LOGALI

SOLUCIÓN EJERCICIO

Subrutinas con parámetros

SAP ABAP Programación

1. En la transacción del Editor ABAP modifique el programa Z_DEC_BANCO_USUARIOSAP*.

2. Empiece modularizando el código de la subrutina EXECUTE_TASK en subrutinas con parámetros. Empiece con la lógica del **CASE p_create** creando la subrutina CREATE_RECORD. La subrutina se debe crear en el mismo include Z_DEC_BANCO_USUARIOSAP_F01.

```
  CASE abap_true.

    WHEN p_create. " Create

      PERFORM create_record USING gwa_inform_banco.
```

3. Mueva todo el código del **CASE p_create** en la subrutina creada modificando el uso de la variable global GWA_INFORM_BANCO con el parámetro pasado a la subrutina.

```
*&---------------------------------------------------------------------
----*
*&      Form  CREATE_RECORD
*&---------------------------------------------------------------------
----*
*       text
*----------------------------------------------------------------------
----*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------
----*
FORM create_record USING p_wa_inform_banco TYPE zib_l100000.

* Sociedad
  p_wa_inform_banco-sociedad   = p_soc.
* Ejercicio
  p_wa_inform_banco-ejercicio  = p_ejer.
* Período contable
  p_wa_inform_banco-per_cont   = p_perio.

* Acreedor
  IF p_acre IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'A'.
* Deudor
  ELSEIF p_deud IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'D'.
  ENDIF.

* Usuario responsable
```

```
  p_wa_inform_banco-usuario_res = sy-uname.

* Fecha de ejecución
  p_wa_inform_banco-fecha_eje   = sy-datum.

* INSERT en base de datos
  INSERT zib_l1100000 FROM p_wa_inform_banco.

* Mensajes de información
  IF sy-subrc EQ 0.
    MESSAGE i002(z_l1100000).
  ELSE.
    MESSAGE i003(z_l1100000).
  ENDIF.

ENDFORM.                    " CREATE_RECORD
```

4. Siga modularizando el código de la subrutina EXECUTE_TASK en subrutinas con parámetros. En la lógica del **CASE p_read** cree la subrutina READ_RECORD. La subrutina se debe crear en el mismo include Z_DEC_BANCO_USUARIOSAP_F01.

```
    WHEN p_read. " Read

      PERFORM read_record USING gwa_inform_banco.
```

5. Mueva todo el código del **CASE p_read** en la subrutina creada modificando el uso de la variable global GWA_INFORM_BANCO con el parámetro pasado a la subrutina.

```
*&---------------------------------------------------------------------
----*
*&      Form  READ_RECORD
*&---------------------------------------------------------------------
----*
*       text
*---------------------------------------------------------------------
----*
*      -->P_GWA_INFORM_BANCO  text
*---------------------------------------------------------------------
----*
FORM read_record  USING  p_wa_inform_banco TYPE zib_l1100000.

  SELECT SINGLE * FROM zib_l1100000
          INTO p_wa_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 = ',
             p_wa_inform_banco-sociedad,
           / 'Ejercicio = ',
             p_wa_inform_banco-ejercicio,
           / 'Período contable = ',
             p_wa_inform_banco-per_cont,
           / 'Tipo ejecución = ',
             p_wa_inform_banco-tipo_ejec,
           / 'Usuario responsable = ',
             p_wa_inform_banco-usuario_res,
           / 'Fecha de ejecución = ',
             p_wa_inform_banco-fecha_eje MM/DD/YYYY.
  ELSE.
    MESSAGE i004(z_l100000).
  ENDIF.

ENDFORM.                    " READ_RECORD
```

6. Siga modularizando el código de la subrutina EXECUTE_TASK en subrutinas con parámetros. En la lógica del **CASE p_modify** cree la subrutina MODIFY_RECORD. La subrutina se debe crear en el mismo include Z_DEC_BANCO_USUARIOSAP_F01.

```
    WHEN p_modify. " Modify

      PERFORM modify_record USING gwa_inform_banco.
```

7. Mueva todo el código del **CASE p_modify** en la subrutina creada modificando el uso de la variable global GWA_INFORM_BANCO con el parámetro pasado a la subrutina.

```
*&---------------------------------------------------------------------
----*
*&      Form  MODIFY_RECORD
*&---------------------------------------------------------------------
----*
*       text
*----------------------------------------------------------------------
----*
*      -->P_GWA_INFORM_BANCO  text
*----------------------------------------------------------------------
----*
FORM modify_record  USING p_wa_inform_banco TYPE zib_l100000.

* Sociedad
  p_wa_inform_banco-sociedad   = p_soc.
* Ejercicio
  p_wa_inform_banco-ejercicio  = p_ejer.
* Período contable
```

```
  p_wa_inform_banco-per_cont    = p_perio.

* Acreedor
  IF p_acre IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'A'.
* Deudor
  ELSEIF p_deud IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'D'.
  ENDIF.

* Usuario responsable
  p_wa_inform_banco-usuario_res = sy-uname.

* Fecha de ejecución
  p_wa_inform_banco-fecha_eje   = sy-datum.

* MODIFY en base de datos
  MODIFY zib_l100000 FROM p_wa_inform_banco.

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

ENDFORM.                    " MODIFY_RECORD
```

8. 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

      PERFORM create_record USING gwa_inform_banco.

    WHEN p_read. " Read

      PERFORM read_record USING gwa_inform_banco.
```

```
    WHEN p_update. " Update

      PERFORM update_record.

    WHEN p_delete. " Delete

      PERFORM delete_record.

    WHEN p_modify. " Modify

      PERFORM modify_record USING gwa_inform_banco.

    WHEN OTHERS.

  ENDCASE.

ENDFORM.                    " EXECUTE_TASK
```

9. Todo el código del include que contiene las subrutinas.

```
*&---------------------------------------------------------------------
----*
*&  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

      PERFORM create_record USING gwa_inform_banco.

    WHEN p_read. " Read

      PERFORM read_record USING gwa_inform_banco.

    WHEN p_update. " Update

      PERFORM update_record.

    WHEN p_delete. " Delete

      PERFORM delete_record.

    WHEN p_modify. " Modify

      PERFORM modify_record USING gwa_inform_banco.

    WHEN OTHERS.

  ENDCASE.

ENDFORM.                    " EXECUTE_TASK
*&---------------------------------------------------------------------
*&      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
*&---------------------------------------------------------------------*
*&      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
*&---------------------------------------------------------------------
----*
*&      Form  CREATE_RECORD
*&---------------------------------------------------------------------
----*
*       text
*---------------------------------------------------------------------
----*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------
----*
FORM create_record USING p_wa_inform_banco TYPE zib_l100000.

* Sociedad
  p_wa_inform_banco-sociedad   = p_soc.
* Ejercicio
  p_wa_inform_banco-ejercicio  = p_ejer.
* Período contable
  p_wa_inform_banco-per_cont   = p_perio.

* Acreedor
  IF p_acre IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'A'.
* Deudor
  ELSEIF p_deud IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'D'.
  ENDIF.

* Usuario responsable
  p_wa_inform_banco-usuario_res = sy-uname.

* Fecha de ejecución
  p_wa_inform_banco-fecha_eje   = sy-datum.

* INSERT en base de datos
  INSERT zib_l100000 FROM p_wa_inform_banco.

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

ENDFORM.                    " CREATE_RECORD
*&---------------------------------------------------------------------
----*
*&      Form  READ_RECORD
*&---------------------------------------------------------------------
----*
*       text
*---------------------------------------------------------------------
```

```
*---------------------------------------------------------------------*
*       -->P_GWA_INFORM_BANCO  text
*---------------------------------------------------------------------*
FORM read_record  USING p_wa_inform_banco TYPE zib_l100000.

  SELECT SINGLE * FROM zib_l100000
          INTO p_wa_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 = ',
             p_wa_inform_banco-sociedad,
           / 'Ejercicio = ',
             p_wa_inform_banco-ejercicio,
           / 'Período contable = ',
             p_wa_inform_banco-per_cont,
           / 'Tipo ejecución = ',
             p_wa_inform_banco-tipo_ejec,
           / 'Usuario responsable = ',
             p_wa_inform_banco-usuario_res,
           / 'Fecha de ejecución = ',
             p_wa_inform_banco-fecha_eje MM/DD/YYYY.
  ELSE.
    MESSAGE i004(z_l100000).
  ENDIF.

ENDFORM.                    " READ_RECORD
*&---------------------------------------------------------------------*
*&      Form  MODIFY_RECORD
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*       -->P_GWA_INFORM_BANCO  text
*---------------------------------------------------------------------*
FORM modify_record  USING p_wa_inform_banco TYPE zib_l100000.

* Sociedad
  p_wa_inform_banco-sociedad   = p_soc.
* Ejercicio
  p_wa_inform_banco-ejercicio  = p_ejer.
* Período contable
  p_wa_inform_banco-per_cont   = p_perio.

* Acreedor
  IF p_acre IS NOT INITIAL.
    p_wa_inform_banco-tipo_ejec  = 'A'.
* Deudor
  ELSEIF p_deud IS NOT INITIAL.
```

```
    p_wa_inform_banco-tipo_ejec  = 'D'.
  ENDIF.

* Usuario responsable
  p_wa_inform_banco-usuario_res = sy-uname.

* Fecha de ejecución
  p_wa_inform_banco-fecha_eje   = sy-datum.

* MODIFY en base de datos
  MODIFY zib_l100000 FROM p_wa_inform_banco.

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

ENDFORM.                    " MODIFY_RECORD
```