  
  
Billletes de 10 EUR: 01  
Billletes de 20 EUR: 01  
Billletes de 50 EUR: 02  
  
ESC - Finalizar ingreso efectivo
```

Se le pide al usuario el número de billetes que desea introducir, una vez que los introduzca y pulse el ENTER aparecerá la siguiente pantalla:



Aparecerá un resumen con el total del valor de los billetes introducidos, así como el saldo restante después de la operación. Si pulsamos al ENTER volveremos al menú inicial, si de nuevo consultamos el saldo comprobamos que se ha guardado correctamente.



Por otra parte, para eliminar los ceros que aparecían cuando no tocaba en ejecución, vimos que se debía a que ACCEPT estaba capturando los valores introducidos cuando simplemente esperaba que pulsases una tecla en concreto para avanzar.

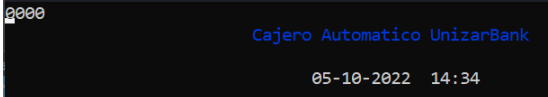
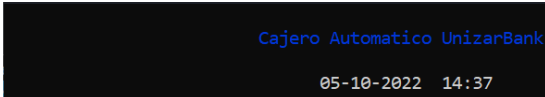
En este caso ACCEPT va a capturar CHOICE, que es una variable que guardará un valor numérico que el usuario introduzca [PIC 9(1)].

```
ACCEPT CHOICE LINE 20 COL 52 ON EXCEPTION
```

Pero si usamos OMITTED lo que conseguimos es que no se capture el valor introducido por el usuario y por consiguiente no aparezcan 1 o más ceros en ejecución.

```
ACCEPT OMITTED ON EXCEPTION  
IF ENTER-PRESSED  
GO TO P2
```

Y el resultado conseguido es el siguiente:

	
--	---

Por último, para garantizar que la ventana de ejecución es siempre de 80x25 caracteres, no se supera 24 en líneas y 79 en columnas:

```
LINE 24 COL 79 ON EXCEPTION
```

**NOTA:** El ejecutable de la práctica está en la carpeta “bin”, es el archivo **BANK1.exe**.