



**LOGALI**

# SOLUCIÓN EJERCICIO

## Subrutinas

---

SAP ABAP Programación





1. 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.

```

1  *-----*
2  * Report  Z_DEC_BANCO_L100000 *
3  *-----*
4  *-----*
5  *-----*
6  *-----*
7  *-----*
8
9  REPORT z_dec_banco_l100000.
10
11
12  INCLUDE z_dec_banco_l100000_top.
13
14  INCLUDE z_dec_banco_l100000_sel.
15
16  INCLUDE z_dec_banco_l100000_f01.
17

```

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.

```

18      INITIALIZATION.
19
20      PERFORM do_init.
21
22      c_test  = text-c01. " Test
23      c_notif = text-c02. " Notificar Manager
24

```

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

```

INCLUDE z_dec_banco_l100000_sel.

INCLUDE z_dec_banco_l100000_f01.

INITIALIZATION.
PERFORM do_init.

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

AT SELECTION-SCREEN ON p_perio.

```



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

```
*&-----*
*-----*
*&  Include              Z_DEC_BANCO_L100000_F01
*&-----*
*-----*
*&      Form  DO_INIT
*&-----*
*-----*
*      text
*-----*
*  -->  p1      text
*  <--  p2      text
*-----*
*-----*
FORM do_init.

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

ENDFORM.              " 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_f01.

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.

```
*&-----*
*-----*
*& Include      Z_DEC_BANCO_L100000_F01
*&-----*
*-----*
*&-----*
*-----*
*&      Form  DO_INIT
*&-----*
*-----*
*      text
*-----*
*-----*
*  -->  p1      text
*  <--  p2      text
```



```

*-----*
*-----*
FORM do_init.

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

ENDFORM.                    " DO_INIT
*&-----*
*-----*
*&      Form  CHECK_VALUES
*&-----*
*-----*
*      text
*-----*
*-----*
* -->  p1      text
* <--  p2      text
*-----*
*-----*
FORM check_values .

    IF p_perio LT 1 OR p_perio GT 12.
        MESSAGE e001(z_l100000).
    ENDIF.

ENDFORM.                    " CHECK_VALUES
*&-----*
*-----*
*&      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'.
            ENDIF.
            * 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_l100000 FROM gwa_inform_banco.

* Mensajes de información
    IF sy-subrc EQ 0.
        MESSAGE i002(z_l100000).
    ELSE.
        MESSAGE i003(z_l100000).
    ENDIF.

WHEN p_read. " Read

    SELECT SINGLE * FROM zib_l100000
        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_l100000).
    ENDIF.

WHEN p_update. " Update

    CASE abap_true.
        WHEN p_acre.
            * Update tipo ejecución con Acreedor
            UPDATE zib_l100000 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_l100000 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.

        WHEN p_delete. " Delete
*           Eliminación del registro
*           de base de datos
            DELETE FROM zib_l100000
            WHERE sociedad EQ p_soc
            AND ejercicio EQ p_ejer
            AND per_cont EQ p_perio.

            IF sy-subrc EQ 0.
                MESSAGE i007(z_l100000).
            ELSE.
                MESSAGE i008(z_l100000).
            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_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.

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_l100000 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_l100000 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_l100000
  WHERE sociedad EQ p_soc
    AND ejercicio EQ p_ejer
    AND per_cont  EQ p_perio.

IF sy-subrc EQ 0.
    MESSAGE i007(z_l100000).
ELSE.
    MESSAGE i008(z_l100000).
ENDIF.

ENDFORM.                    " DELETE_RECORD

```



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

```
*&-----*
*-----*
*&      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_l100000
  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_l100000).
ENDIF.

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'.
* 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
```