

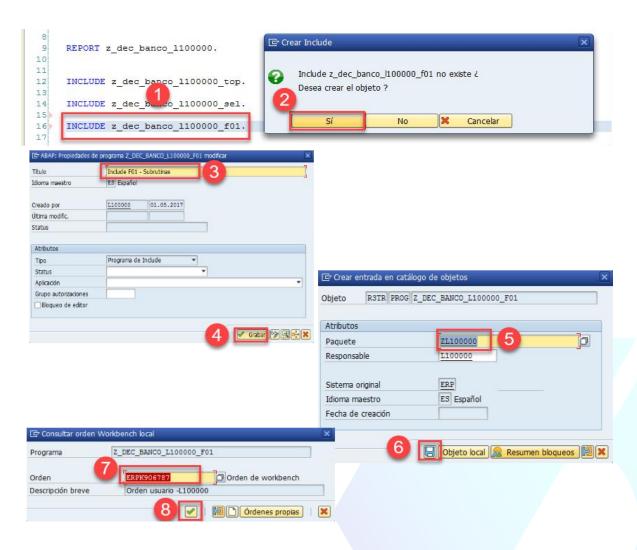
## SOLUCIÓN EJERCICIO Subrutinas

SAP ABAP Programación



- En la transacción del Editor ABAP modifique el programa
   Z\_DEC\_BANCO\_USUARIOSAP\*.
- 2. Añada la sentencia del include que va a contener las subrutinas.

3. Con doble clic sobre el nombre del include cree el objeto agrupándolo en el paquete de desarrollo y asignando la orden de transporte.



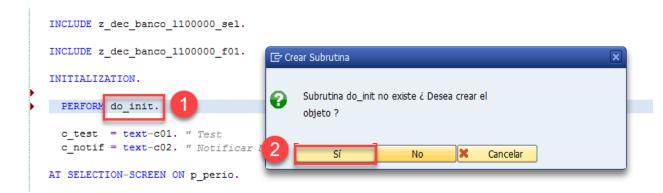
- 4. Active el include y el programa principal.
- 5. Cree la llamada a la subrutina DO\_INIT en el evento INITIALIZATION.

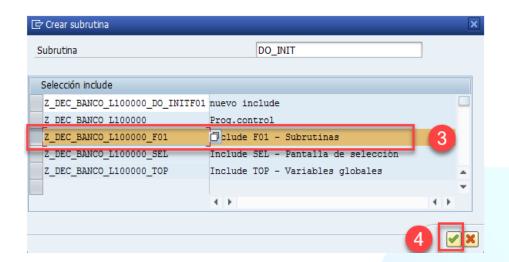
```
INITIALIZATION.

PERFORM do_init.

c_test = text-c01. " Test
c_notif = text-c02. " Notificar Manager
```

6. Con doble clic sobre el nombre de la subrutina cree el objeto agrupándolo en el include Z\_DEC\_BANCO\_USUARIOSAP\_F01.





7. Mueva todo código del evento INITIALIZATION en la subrutina DO\_INIT.

8. Active el include y el programa principal. Ahora en el evento INITIALIZATION existe sólo una llamada a la subrutina DO\_INIT, ya que se encapsuló toda la lógica dentro de la subrutina.

```
INITIALIZATION.

PERFORM do init.
```

- 9. Repita los mismos pasos creando la subrutina CHECK\_VALUES asignando todo el código del evento AT-SELECTION-SCREEN-ON p\_perio.
- 10. Repita los mismos pasos creando la subrutina EXECUTE\_TASK asignando todo el código del evento START-OF-SELECTION.
- 11. Active el include de las subrutinas y el programa principal.

## 12. El código del programa principal ya modularizado.

```
*&----*

*& Report Z_DEC_BANCO_L100000

*&

*&

*&-----*

*&

*&

*&-----*

REPORT z_dec_banco_l100000.

INCLUDE z_dec_banco_l100000_top.

INCLUDE z_dec_banco_l100000_sel.

INCLUDE z_dec_banco_l100000_fol.

INITIALIZATION.

PERFORM do_init.

AT SELECTION-SCREEN ON p_perio.

PERFORM check_values.

START-OF-SELECTION.

PERFORM execute_task.
```

13.El código del include con las subrutinas Z\_DEC\_BANCO\_USUARIOSAP\_F01.

```
FORM do init.
 c test = text-c01. " Test
 c notif = text-c02. " Notificar Manager
                         " DO INIT
ENDFORM.
*& Form CHECK_VALUES
* text
  --> p1
              text
* <-- p2
____*
FORM check_values .
 IF p_perio LT 1 OR p_perio GT 12.
   MESSAGE e001(z_1100000).
 ENDIF.
                        " CHECK VALUES
ENDFORM.
*& Form EXECUTE_TASK
*&-----
* text
* --> p1 text
* <-- p2 text
FORM execute task .
 CASE abap true.
   WHEN p_create. " Create
* Sociedad
    gwa_inform_banco-sociedad = p_soc.
* Ejercicio
    gwa inform banco-ejercicio = p ejer.
* Período contable
     gwa_inform_banco-per_cont = p_perio.
* Acreedor
     IF p acre IS NOT INITIAL.
      gwa_inform_banco-tipo_ejec = 'A'.
* Deudor
```

```
ELSEIF p deud IS NOT INITIAL.
       gwa_inform_banco-tipo_ejec = 'D'.
     ENDIF.
* Usuario responsable
     gwa inform banco-usuario res = sy-uname.
* Fecha de ejecución
     gwa inform banco-fecha eje = sy-datum.
* INSERT en base de datos
     INSERT zib 1100000 FROM gwa inform banco.
* Mensajes de información
     IF sy-subrc EQ 0.
       MESSAGE i002(z 1100000).
     ELSE.
       MESSAGE i003(z 1100000).
     ENDIF.
   WHEN p_read. " Read
     SELECT SINGLE * FROM zib 1100000
        INTO gwa_inform_banco
        WHERE sociedad EQ p_soc
          AND ejercicio EQ p_ejer
          AND per cont EQ p perio.
     IF sy-subrc EQ 0.
       WRITE: / 'Sociedad = ',
                gwa inform banco-sociedad,
               / 'Ejercicio = ',
                gwa inform banco-ejercicio,
               / 'Período contable = ',
                gwa inform banco-per cont,
               / 'Tipo ejecución = ',
                gwa inform banco-tipo ejec,
               / 'Usuario responsable = ',
                gwa inform banco-usuario res,
               / 'Fecha de ejecución = ',
                gwa inform banco-fecha eje MM/DD/YYYY.
     ELSE.
       MESSAGE i004(z 1100000).
     ENDIF.
   WHEN p_update. " Update
     CASE abap true.
       WHEN p acre.
         Update tipo ejecución con Acreedor
         UPDATE zib 1100000 SET tipo ejec = 'A'
          WHERE sociedad
                          EQ p_soc
           AND ejercicio EQ p ejer
           AND per cont
                            EQ p_perio.
```

```
WHEN p_deud.
         Update tipo ejecución con Deudor
         UPDATE zib 1100000 SET tipo ejec = 'D'
          WHERE sociedad EQ p soc
            AND ejercicio EQ p_ejer
            AND per cont EQ p perio.
     ENDCASE.
     IF sy-subrc EQ 0.
       MESSAGE i005(z_1100000).
       MESSAGE i006(z 1100000).
     ENDIF.
   WHEN p delete. " Delete
* Eliminación del registro
* de base de datos
     DELETE FROM zib 1100000
      WHERE sociedad EQ p soc
        AND ejercicio EQ p ejer
        AND per cont EQ p perio.
     IF sy-subrc EQ 0.
       MESSAGE i007(z 1100000).
     ELSE.
       MESSAGE i008(z 1100000).
     ENDIF.
   WHEN p modify. " Modify
* Sociedad
     gwa inform banco-sociedad = p soc.
* Ejercicio
     gwa inform banco-ejercicio = p ejer.
* Período contable
     gwa_inform_banco-per_cont = p_perio.
* Acreedor
     IF p acre IS NOT INITIAL.
       gwa inform banco-tipo ejec = 'A'.
* Deudor
     ELSEIF p deud IS NOT INITIAL.
       gwa_inform_banco-tipo_ejec = 'D'.
     ENDIF.
* Usuario responsable
     gwa inform banco-usuario res = sy-uname.
* Fecha de ejecución
     gwa_inform_banco-fecha_eje = sy-datum.
* MODIFY en base de datos
     MODIFY zib_1100000 FROM gwa inform banco.
* Mensajes de información
```

```
IF sy-subrc EQ 0.
    MESSAGE i009(z_1100000).
ELSE.
    MESSAGE i010(z_1100000).
ENDIF.

WHEN OTHERS.

ENDCASE.

ENDFORM.

" EXECUTE TASK
```

14. Siga con la segunda parte del ejercicio modularizando el código de la subrutina EXECUTE\_TASK en subrutinas anidadas. Empiece con la lógica del **CASE p\_update** creando la subrutina UPDATE\_RECORD. La subrutina se debe crear en el mismo include Z\_DEC\_BANCO\_USUARIOSAP\_F01.

```
WHEN p_update. " Update
PERFORM update record.
```

15. Mueva todo el código del CASE p\_update en la subrutina creada.

```
*& Form UPDATE RECORD
      text
* --> p1
               text
  <-- p2
               text
FORM update_record.
 CASE abap true.
   WHEN p acre.
        Update tipo ejecución con Acreedor
     UPDATE zib 1100000 SET tipo ejec = 'A'
      WHERE sociedad EQ p_soc
       AND ejercicio EQ p_ejer
       AND per_cont
                     EQ p_perio.
   WHEN p deud.
        Update tipo ejecución con Deudor
     UPDATE zib 1100000 SET tipo ejec = 'D'
      WHERE sociedad EQ p_soc
        AND ejercicio EQ p ejer
```

```
AND per_cont EQ p_perio.
ENDCASE.

IF sy-subrc EQ 0.
   MESSAGE i005(z_l100000).
ELSE.
   MESSAGE i006(z_l100000).
ENDIF.

ENDFORM.

" UPDATE RECORD
```

16. Repita el mismo paso creando la subrutina DELETE\_RECORD para la lógica del CASSE p\_delete.

```
WHEN p_delete. " Delete
PERFORM delete record.
```

17. Mueva todo el código del CASE p\_delete en la subrutina creada.

```
*& Form DELETE_RECORD
* text
* --> p1 text
* <-- p2 text
FORM delete record.
* Eliminación del registro
* de base de datos
 DELETE FROM zib 1100000
  WHERE sociedad EQ p soc
    AND ejercicio EQ p_ejer
    AND per_cont EQ p_perio.
  IF sy-subrc EQ 0.
   MESSAGE i007(z 1100000).
  ELSE.
   MESSAGE i008(z_1100000).
 ENDIF.
ENDFORM.
                            " DELETE RECORD
```

## 18. Código de la implementación final de la subrutina EXECUTE\_TASK.

```
Form EXECUTE_TASK
* text
* --> p1
* <-- p2
               text
FORM execute task .
 CASE abap true.
   WHEN p create. " Create
* Sociedad
     gwa inform banco-sociedad = p soc.
* Ejercicio
     gwa inform banco-ejercicio = p ejer.
* Período contable
     gwa_inform_banco-per_cont = p_perio.
* Acreedor
     IF p acre IS NOT INITIAL.
       gwa inform banco-tipo ejec = 'A'.
* Deudor
     ELSEIF p_deud IS NOT INITIAL.
       gwa inform banco-tipo ejec = 'D'.
     ENDIF.
* Usuario responsable
     gwa_inform_banco-usuario_res = sy-uname.
* Fecha de ejecución
     gwa inform banco-fecha eje = sy-datum.
* INSERT en base de datos
     INSERT zib 1100000 FROM gwa inform banco.
* Mensajes de información
     IF sy-subrc EQ 0.
       MESSAGE i002(z 1100000).
       MESSAGE i003(z 1100000).
     ENDIF.
```

WHEN p read. " Read

```
SELECT SINGLE * FROM zib 1100000
        INTO gwa inform banco
        WHERE sociedad EQ p_soc
          AND ejercicio EQ p ejer
          AND per cont EQ p perio.
     IF sy-subrc EQ 0.
       WRITE: / 'Sociedad = ',
                gwa_inform_banco-sociedad,
               / 'Ejercicio = ',
                gwa inform banco-ejercicio,
               / 'Período contable = ',
                gwa inform banco-per cont,
               / 'Tipo ejecución = ',
                gwa inform banco-tipo ejec,
               / 'Usuario responsable = ',
                gwa inform banco-usuario res,
               / 'Fecha de ejecución = ',
                gwa inform banco-fecha eje MM/DD/YYYY.
     ELSE.
       MESSAGE i004(z 1100000).
   WHEN p update. " Update
     PERFORM update record.
   WHEN p delete. " Delete
     PERFORM delete_record.
   WHEN p modify. " Modify
* Sociedad
     gwa inform banco-sociedad = p soc.
* Ejercicio
     gwa inform banco-ejercicio = p ejer.
* Período contable
     gwa inform banco-per cont = p perio.
* Acreedor
     IF p acre IS NOT INITIAL.
       gwa_inform_banco-tipo_ejec = 'A'.
     ELSEIF p deud IS NOT INITIAL.
       gwa_inform_banco-tipo_ejec = 'D'.
     ENDIF.
* Usuario responsable
     gwa inform banco-usuario res = sy-uname.
* Fecha de ejecución
     gwa inform banco-fecha eje = sy-datum.
* MODIFY en base de datos
```

```
MODIFY zib_l100000 FROM gwa_inform_banco.

* Mensajes de información
    IF sy-subrc EQ 0.
        MESSAGE i009(z_l100000).
    ELSE.
        MESSAGE i010(z_l100000).
    ENDIF.

WHEN OTHERS.

ENDCASE.

" EXECUTE_TASK
```