Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Iñigo Gascón Royo 685215
Rubén Moreno Jimeno 680882
Guillermo Robles González 604409

Sistemas Legados
Práctica 1
Añadir una nueva funcionalidad a un programa COBOL

7º cuatrimestre
Ingeniería Informática
18 de octubre de 2016,
Zaragoza

Contenido

1. Asegurar que la aplicación funciona correctamente

Para empezar el proyecto lo primero se ha escogido el compilador cobol con el que se iba a trabajar, el cual ha sido OpenCobol. El fichero original se ha dividido en 3 ficheros: main.cob, EMPWRITE.cob y EMPREAD.cob, para una lectura más sencilla, al ser un fichero con muchísimas líneas.

Se han realizado algunos cambios de formatos en partes del código:

- Los formatos de los "DISPLAY" y "ACCEPT" que se han proporcionado se han sustituido por un formato que admite el compilador OpenCobol.
- Los formatos de los SELECT y los FD se ha cambiado al formato de OpenCOBOL, eliminando las opciones LABEL y VALUE, y moviendo la ruta del fichero a la sentencia SELECT.

De esta forma, se ha conseguido compilar el código proporcionado originalmente, aunque la división en diferentes ficheros no era estrictamente necesaria.

Aunque tampoco era necesario, se ha usado la característica de los "copybooks". Ésta nos permite crear archivos especiales con extensión cpy, los cuales podemos insertar en lugares determinados en el código (de manera similar a los *inline*) de lenguajes modernos. En nuestro caso se han usado 2, **FS-MSG**, el cual almacena los diferentes códigos de error que pueden ocurrir al realizar acciones sobre ficheros, de manera que para comprobar el error que ha ocurrido en el programa utilizamos la siguiente sentencia:

```
*>>D    COPY FS-MSG REPLACING STATUS BY FSB
*>>D    MSG     BY FS_MSG.
*>>D    STRING "OPEN I-O BRANCHFILE.: " FS_MSG INTO STUFF.
*>>D    DISPLAY STUFF AT 3099.
```

La cual nos muestra el mensaje de error de manera textual.

El otro copybook que se ha utilizado es el de **CLEAR-SCREEN**, dado que eso nos permite portar el código a otros dialectos, sin más que cambiar de copybook.

También se ha utilizado ampliamente la opción de debug del compilador seleccionado (OpenCobol). Esta consiste en que si el séptimo carácter de una línea es una "D", esa línea será ignorada a menos que se pase la opción "-fdebugging-line".

Se ha decidido automatizar la compilación mediante "make". Para ello simplemente se ha tenido que crear un fichero Makefile el cual contiene todos los objetivos de compilación. En nuestro caso solo se han usado 2, *all* y *debug*, siendo la única diferencia entre ellos el uso de la opción "-fdebugging-line" comentada anteriormente.

2. Nueva funcionalidad: “LIST BRANCH FILE”

Se ha editado main.cob para añadir la nueva opción del menú y se ha implementado la funcionalidad en el nuevo fichero BRANCHLIST.cob.

No se ha especificado el formato en el que se identifica una ciudad, por tanto, se tomado la decisión de identificar las ciudades por una cadena de 3 caracteres, que se coloca al final de la dirección, separada por una coma. Esta decisión de diseño se basa en el hecho de que la dirección mide solo 30 caracteres, y eso podría dar problemas con calles con nombres largos (Ej.: “Calle Franco y López, n 12, Zaragoza, España”, con 44 caracteres).

El programa se divide en dos secciones: definiciones y procedimientos. En la sección de definiciones se agrupan declaraciones de variables o segmentos usados para definir ciertas características. A parte de las secciones ya usadas en el programa principal, se han usado dos extra:

- **SPECIAL-NAMES**: Se ha usado para la variable especial **CRT-STATUS**. Esta variable almacena el contenido de la tecla pulsada en las sentencias **ACCEPT**. En COBOL, en el momento en el que se llega a una sentencia **ACCEPT** el flujo del programa se detiene, y queda a la espera de una pulsación de teclado. Si el usuario pulsa un carácter imprimible (letras, números, símbolos), estos se imprimen por pantalla. Sin embargo, si se pulsa una tecla especial (Enter, F<n>, Tab…) el **ACCEPT** termina, guardando en **CRT-STATUS** un código que identifica la tecla pulsada (por ejemplo, Enter es 1000, F<n> es 1000+n…) esto nos permite reaccionar de formas especiales al pulsado de ciertas teclas, como se ejemplifica en la pantalla de listado.
- **SCREEN SECTION:** Esta sección contiene configuraciones de pantalla, como colores o estructuras de TUI (*Terminal User Interface*). En nuestro caso se ha usado para describir diferentes estructuras a imprimir (**HEADER**, usada para la cabecera de la tabla; y **ROW** usada para cada fila de la tabla), esto nos permite modificar con comodidad toda la estructura de la interfaz de usuario desde un mismo lugar de manera cómoda.

En la sección de procedimientos solo existe uno, llamado **MAIN-PARA**, dado que no hemos considerado necesaria mayor subdivisión. Dicho procedimiento se divide en 3 partes:

1. **Inicialización (Líneas 87-97):** Sirve para situar el programa en un estado consistente, rellena variables, abre ficheros, prepara la pantalla y pregunta al usuario la ciudad a mostrar.
2. **Bucle (Líneas 98-136):** Repite hasta que se acaba el fichero el siguiente proceso:
    - Lee un registro.
    - Comprueba si cumple la condición de ciudad definida (Si la ciudad es vacía lo muestra, si contiene algo comprueba si la dirección pertenece a la ciudad dada).
    - Imprime la fila.
    - Si ya ha impreso 10 o ha llegado al final del fichero, se detiene a esperar la entrada del usuario.

- Si el usuario selecciona siguiente, borra la pantalla y vuelve al principio, si el usuario selecciona salir sale de la pantalla.

3. **Finalización (Líneas 137-157):** Espera a que el usuario de la orden y vuelven al menú principal, pero antes rebobina el fichero, lo cual se hace leyendo los registros en dirección contraria hasta llegar al inicio de nuevo. Esto es necesario porque el compilador que se ha usado tiene la característica de mantener la posición del cursor entre cerrados y aperturas del fichero, por lo que si se eligiera "List Branch file" varias veces, a partir de la primera solo se mostraría el registro final.

La comprobación de ciudad se ha hecho mediante la sentencia **INSPECT BBRADD TALLYING NUMCITY FOR ALL CITYT.**

3. Inserción de datos

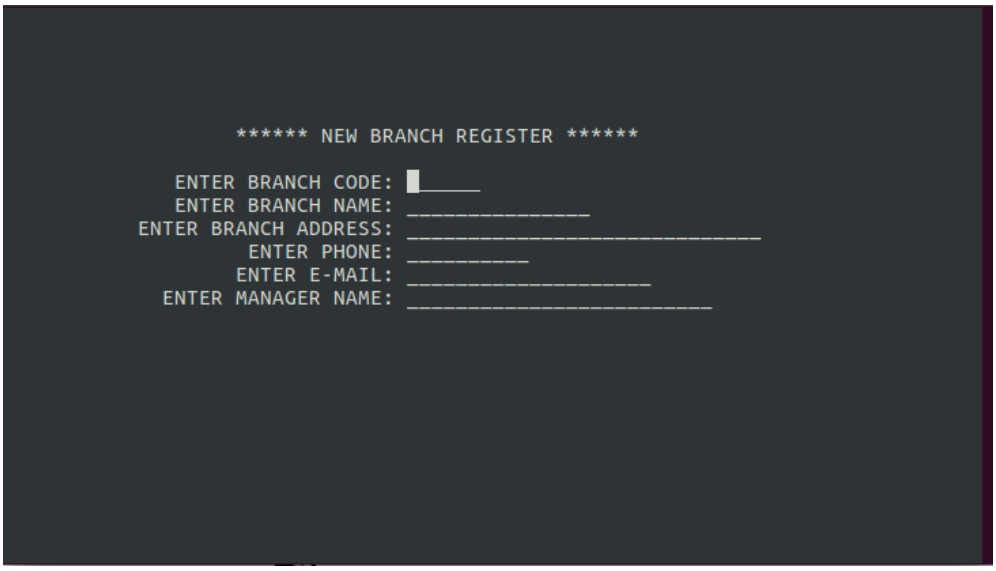
En primer lugar, se ha probado que la aplicación introduce datos correctamente en el sistema, mediante el uso de la interfaz TUI. Una vez decidimos que la TUI funcionaba correctamente (tras la inserción de 5 registros), se decidió automatizar la introducción de datos. Esto se hizo mediante un script bash, el cual genera un programa en COBOL que, tras compilarlo y ejecutarlo, introduce los registros deseados en el fichero. Gracias a este programa comprobamos que el sistema de paginado maneja adecuadamente varios miles de registros.

4. Optativo B: Mejora de la interfaz

Se ha realizado lo que se pedía poniendo todos los **DISPLAY** primero y a continuación todos los **ACCEPT**. Esto se hace así porque **ACCEPT** es una operación bloqueante. Otra posibilidad sería utilizar la ya mencionada sección **SCREEN SECTION** para almacenar todos los formularios, de tal forma que su mostrado se resuma a una línea. Para mostrar la longitud de los campos han sido necesarios dos cambios:

- En los **DISPLAY** se ha insertado manualmente tantas barras bajas como caracteres tiene el campo.
- En los **ACCEPT** se ha usado la opción **WITH UNDERLINE**, la cual indica que queremos mostrar la longitud del **ACCEPT** con barras bajas.

Por último, se ha retocado un poco la interfaz para que el resultado final quedara parecido al del guion de la práctica:



5. Problemas encontrados y organización del trabajo

Los principales problemas a los que se ha enfrentado el grupo a la hora de realizar la práctica han sido los siguientes:

- Escasa documentación (no obstante, la documentación encontrada en OpenCobol es bastante útil).
- Inexistencia de un IDE de COBOL bueno.
- En los menús con más de 9 opciones, existen variables con 1 y 2 dígitos. Eso provocaba que, al meter una opción de menú con 1 dígito, COBOL automáticamente añade un espacio después para rellenar la variable de 2 dígitos, lo cual provocaba que compararlo con "1" no fuese cierto. Como solución, se ha programado que las opciones de menú con un solo dígito se comparen con ese dígito más un espacio para que la comparación sea correcta.

En lo referente al reparto de tareas y organización del trabajo, tanto la realización de la práctica como la redacción de la memoria se ha realizado en grupo en todo momento quedando los 3 miembros de forma presencial o vía Skype participando y consensuando así, todas las partes de la práctica.

6. Referencias

- Web de la asignatura *[http://webdiis.unizar.es/asignaturas/SL/]*
- Escobol *[http://www.escobol.com/]*
- *Libro: "Cobol/ Pedro Jarillo Cerrato, Ma. Dolores Jarillo Cerrato. ISBN: 8476145934"*
- *Libro: "Lenguaje de programación COBOL/ J. Astor Vingau, ISBN: 8460014827"*
- *OpenCOBOL 1.1 Programmer's Guide 1st Edition, 17 September 2010 [*http://open-cobol.sourceforge.net/guides/OpenCOBOL%20Programmers%20Guide.pdf*]*